دانشگاه تهران
دانشکده علوم
گروه ریاضی و علوم کامپیوتر

تولید درخت‌هایی با n گره داخلی و m برگ

نگارش
سید احسان سیدی طبری

استاد راهنما
دکتر هایده اهرابیان

رساله برای دریافت درجه کارشناسی ارشد
در
ریاضی محض گرایش کامپیوتر نظری

بهار ۱۳۸۵

برای میم

چکیده

در این پایان‌نامه مجموعه درخت‌هایی مطالعه می‌شود که دارای n گره داخلی و m گره خارجی (برگ) می‌باشند و الگوریتم‌های تولید، رتبه‌گذاری و رتبه‌گشایی این درخت‌ها در ترتیب A-Order ارایه می‌شود. به منظور کدگذاری این درخت‌ها، دنباله عددی جدیدی به نام E-Sequence ارایه شده است که اگر در ترتیب لغت‌نامه‌ایی تولید شود، درختان متناظر در ترتیب A-Order خواهند بود. الگوریتم تولید، رتبه‌گذاری و رتبه‌گشایی طراحی شده، بر اساس این دنباله‌های عددی در ترتیب لغت‌نامه‌ایی عمل می‌کنند.

درخت‌هایی با n گره داخلی و m برگ در تولید ساختارهای دوم ممکن یک رشته RNA با $۲n + m - ۲$ نوکلوتاید و $n - ۱$ جفت‌باز کاربرد دارند.

پیچیدگی زمانی الگوریتم تولید طراحی شده برابر $O(n + m)$ می‌باشد. تنها الگوریتم تولید موجود دارای پیچیدگی زمانی $O(nm)$ است. الگوریتم رتبه‌گذاری و رتبه‌گشایی دیگری برای این درختان وجود ندارد.

پیشگفتار

یکی از مسائل مورد توجه محققان علوم کامپیوتر، طراحی الگوریتم‌های تولید اشیای ترکیبیاتی است. در حالت کلی این مسئله می‌تواند به‌عنوان طراحی یک الگوریتم کارا برای تولید تمام عناصر یک شی ترکیبیاتی داده شده، تعریف گردد [۱۰]. تولید برخی از اشیای ترکیبیاتی مانند جایگشت‌ها، ترکیب‌ها، افرازها، درخت‌ها و درخت‌های فراگیر بیشتر مورد توجه قرار گرفته‌اند.

تاکنون الگوریتم‌های متعددی برای تولید درخت‌های باینری و درخت‌های k-تایی با تعداد گره‌های ثابت ارایه شده‌است [۲۱ ،۱۸ ،۱۶ ،۱۴ ،?]. در اغلب این الگوریتم‌ها، ابتدا درخت‌ها با روش خاصی به‌صورت یک دنباله عددی یا حرفی کدگذاری می‌شوند و سپس این دنباله‌ها با یک ترتیب مشخص تولید می‌گردند [۱۲]. ترتیبی که دنباله‌ها براساس آن تولید می‌شوند باعث می‌گردد که درخت‌های متناظر با آن‌ها نیز در یک ترتیب خاص قرار گیرند. برای درخت‌ها، دو ترتیب توسط زکس[۱] [۲۱] تعریف شده است که آن‌ها را ترتیب‌های A-Order و B-Order می‌نامند و برای دنباله‌ها نیز ترتیب‌های متفاوتی وجود دارد که ترتیب قاموسی و ترتیب گِری‌کد از مشهورترین آن‌ها هستند. حالت ایده‌آل در تولید دنباله‌های متناظر با درخت‌ها به این صورت است که دنباله‌ها در ترتیب قاموسی یا گری‌کد تولید شوند در حالی که درخت‌های متناظر با آن‌ها در ترتیب A-Order یا

[۱] S.Zaks

B-Order باشند.

در کنار الگوریتم‌های تولید، الگوریتم‌های رتبه‌گذاری دنباله، بازیابی دنباله از رتبهٔ دنباله (رتبه‌گشایی)، کدگذاری درخت و بازیابی درخت از روی کد (کدگشایی) نیز حائز اهمیت هستند. در اکثر مقالات ارائه شده برای تولید درخت‌ها، این الگوریتم‌ها نیز مطرح شده و بحث گردیده‌اند.

در این پایان‌نامه، توجه خود را به مجموعه ایی از درخت‌های مرتب معطوف می‌کنیم که دارای تعداد گره‌های داخلی ثابت و گره‌های خارجی (برگ‌های) ثابت هستند. این درخت‌ها پیشتر توسط پالو[۲] به کمک نشاندن در درختان k-تایی در ترتیب B-Order تولید شده‌اند [۱۴]. پالو الگوریتمی برای رتبه‌گذاری یا رتبه‌گشایی دنباله‌ها ارایه ننموده است. الگوریتم ارایه شده در این پایان‌نامه این درخت‌ها را در ترتیب A-Order با پیچیدگی زمانی کمتر تولید می‌کند. همچنین، الگوریتم‌های رتبه‌گذاری و رتبه‌گشایی این درخت‌ها طراحی شده‌اند.

ساختار این پایان‌نامه به‌صورت زیر است: بخش ۱ شامل برخی از مفاهیم اولیه است که در رابطه با این اشیای ترکیبیاتی نیاز داریم. دربخش ۲ تولید درخت‌ها با ارایه روش زکس شرح داده می‌شود. در بخش ۳ روش پالو برای تولید درخت‌هایی با تعداد گره‌های ثابت و برگ‌های ثابت را در ترتیب B-Order ارایه می‌کند. در بخش ۴ الگوریتم‌های جدیدی برای تولید، رتبه‌گذاری و رتبه‌گشایی این درخت‌ها ارایه می‌شود.

---

[۲] J.M.Pallo

تشکر و قدردانی

در طول دوره‌ایی که مقطع کارشناسی ارشد علوم کامپیوتر را در دانشگاه تهران می‌گذارانده‌م، محبتهای بزرگوارانی که در این جا از آنها یاد می‌نمایم، سبب تغییرات شگرفی در زندگی من شد.

استاد ارجمندم خانم دکتر اهرابیان نه تنها در این پایان‌نامه، که در تمام امور تحصیل و پژوهش راهنمای من بوده‌اند.

استاد گرانقدر جناب آقای دکتر نوذری که راهنمایی‌های ایشان در این پایان‌نامه و سایر امور زندگی تاثیرگذار بوده‌است.

و مادرم و پدرم که عزیزند و تنها پشتوانه‌های من در طول زندگی.

احسان طبری

فهرست

۵ واژه‌نامه انگلیسی به فارسی

فهرست شکل‌ها

فهرست جدولها

فهرست الگوریتم‌ها

۱

مفاهیم اولیه

در این فصل مفاهیم اساسی مورد نیاز فصل‌های آتی و لزوم پرداخت به این موضوع ارایه می‌شود. در بخش اول تعاریف مربوط به گراف و درختان ارایه می‌شود[۵، ۴، ۸]. بخش ۲.۱ مفاهیم ترکیبیاتی مورد نیاز را بیان می‌کند[۱۰، ۱۳]. بخش بعد به الگوریتم‌ها و نرخ رشد آنها می‌پردازد[۴] و در نهایت به شرح کاربرد موضوع این پایان‌نامه پرداخته می‌شود[۲۰]. تعاریف و نمادگذاری‌هایی که در این فصل ارایه می شود، در سایر فصول استفاده خواهد شد.

۱.۱ تعاریف و نحوه نمایش

۱.۱.۱ گراف

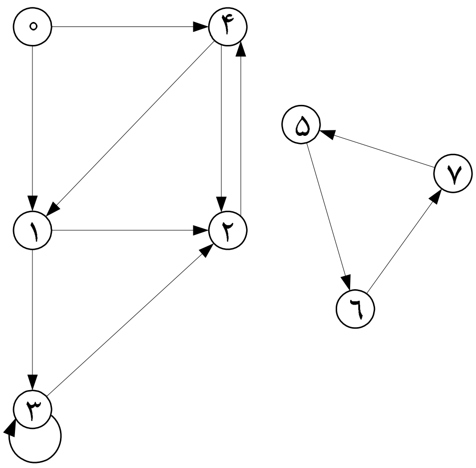
گراف جهت‌دار G زوج (V, E) است به گونه‌ایی که V مجموعه‌ایی غیرتهی و متناهی است و E یک رابطه دودویی بر روی مجموعه V است. مجموعه V مجموعه رئوس گراف G و عناصر آن رأس ها یاگره‌های آن می باشند. مجموعه E مجموعه یال‌های گراف G و اعضای آن یال‌ها یا لبه های آن نامیده می شود. در یک گراف غیرجهت‌دار مجموعه یال‌ها به جای زوج های مرتب، از زوج‌های نامرتب تشکیل می‌شوند.

برای سادگی اغلب گراف ها را به صورت تصویری نمایش می‌دهند. در این نمایش، رئوس را با دایره‌های کوچک (نام‌گذاری شده) ویال‌ها را با خط یا پیکان نمایش می‌دهیم. پیکان‌ها نمایانگر جهت یال‌ها می‌باشند. شکل ۱−۱ نمایش گرافیکی گرافی با هشت گره و یازده یال است. مجموعه گره‌های این گراف $V=\{۰, ۱, ۲, ۳, ۴, ۵, ۶, ۷\}$ و مجموعه

$$E = \{(۰, ۱), (۰, ۴), (۱, ۲), (۱, ۳), (۲, ۴), (۳, ۲), (۴, ۱), (۴, ۲), (۵, ۶), (۶, ۷), (۷, ۵)\}$$

می‌باشد.

بسیاری از تعاریف و قضایا برای گراف‌های جهت‌دار و ساده یکسان می باشند. اگر گراف $G = (V, E)$ یک گراف جهت‌دار باشد، یک یال $e \in E$ زوج مرتبی مانند (v_s, v_t) است با این شرط که $v_s, v_t \in V$؛ می‌گوییم e گره v_s را به گره v_t متصل نموده است و v_t (از طریق یال e) مجاور v_s است. یال e از v_s خارج شده و به v_t وارد می‌شود. v_s را مبدا یال و



شکل ۱—۱: نمایش گرافیکی گراف $G_۱$

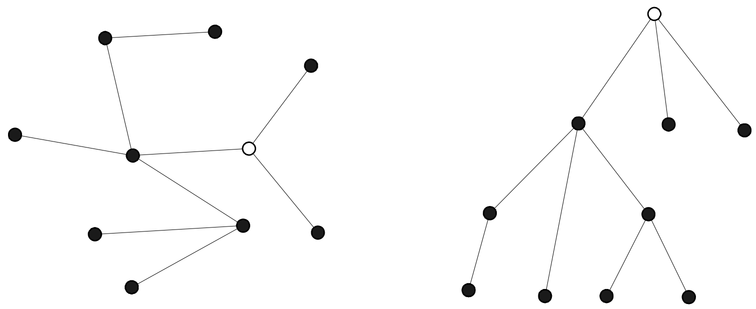
v_t را مقصد یال می‌نامیم. اگرمبدا و مقصد یالی یک گره باشد، به آن حلقه می‌گوییم. همسایگان گره v که با $N(v)$ نمایش داده می‌شود، مجموعه همه رأس‌هایی هستند که به v متصل می‌باشند. مجموعه $N(v)$ را می‌توان به دو مجموعه $N_i(v)$ و $N_o(v)$ تقسیم نمود. $N_i(v)$ شامل همه همسایگانی است که v مجاور آن‌هاست و $N_o(v)$ شامل همه گره‌هایی است که مجاور v هستند. این دو مجموعه لزوما ناسازگار نیستند. درجه رأس v در یک گراف که با $deg(v)$ نشان داده می‌شود، برابر با مجموع تعداد یال‌هایی است که به v متصل می‌باشند. این رابطه عجیب از آن روست که یال‌های حلقوی دو بار در درجه گراف جهت دار شمرده می‌شوند. در یک گراف ساده درجه رأس v تهداد یال‌هایی است که به آن وارد می‌شود. در یک گراف جهت‌دار درجه ورودی رأس v اندازه مجموعه $N_i(v)$ است که با $deg_i(v)$ نشان داده می‌شود و درجه خروجی رأس v اندازه مجموعه $N_o(v)$ است که با $deg_o(v)$ نشان داده می‌شود.

یک مسیر مانند p دنباله‌ایی به شکل $(v_۰, v_۱, \ldots, v_k)$ با شرط $k > ۰$ از رئوس است

که برای هر $i \in \{۱, \dots, k\}$ زوج $(v_{i-۱}, v_i)$ یک یال باشد. این مسیر گره $v_۰$ را به گره v_k متصل می‌نماید و طول آن k می‌باشد. مسیر ساده، مسیری است که برای هر زوج i و j از $\{۱, \dots, k-۱\}$ رابطه $v_{i-۱} \neq v_i$ صدق کند. دور، مسیر ساده‌ایی است که یک راس را به خودش متصل می‌کند. یک حلقه، دوری به طول یک می‌باشد. گراف غیرجهت‌داری را همبند گویند اگر یک مسیر بین هر دو رأس آن وجود داشته باشد. یک گراف جهت‌دار را قویا همبند گویند اگر برای هر دو راس انتخاب شده، مسیری از هر کدام به دیگری وجود داشته باشد. یک گراف جهت‌دار همبند است اگر بدون در نظر گرفتن جهت یال‌ها، بین هر دو رأس آن یک مسیر وجود داشته باشد. گراف $G = (V, E)$ را مسطح گویند اگر بتوان آن را بر روی یک سطح به گونه‌ایی رسم نمود که یال‌های آن تقاطعی جز رأس‌ها نداشته باشند.

۲.۱.۱ درخت

یک درخت آزاد به صورت کلی، به یک گراف ساده همبند بدون‌دور اطلاق می‌شود. به گراف بدون‌دور و غیر جهت دار جنگل می‌گویند. درخت ریشه دار، درختی است که در آن یکی از رأس‌ها از بقیه رأس‌ها متمایز گشته باشد. به آن رأس متمایز شده، ریشه اطلاق می‌شود. معمولا به رأس‌های یک درخت ریشه دار، گره گفته می‌شود. شکل ۱—۲(الف) نمایش یک درخت آزاد و شکل ۱—۲(ب) همان درخت با در نظر گرفتن گره سفید به عنوان ریشه است.



(الف) (ب)

شکل ۱—۲: مثال‌هایی از نمایش درخت

اگر x گره‌ایی در درخت ریشه‌دار T با ریشه r باشد، هر گره‌ایی مانند y که در یک مسیر منحصر به فرد از ریشه تا x قرار دارند را جد x می‌گویند. اگر y جد x باشد، x نواده y است. هر گره جد و نواده خودش می‌باشد. تعداد گره‌های درخت T با برابر تعداد نوادگان r می‌باشد که با $|T|$ نشان داده و به آن اندازه یا وزن درخت T گفته می‌شود.

اگر y جد x باشد و $x \neq y$، به y جد کامل و x نواده کامل گفته می‌شود. زیردرختی با ریشه x درختی است که از نوادگان x به وجود می‌آید.

اگر در درخت T در مسیر ریشه به گره x، آخرین یال (y, x) باشد، y را پدر x و x را فرزند y می‌گویند. ریشه تنها گره‌ایی از درخت است که پدر ندارد. اگر دو گره پدرهای یکسان داشته باشند، آن دو را برادر یا همزاد می‌گویند. به گره‌ایی که فرزند نداشته باشد، برگ یا گره انتهایی می‌گویند. به گره‌ایی که برگ نباشد، گره داخلی گفته می‌شود. درجه گره x در یک درخت ریشه‌دار تعداد فرزندان آن است و با $deg(x)$ نشان داده می‌شود. مقصود از $deg(T)$ درجه ریشه درخت T می‌باشد. به طول مسیر از ریشه تا گره x را عمق x می‌گویند. ارتفاع یک گره طول بلندترین مسیر از آن گره به یک برگ می‌باشد. ارتفاع

درخت به ارتفاع ریشه آن گفته می‌شود. بدیهی است که این مقدار با حداکثر مقدار عمق برگ‌های درخت برابر است. یک درخت مرتب به درخت ریشه داری گفته می‌شود که فرزندان هر گره مرتب شده باشند. در یک درخت مرتب، زیردرخت‌های گره‌ایی مانند x را می‌توان به صورت $\{T_i : i \in \{۱, \dots, deg(x)\}\}$ نشان داد؛ به گونه‌ایی که $T_۱$ اولین زیردرخت و $T_{deg(x)}$ آخرین زیر درخت x می‌باشند. در نمایش گرافیکی درخت مرتب، زیر درخت‌های یک گره را به ترتیب از چپ به راست رسم می‌نمایند.

درخت دودویی، درختی ریشه‌دار و مرتب است که به صورت بازگشتی به راحتی می‌توان آن را تعریف نمود: هر درخت دودویی یا تهی است یا دارای یک ریشه است که دو زیردرخت باینری در راست و چپ دارد. بدیهی است که درجه هر گره حداکثر ۲ است. یک درخت دودویی را منظم می‌گویند اگر درجه هر گره صفر یا دو باشد.

درخت k-تایی، از تعمیم درخت دودویی به دست می‌آید. در این نوع درختان هر گره k زیر درخت دارد (که البته می‌تواند تهی باشد). یک درخت k-تایی را منظم می‌گویند اگر درجه هر گره صفر یا k باشد. در یک درخت k-تایی منظم، گره‌های بدون فرزند را گره خارجی می‌گویند.

پیمایش درخت، فرایند مشاهده همه گره‌های یک درخت است. این فرایند، که قدم زدن بر درخت نیز گفته می‌شود، یک دنباله خطی از گره‌های درخت ایجاد می‌کند. این پیمایش‌ها، بر اساس ترتیب مشاهده گره‌ها، دسته بندی می‌شوند. از پیمایش‌های مهم می‌توان پیمایش پیش‌ترتیب و پس‌ترتیب را نام برد. در پیمایش پیش‌ترتیب، هر گره قبل از پیمایش زیر درختانش مشاهده شده و به دنباله خروجی اضافه می‌شود. در پیمایش پس‌ترتیب، هر گره پس از پیمایش زیر درختانش مشاهده می‌شود. در درخت‌های مرتب، پیمایش زیردرختان

از اولین زیردرخت یک گره به آخرین زیردرخت آن (از T_1 به $T_{deg(x)}$) انجام می‌شود. برای درختان مرتب، پیمایش معکوس نیز متصور است. بدین صورت که فرزندان یک گره عکس ترتیبشان پیمایش می‌شوند. برای درختان دودویی پیمایش مهم دیگری به نام میان‌ترتیب تعریف می‌شود که در آن هر گره پس از پیمایش زیردرخت چپ و قبل از پیمایش زیر درخت راست مشاهده می‌شود.

۲.۱ مفاهیم ترکیبیاتی

هر ساختار ترکیبیاتی، CS، مجموعه‌ایی از اشیا یا عناصری است که یک خاصیت ریاضی خاص را حفظ می‌کند. شمارش و تولید اشیا یک ساختار دو مساله مهم ترکیبیات و علوم کامپیوتر می‌باشند؛ که به معنی ایجاد همه اشیا آن ساختار هستند. عموما، مابین اشیای ترکیبیاتی یک ساختار، یک ترتیب خطی در نظر گرفته می‌شود؛ که برای هر دو شی متمایز عضو ساختار، یکی ماقبل دیگری، در آن ترتیب خطی، قرار می‌گیرد. به زبان دیگر، مرتب سازی، پیدا کردن یک رابطه یک‌به‌یک و پوشا بین اعداد $\{0, \ldots, |CS| - 1\}$ و اشیا یک ساختار ترکیبیاتی است. به هر شی، رتبه‌ایی داده می‌شود که برابر با تعداد اشیا ماقبل آن در ترتیب تعریف شده می‌باشد.

تولید اشیای یک ساختار در یک ترتیب تعریف شده معمولا توسط سه الگوریتم مستقل

انجام می‌شود: (۱) الگوریتم ایجاد شی بعدی که با در دست داشتن یک شی، شی بعدی را در همان ترتیب تولید می‌کند. (۲) الگوریتم رتبه‌گذاری که با در دست داشتن یک شی، رتبه آن را محاسبه می‌کند و (۳) الگوریتم رتبه گشایی که بر اساس رتبه، شی متناظر آن رتبه را تولید می‌نماید.

۱.۲.۱ ترتیب درختان

درختان همانند گراف‌ها اشیایی ترکیبیاتی می‌باشند. برای ترتیب درخت‌های مرتب ریشه‌دار دو تعریف مستقل وجود دارد. تعاریف ۱.۱ و ۲.۱ دو ترتیب برای درختان مرتب ریشه‌دار ارایه می‌کند. ترتیب اول به A-Order یا ترتیب طبیعی معروف است و دومی را با عنوان B-Order می‌شناسند.

تعریف ۱.۱ برای دو درخت مرتب ریشه‌دار T و T' گویند $T \prec_A T'$ اگر:

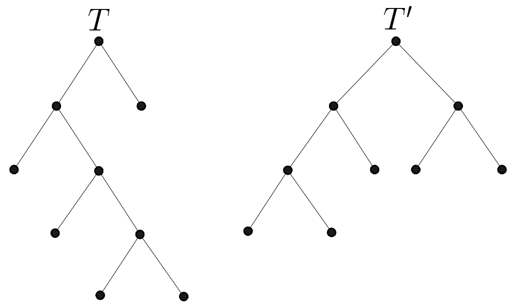
۱) $|T| < |T'|$، یا

۲) $|T| = |T'|$ و با فرض i بزرگترین مقداری که برای $j = ۱, \ldots, i$ وجود داشته باشد $T_j = T'_j$، آنگاه باید:

(a) $i = deg(T)$، یا

(b) $T_{i+۱} \prec_A T'_{i+۱}$

$\triangledown$



شکل ۱—۳: مثالی از ترتیب دو درخت $T \prec_{B} T'$ و $T' \prec_{A} T$

تعریف ۲.۱ برای دو درخت مرتب ریشه‌دار T و T' گویند $T' \prec_{B} T$ اگر:

۱) $deg(T) < deg(T')$، یا

۲) $deg(T) = deg(T')$ و i بزرگترین مقداری باشد که برای $j = ۱, \ldots, i$ وجود داشته باشد $T_j = T'_j$، آنگاه باید:

(a) $i = deg(T)$، یا

(b) $T_{i+۱} \prec_{B} T'_{i+۱}$

$\bigtriangledown$

لازم به ذکر است که $B - Order$ از اطلاعات محلی هر گره (درجه گره) برای مرتب سازی استفاده می‌کند، در حالی که $A - Order$ اطلاعات global درخت (تعداد فرزندان هر گره) را به کار می‌گیرد. شکل ۱—۳ دو درخت دودویی منظم با ۴ گره داخلی را به نام‌های T و T' نشان می دهد که $T \prec_{B} T'$ اما $T' \prec_{A} T$.

۳.۱ الگوریتم و پیچیدگی زمانی

یک الگوریتم مجموعه‌ای متناهی از دستورالعمل‌های تعریف شده‌ایی است که از یک حالت آغازین شروع می‌شود و پس از برآوردن هدف خاصی به یک حالت پایانی می‌رسد. کنوث[۱] [۸] پنج خاصیت مهم الگوریتم‌ها را چنین برشمرده است:

۱) یک الگوریتم بایست پس از یک تعداد قطعی گام پایان یابد.
۲) همه گام‌های یک الگوریتم باید به طور صریح و دقیق تعریف شده باشد.
۳) ورودی‌های الگوریتم (صفر یا بیشتر) باید مشخص شده باشند.
۴) خروجی‌های الگوریتم (یک یا بیشتر) باید مشخص شده باشند.
۵) الگوریتم باید انجام شدنی باشد؛ بدین معنی که عملیات یک الگوریتم باید ساده باشند به گونه‌ایی که قابلیت انجام توسط یک شخص در زمان قطعی وجود داشته باشد. این زمان قطعی می‌تواند یک میلیارد سال باشد.

۱.۳.۱ تحلیل الگوریتم

الگوریتم‌ها از جهات مختلفی مورد بررسی قرار می‌گیرند ولی دو معیار مهمی که در بررسی و مقایسه الگوریتم‌ها بیشتر مدنظر قرار می‌گیرد، زمان اجرا و میزان حافظه مورد نیاز برای یک مجموعه ورودی مشخص است.

برای مقایسه الگوریتم‌ها معمولا از تابعی مانند $f(n)$ که به آن تابع پیچیدگی گفته می‌شود، استفاده می‌گردد. این تابع بیان‌کننده میزان زمان یا میزان حافظه مصرفی یک الگوریتم

[۱] D.E. Knuth

به ازاء یک ورودی با اندازه n است. معمولا تابع پیچیدگی را برای بدترین حالت یا حالت متوسط محاسبه می‌کنند. بدترین حالت زمانی است که الگوریتم بیشترین زمان اجرا و یا بیشترین حافظه مصرفی را دارد. در حالت متوسط زمان الگوریتم را برای تمام ورودی‌های ممکن محاسبه نموده و متوسط آن‌ها را برای ارزیابی الگوریتم مورد استفاده قرار می‌دهند. برای بیان نرخ رشد توابع پیچیدگی از نماد O استفاده می‌گردد. در واقع وقتی می‌گوییم $f(n) = O(g(n))$ منظور این است که $g(n)$ یک کران بالا برای نرخ رشد $f(n)$ است یا به‌عبارت دیگر

$$f(n) = O(g(n)) : \; \exists\, c > ۰,\; n_۰ > ۰ \mid \forall\, n \geq n_۰ \;\; ۰ \leq f(n) \leq cg(n).$$

کمترین نرخ رشد برای الگوریتم‌ها $O(۱)$ است. به‌همین دلیل در طراحی الگوریتم‌ها، این کلاس از الگوریتم‌ها بیشتر مورد توجه قرار می‌گیرند و پس از آن‌ها الگوریتم‌هایی با نرخ رشد $O(\log n)$ و $O(n)$ دارای اهمیت هستند.

۴.۱ زیست‌شناسی مولکولی و ساختار دوم RNA

یکی از مسایل پایه‌ایی زیست‌شناسی شناخت وراثت است. در ۱۸۶۵ مندل[۲] یک مدل انتزاعی ریاضی ارایه نمود که در آن ژن‌ها واحدهای پایه‌ایی وراثت بودند. در ۱۹۴۴ کشف شد که ژن‌ها از DNA ها ساخته می‌شوند و در ۱۹۵۳ ساختاردومارپیچی DNA کشف

[۲] Mendel

شد. مولکول‌های یک سلول به دو دسته تقسیم می‌شوند: مولکول‌های بزرگ و مولکول‌های کوچک. مولکول‌های بزرگ سلولی، که ابرمولکول نیز خوانده می‌شوند، خود سه نوع می‌باشند: DNA، RNA و پروتین ها.

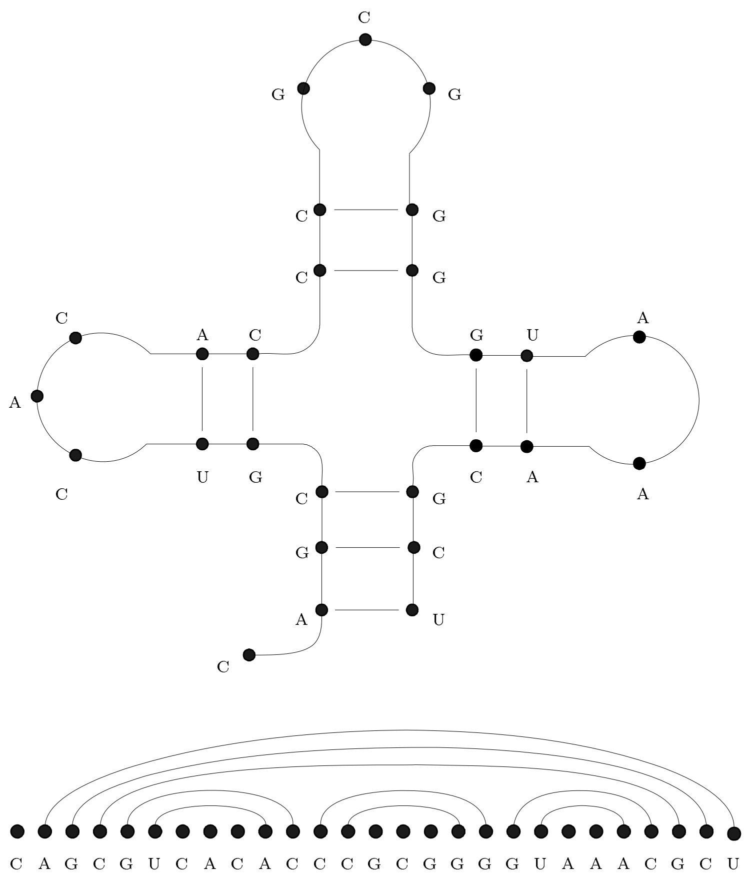
DNA مولکول پایه‌ایی وراثت است و یک پلی‌مر ساخته شده از مولکول‌های کوچکتر به نام نوکلوتاید می‌باشد. در مجموع چهار نوکلوتاید وجود دارند که یک DNA را می سازند.

شکل ابرملکول‌ها و پیچیدگی ساختاری آنها نوع تراکنشات درون سلولی و در نتیجه فرایند حیات را توصیف می‌کند.

یک RNA تک‌لایه به صورت یک دنباله خطی از ریبونوکلوتایدها مانند $\mathbf{a} = a_۱ a_۲ \ldots a_n$ نمایش داده می‌شود. دنباله $\mathbf{a}$ ساختار اصلی RNA نامیده می‌شود. هر a_i توسط یکی از چهار نوکلوتاید پایه‌ایی A، C، G و U مشخص می‌شود. این سازنده‌ها می‌توانند با هم جفت شده و یک جفت‌باز را تشکیل دهند. مثلا A با U و C با G می‌تواند یک جفت‌باز را تشکیل دهد. به این‌ها جفت‌های واتسن—کریک[۳] گفته می‌شود. به‌علاوه امکان جفت شدن نوکلوتایدهای G و U هم وجود دارد. ساختار دوم RNA گراف مسطحی است که این گزاره را براورده سازد: اگر a_i با a_j و a_k با a_l جفت شوند و $i < k < j$، در آن صورت وجود داشته باشد $i < l < j$.

در شکل ۱—۴(الف) برای نمونه، یک ساختار برگ‌شبدری برای رشته RNA معادل دنباله $\mathbf{a} = \mathrm{C\ A\ G\ C\ G\ U\ C\ A\ C\ C\ C\ G\ C\ G\ G\ G\ U\ A\ A\ A\ C\ G\ C\ U}$ نشان داده شده‌است. شکل ۱—۴(ب) همان رشته و همان ساختار دوم را به گونه‌ایی نشان داده است که ساختار اصلی در امتداد محور افقی و جفت‌بازها به صورت کمان رسم شده‌اند.

[۳] Watson-Crick



شکل ۱—۴: دو نمایش از ساختار دوم RNA

۱.۴.۱ ارتباط RNA با ترکیبات

با توجه به اهمیت ساختارهای دوم RNA و تاثیر آن در عملکرد مولکول، تولید این ساختارها، از اهمیت بالایی برخوردار می‌باشند. در این قسمت، نشان داده می‌شود که برای تولید همه ساختارهای دوم ممکن یک رشته RNA کافی‌است، درختان خاصی را تولید نمود. بنابراین،

با هدف شمارش و تولید همه توپولوژی‌های ممکن یک رشته RNA، بدون توجه به بازهای تشکیل دهنده آن، و بدون در نظر گرفتن امکان تولید جفت‌باز بین هر دو نوکلوتاید، تولید ساختارهای دوم یک دنباله بررسی می‌شود.

اگر $S(n)$ مجموعه ساختارهای دوم ممکن برای یک رشته RNA به طول n باشد و $s(n) = |S(n)|$؛ می‌توان نشان داد که [۲۰]

$$s(n+۱) = s(n) + s(n-۱) + \sum_{k=۱}^{n-۲} s(k)s(n-k-۱),$$

و همچنین هنگامی که $n \longrightarrow \infty$

$$s(n) \sim \sqrt{\frac{۱۲ + ۷\sqrt{۵}}{۸\pi}} n^{-۳/۲} \left(\frac{۳ + \sqrt{۵}}{۲}\right)^n . \qquad (۱)$$

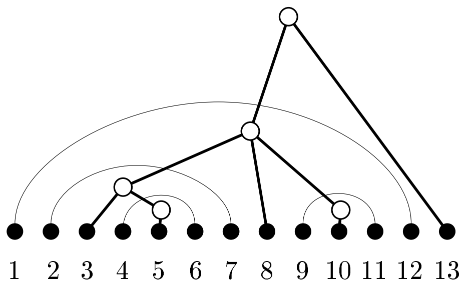
تقریب (۱) به وضوح نشان می‌دهد که تعداد ساختارهای دوم ممکن برای یک رشته RNA با طول n بسیار زیاد می‌باشد. از این رو، برای محدودسازی تولید ساختارهای دوم یک رشته RNA، در یک رویکرد، تعداد جفت‌بازها را محدود شده در نظر می‌گیرند. به عبارت دیگر، ساختارهای دومی را تولید می‌نمایند که دقیقا k جفت باز داشته باشند.

حال اگر $S(n,k)$ مجموعه ساختارهای دوم یک دنباله RNA با طول n و دقیقا k جفت‌باز باشد، و $s(n,k) = |S(n,k)|$؛ می‌توان به سادگی نشان داد که برای همه n ها $s(n,۰) = ۱$ بوده و برای $k \geqslant n/۲$ نیز $s(n,k) = ۰$ می‌باشد. همچنین برای $n \geqslant ۲$ وجود دارد [۲۰]

$$s(n+۱, k+۱) = s(n, k+۱) + \sum_{j=۱}^{n-۱} \left[\sum_{i=۰}^{k} s(j-۱, i)s(n-j, k-i)\right],$$

و بدیهی است که

$$s(n) = \sum_{k=۰}^{\lfloor n/۲ \rfloor} s(n,k).$$



شکل ۱—۵: تبدیل ساختار دوم RNA به یک درخت مرتب

واترمن[۴] با ارایه یک روش تبدیل، نشان داده است که می‌توان به هر یک از ساختارهای دوم یک RNA با طول n و k جفت‌باز، درختی متناظر نمود. درخت‌هایی که براساس روش واترمن از روی اعضای مجموعه $S(n,k)$ ساخته می‌شوند، درخت‌هایی مرتب با $k+۱$ گره داخلی و $n-۲k$ گره خارجی (برگ) می‌باشند. این تبدیل، یک رابطه یک‌به‌یک و پوشا از مجموعه $S(n,k)$ به درختانی با $k+۱$ گره داخلی و $n-۲k$ برگ است.

در این روش، برای هر یک از اعضای $S(n,k)$ به درختی معادل تبدیل می‌شود. برای این تبدیل، ابتدا ساختار اصلی دنباله RNA در امتداد محور افقی و جفت‌بازها به صورت کمان رسم می‌شوند. سپس، چنانچه در شکل ۱—۵ نمایش داده شده است، ریشه درخت، در بالای گراف RNA قرار داده می‌شود و در درون هر کمان نیز یک گره داخلی درج می‌شود. سپس، هر گره داخلی به گره‌ایی که در حلقه‌ایی بالاتر قرار دارد متصل می‌شود. سرانجام، هر گره به بازهای منفرد درون حلقه خود متصل می‌شود.

[۴] M.S. Waterman

۲

کدگذاری و تولید درختان

در این فصل مفاهیم کدگذاری درختان بررسی می‌شود و ترتیب کدهای متناظر با درختان و کدهای مطلوب مطالعه می‌گردد. ازاین رو کدگذاری‌های مهم و خواص اساسی آنها مرور می‌گردد. تمرکز اصلی این فصل به کدگذاری‌های درختان k-تایی [۱۴، ۱۸، ۲۱] خواهد بود. سپس تولید درخت‌ها به کمک دنباله‌ها دقیق‌تر مورد بررسی قرار خواهد گرفت.

۱.۲ کدگذاری و تولید درخت‌ها

در بسیاری از کاربردها، همانند کاربردهای ترکیبیاتی، ساختار تشکیل دهنده درخت‌ها بدون توجه به داده‌هایی که ممکن است در آن ذخیره شده باشد، مورد بررسی قرار می‌گیرند. در اختیار داشتن نمایشی از درخت بدون در نظر گرفتن اشاره گرها و ساختارهای اطلاعاتی سازنده آن، برای این نوع کاربردها بسیار مطلوب است. در چنین کاربردهایی، درخت‌ها را با ساختاری متفاوت از روش‌های ساختمان‌های داده‌ایی متداول، که معمولا مملو از اشاره‌گرها است، نشان داده و ذخیره می‌کنند. به این روش‌ها در حالت کلی کدگذاری درخت و به نمایش حاصل از روش اتخاذ شده کد متناظر آن گفته می‌شود. کدهای متناظر درخت‌ها معمولا به شکل دنباله‌ایی از بیت‌ها، اعداد صحیح یا کاراکترها می‌باشند.

هر یک از روش‌های کدگذاری باید دارای سه خصوصیت اصلی باشد: (۱) باید به ازای هر درخت یک کد متناظر وجود داشته باشد و همچنین به ازای هر کد تنها یک درخت وجود داشته باشد. (۲) مجموعه کدهای متناظر با یک مجموعه درختان خاص باید بدون هیچ گسستگی باشند و (۳) الگوریتم مناسبی برای تبدیل یک کد به درخت متناظرش و بالعکس وجود داشته باشد. در ضمن، بسیار مناسب است که در صورتی که درختان یک مجموعه در یک ترتیب خاص مانند A-Order یا B-Order باشند، کدهای متناظر آنها در ترتیب لغت‌نامه‌ایی یا ترتیب معکوس لغت‌نامه‌ایی باشند و یا خاصیت گِرِی داشته باشند.

گونه‌های بسیار متنوعی از روش‌های کدگذاری درخت‌ها در متون ارایه شده است. بسیاری از این روش‌های کدگذاری مخصوص درختان دودویی با یک تعداد گره داخلی معین است. این کدگذاری‌ها دنباله‌هایی بیتی [۲۱]، صحیح [۲، ۱۳، ۱۷، ۲۱] یا حرفی[۱۵] تولید می‌نمایند. منبع [۱۲] به بررسی و مقایسه برخی از این کدگذاری‌ها می‌پردازد و [۲۳] به

تبدیل آن‌ها به یکدیگر می‌پردازد. این کدگذاری‌ها عموما به درخت‌های k-تایی نیز تعمیم می‌یابند[۱، ۳، ۹، ۱۴، ۱۸، ۱۹، ۲۱]. در [٦] روشی برای کدگذاری درخت‌های 2-3 Tree معرفی شده است. همچنین [۷] روشی برای کدگذاری درخت‌های B-Tree و [۱۱] روشی برای کدگذاری درخت‌های AVL ارایه کرده است.

چنانچه در بخش ۲.۱ آمده است، درخت به عنوان یک ساختار ترکیبیاتی شمارش و تولید می‌شوند. در این تولید، الگوریتم‌های ایجاد شی بعدی، رتبه‌گذاری و رتبه‌گشایی بر روی کدهای متناظر درختان عمل می‌کنند. در حقیقت، روش‌های کدگذاری مستقل از تولید درختان نمی‌باشند و رابطه تنگاتنگی بین روش‌های کدگذاری و الگوریتم‌های تولید وجود دارد.

برای آشنایی بیشتر با مفاهیم کدگذاری و تولید درخت‌ها، در این فصل روش کدگذاری و تولید درخت های k-تایی منظم با n گره داخلی را بررسی می‌گردد. از این رو، در این فصل، روش زکس[۱] در تولید این درخت‌ها شرح داده می‌شود.

زکس دو دنباله به نام‌های x-Sequence و z-Sequence برای کدگذاری درخت‌های -kتایی منظم ارایه نموده است. دنباله x یک دنباله بیتی است. و یک درخت k-تایی منظم با n گره داخلی را با kn بیت کدگذاری می‌کند. دنباله z، که دنباله‌ایی صحیح می‌باشد، بر اساس دنباله x ساخته می‌شود. الگوریتم‌های زکس براساس دنباله z ارایه شده‌اند. این الگوریتم‌ها درخت‌های فوق را براساس عکس ترتیب B-Order تولید می‌کنند. در بخش ۲.۲ این دنباله ها و خصوصیات آنها ارایه می‌شود. بخش ۳.۲ الگوریتم تولید زکس به همراه تحلیل آن ارایه می‌دهد. سرانجام، بخش آخر روش‌های رتبه‌گذاری و رتبه‌گشایی بر

[۱] S. Zaks

اساس دنباله z-Sequence را بررسی می‌کند.

۲.۲ کدگذاری درخت‌های k-تایی توسط دنباله‌های زَکس

در این بخش دنباله‌های x و z برای کدگذاری یک درخت k-تایی منظم ارایه می‌شود. روابط، تعاریف و عملگرهای لازم برای بخش‌های بعد ارایه می‌شود.

در این پایان‌نامه فرض خواهیم کرد که $T_{n,k}$ مجموعه همه درخت‌های k-تایی منظم با n گره داخلی باشد. بنابراین

$$t_{n,k} = |T_{n,k}| = \frac{1}{kn+1}\binom{kn+1}{n}.$$

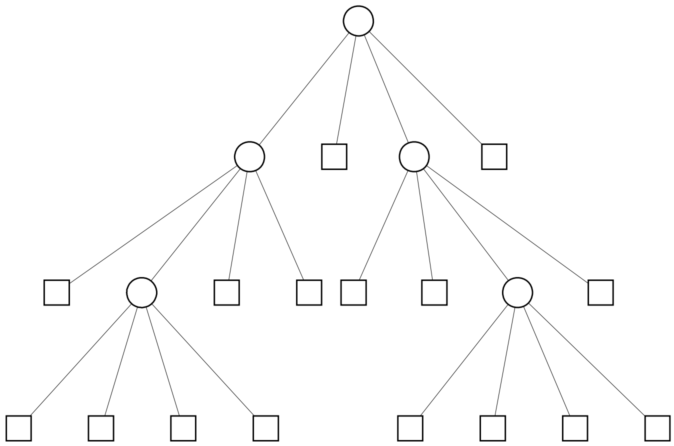
بدیهی است که اگر T عضوی از مجموعه $T_{n,k}$ باشد، تعداد برگ‌های (گره‌های خارجی) درخت T برابر $n(k-1)+1$ خواهد بود.

برای یک درخت k-تایی منظم مانند T، اگر گره‌های داخلی با عدد یک و گره های خارجی با عدد صفر برچسب‌گذاری شود و درخت به صورت پیش‌ترتیب پیمایش گردد، دنباله حاصل، یک دنباله بیتی به طول $kn+1$ می باشد که با حذف صفر آخر، دنباله x متناظر درخت T نامیده می‌شود. دنباله z، دنباله‌ایی صحیح به طول n است، که در آن z_i جایگاه i اُمین یک (گره داخلی) در دنباله x است.

مثال ۱.۲. برای درختی از مجموعه $T_{5,4}$ که در شکل زیر نشان داده شده است، دنباله‌های x و z به صورت زیر است:

$$x = 1101000000001001000000$$

$$z = 1, 2, 4, 12, 15$$



قضیه ۱.۲ یک دنباله دودویی مانند $x = (x_1, \ldots x_{kn})$ می‌تواند x-Sequence یک درخت k-تایی منظم با n گره داخلی باشد اگر و تنها اگر:

۱) n عنصر یک و $k(n-1)$ عنصر برابر صفر داشته باشد.
۲) برای هر زیردنباله آغازین به طول l تعداد عناصر یک، حداقل $\lceil l/k \rceil$ باشد.

برهان.

نشان داده می‌شود که برای دنباله x متناظر به یک درخت k-تایی منظم با n گره داخلی، مانند T، شرط‌های فوق صدق می کند. چنانچه گفته شد، درخت T دقیقا n گره داخلی و $k(n-1)+1$ برگ دارد که با در نظر نگرفتن برگ آخر، در شرط (۱) صدق می‌کند.

همچنین، هر زیر دنباله آغازین از x متناظر به یک درخت k-تایی غیر منظم است. از این رو تعداد برگ‌ها (صفرها) حداکثر $k - ۱$ برابر تعداد گره‌های داخلی است. به عبارت دیگر تعداد یک‌ها $\lceil ۱/k \rceil$ طول دنباله است و شرط (۲) برقرار است.

حال باید نشان داد که دنباله‌هایی که شرایط فوق را دارند متناظر به درخت‌های مجموعه $T_{n,k}$ می‌باشند: دنباله‌هایی با شرایط (۱) و (۲) جواب مساله ترکیبیاتی box-office[۲] است؛ از آن رو که تعداد این دنباله‌ها با تعداد درختان $T_{n,k}$ برابر است و برای هر درخت یک دنباله منحصر به فرد وجود دارد، شرط‌های فوق کافی‌است. □

قضیه ۲.۲ یک دنباله صحیح مانند $x = (z_۱, \dots z_n)$ می‌تواند z-Sequence یک درخت k-تایی منظم با n گره داخلی باشد اگر و تنها اگر:

۱) $۰ < z_۱ < z_۲ < \cdots < z_n$.

۲) برای هر $i = ۱, ۲, \dots n$ وجود داشته باشد $z_i < ki - (k - ۱)$.

برهان.

به سادگی براساس تعریف z-Sequence می‌توان نشان داد به ازای هر دنباله صحیحی با شرط‌های فوق یک دنباله x-Sequence متناظر وجود دارد. از این رو بر اساس قضیه ۱.۲ شرط‌های فوق هم‌ارزی مجموعه این دنباله‌ها و درخت‌های $T_{n,k}$ لازم و کافی است.

□

[۲] در این مساله kn مشتری در یک صف بلیط فروشی هستند که n نفر از آن‌ها هر کدام $k - ۱$ اسکناس یک تومانی و بقیه تنها اسکناس های k تومانی دارند. اگر قیمت هر بلیط $k - ۱$ تومان باشد و هر مشتری یک بلیط خریداری نماید، مشتری ها به چند طریق می‌توانند در یک صف قرار گیرند که کسی برای پول خرد منتظر نماند[۲۱].

در جدول ۲ـ۱ لیست ۱۲ دنباله x و z متناظر با درختان $T_{۳,۳}$ به ترتیب B-Order نشان داده شده است.

قضیه ۳.۲ برای دو درخت k-تایی منظم T و T' از مجموعه $T_{n,k}$ رابطه $T \prec_{B} T'$ صحیح است اگر و تنها اگر دنباله x متناظر درخت T (x_T) به صورت لغت‌نامه‌ایی از $x_{T'}$ کوچک‌تر باشد؛ یا دنباله z متناظر درخت T (z_T) به صورت لغت‌نامه‌ایی از $z_{T'}$ بزرگ‌تر باشد.

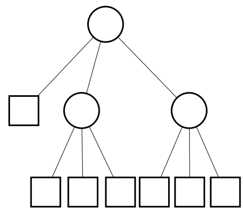
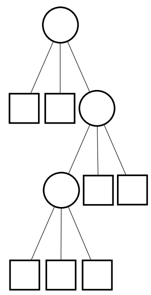
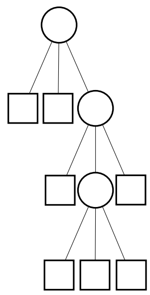
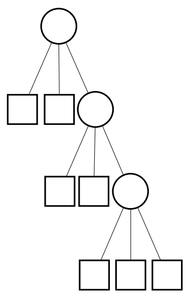
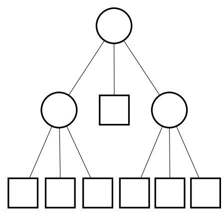
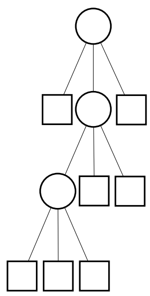
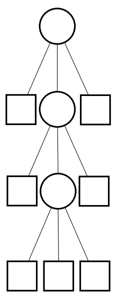
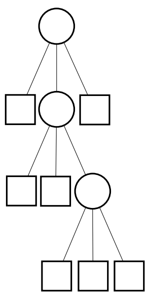
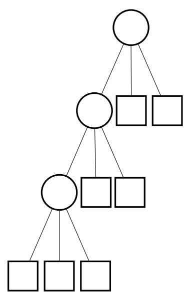
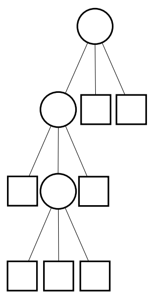
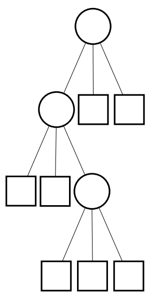
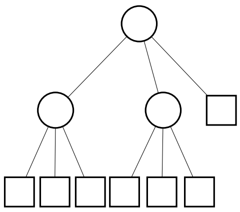
برهان.

بر اساس تعریف x و z بدیهی است که اگر $x_T < x_{T'}$ در آن صورت $z_T > z_{T'}$. چرا که اگر x_T از $x_{T'}$ کوچک‌تر باشد در آن‌صورت اندیسی مانند i وجود دارد که برای مقادیر $j = ۱, ۲, \ldots, i - ۱$ این دو دنباله برابرند و در اندیس i مقدار x_T از $x_{T'}$ کوچک‌تر است؛ به عبارت دیگر، در آن موقعیت دنباله x_T صفر و دنباله $x_{T'}$ یک می‌باشد. حال، از آن رو که دنباله z که جایگاه یک‌ها در دنباله x است، دنباله z متناظر به x_T از دنباله z متناظر به $x_{T'}$ بزرگ‌تر خواهد بود و بالعکس.

بنابراین، کافی است که نشان داده شود اگر $x_T < x_{T'}$ باشد، درخت T پیش از درخت T' در ترتیب B-Order است و بالعکس. بر اساس تعریف $T \prec_{B} T'$ است اگر و تنها اگر در پیمایش پیش‌ترتیب l اُمین گره مشاهده شده در درخت T یک برگ باشد و در درخت T' یک گره داخلی؛ همچنین گره‌های اول، دوم، ... و $l - ۱$ اُم در این دو درخت مشابه باشند. این، در وضعی اتفاق می‌افتد که برای همه $i = ۱, ۲, \ldots, l - ۱$ وجود داشته باشد $x_T(i) = x_{T'}(i)$ و $x_T(l) = ۰ < x_{T'}(l) = ۱$؛ یا به عبارت دیگر $x_T < x_{T'}$.

□

جدول ۲ـ۱: درختان $T_{۳,۳}$ به همراه دنباله‌های x و z متناظر

			
$x =$ (۱۰۱۰۰۰۱۰۰)	$x =$ (۱۰۰۱۱۰۰۰۰)	$x =$ (۱۰۰۱۰۱۰۰۰)	$x =$ (۱۰۰۱۰۰۱۰۰)
$z =$ (۱۳۷)	$z =$ (۱۴۵)	$z =$ (۱۴۶)	$z =$ (۱۴۷)
			
$x =$ (۱۱۰۰۰۰۱۰۰)	$x =$ (۱۰۱۱۰۰۰۰۰)	$x =$ (۱۰۱۰۱۰۰۰۰)	$x =$ (۱۰۱۰۰۱۰۰۰)
$z =$ (۱۲۷)	$z =$ (۱۳۴)	$z =$ (۱۳۵)	$z =$ (۱۳۶)
			
$x =$ (۱۱۱۰۰۰۰۰۰)	$x =$ (۱۱۰۱۰۰۰۰۰)	$x =$ (۱۱۰۰۱۰۰۰۰)	$x =$ (۱۱۰۰۰۱۰۰۰)
$z =$ (۱۲۳)	$z =$ (۱۲۴)	$z =$ (۱۲۵)	$z =$ (۱۲۶)

۳.۲ الگوریتم تولید درخت k-تایی براساس دنباله z

پیشتر گفته شده است که ترتیب عکس لغت‌نامه‌ایی بر روی z-Sequence با ترتیب B-Order بر روی درخت‌های متناظر، منطبق است. از این رو، یک روش مناسب برای تولید تمام درخت‌های k-تایی با n گره داخلی، براساس دنباله z-Sequence متناظر آن‌ها این است که ابتدا اولین دنباله z-Sequence در ترتیب لغت‌نامه‌ایی تولید شود؛ سپس به کمک الگوریتمی که با دریافت یک دنباله ورودی، دقیقا دنباله بعدی را در ترتیب لغت‌نامه‌ایی تولید می‌کند، درخت‌های مجموعه $T_{n,k}$ یکی پس از دیگری در ترتیب عکس B-Order تولید گردند. این روش که در الگوریتم ۲—۱ نشان داده شده است، درخت‌های k-تایی بعدی در ترتیب عکس B-Order را تولید می‌کند.

دنباله z-Sequence آغازین در ترتیب لغت‌نامه‌ایی به صورت زیر می‌باشد:

$$(z_۱, \ldots, z_n) = (۱, ۲, \ldots, n),$$

و دنباله پایانی در این ترتیب اینگونه است:

$$(z_۱, \ldots, z_n) = (۱, k + ۱, ۲k + ۱, \ldots, (n - ۱)k + ۱).$$

الگوریتم ۲—۱ برای تولید دنباله بعدی، انتهایی ترین عنصر دنباله، که مقدار آن کوچک‌تر از $(i - ۱)k + ۱$ باشد، پیدا نموده و یک واحد افزایش می‌دهد. سپس، ادامه دنباله را با کوچک‌ترین مقادیر ممکن z_i، که $z_{i-۱} + ۱$ می‌باشد، مقداردهی می‌کند.

```
procedure BOrderNextzSeq (var z: zSeq; n, k: integer);
begin
   for i := n downto 1 do
      if z_i < (i − 1)k + 1 then
         z_i := z_i + 1;
         for j := i + 1 to n do  z_j := z_{j−1} + 1;
         return;
      end;
   end;
end;
```

الگوریتم ۲—۱: الگوریتم زکس برای تولید درختان k-تایی در B-Order

پیچیدگی الگوریتم ۲—۱ وابسته به تعداد مقایسه‌های لازم جهت یافتن موقعیت انتهایی‌ترین عنصر کوچک‌تر از $(i-۱)k+۱$ در اولین حلقه for آن می‌باشد.

تعریف ۱.۲ $R^z_{n,k,c}$ مجموعه تمام دنباله‌های z-Sequence متناظر به درخت‌های $T_{n,k}$ می‌باشد که در الگوریتم ۲—۱ به c مقایسه نیاز دارند. تعداد اعضای این مجموعه با $r^z_{n,k,c}$ نشان داده می‌شود. $\triangledown$

محاسبه $r^z_{n,k,c}$ روشی برای یافتن نرخ رشد الگوریتم ۲—۱ است. بدین صورت که میانگین تعداد مقایسه های لازم برای تولید هر دنباله از رابطه زیر به دست می آید:

$$\frac{۱}{t_{n,k}}\sum_{c=۱}^{n} c\ r^z_{n,k,c}. \qquad (۱)$$

لم ۱.۲ برای محاسبه $r^z_{n,k,c}$ روابط زیر به صورت بازگشتی وجود دارد:

$$r^z_{n,k,c} = t_{n-c+1,k} - t_{n-c,k}\ , \qquad 1 \leqslant c < n, \tag{2}$$

$$r^z_{n,k,c} = 1\ , \qquad c = n. \tag{3}$$

برهان.

برای نشان دادن صحت رابطه (۲) توجه به این نکته کافی است که $t_{n-c+1,k}$ شامل همه دنباله‌هایی است که حداکثر به c مقایسه نیاز دارند و $t_{n-c,k}$ شامل دنباله‌هایی است که حداکثر به $c - 1$ مقایسه نیاز دارند. از این رو تفاوت این دو تعداد دنباله‌هایی است که دقیقا به c مقایسه نیاز دارند. رابطه (۳) بسیار بدیهی است: تنها در حالتی n مقایسه انجام می‌شود که دنباله ورودی دنباله پایانی در ترتیب لغت‌نامه‌ایی باشد. □

می‌توان نشان داد تعداد مقایسه‌ها در الگوریتم ۲—۱ دارای حد زیر است[۲۱]

$$\lim_{n\to\infty} \frac{r^z_{n,k,c}}{t_{n,k}} = (1 - \bar{k})\bar{k}^{c-1}. \tag{4}$$

که در آن $\bar{k} = (k-1)^{k-1}/k^k$.

و به سادگی می‌توان برای $\bar{k}$ حد بالایی محاسبه نمود:

$$\bar{k} = \frac{(k-1)^{k-1}}{k^k} = \frac{(1-\frac{1}{k})^k}{k-1} < \frac{1}{(k-1)e}. \tag{5}$$

نتیجه حاصل از حد رابطه (۴) بسیار قابل توجه است. برای مثال، به سادگی می‌توان مشاهده نمود که مجموعه $R^z_{n,k,1}$، یعنی دنباله‌هایی که تنها به یک مقایسه نیاز دارند، بخش

بزرگی از مجموعه $T_{n,k}$ می‌باشند؛ زیرا:

$$\lim_{n\to\infty}\frac{r^z_{n,k,۱}}{t_{n,k}} = ۱ - \bar{k} = ۱ - \frac{(k-۱)^{k-۱}}{k^k} < ۱ - \frac{۱}{(k-۱)e}.$$

برای حالت خاص $k=۲$ (درختان دودیی) این مقدار تقریبا برابر ۷۵ درصد کل دنباله‌ها می‌باشد و برای $k=۳$ تقریبا ۸۵ درصد دنباله‌های مجموعه $T_{n,۳}$ را شامل می‌شود.

به علاوه، با ترکیب رابطه‌های (۱)، (۴) و (۵) می‌توان نوشت:

$$\lim_{n\to\infty}\frac{\sum_{c\geqslant ۱} c r^z_{n,k,c}}{t_{n,k}} = \frac{۱}{۱-\bar{k}} < \frac{۱}{۱-۱/(k-۱)e}$$

این بدین معناست که میانگین تعداد مقایسه های لازم برای تولید هر دنباله برای n های بزرگ برابر $\frac{۱}{۱-\bar{k}}$ است که کوچک‌تر از $\frac{(k-۱)e}{(k-۱)e-۱}$ می‌باشد.

۴.۲ رتبه‌گذاری و رتبه‌گشایی درخت k-تایی

در این بخش به الگوریتم هایی می‌پردازد که رتبه یک درخت k-تایی منظم را از روی دنباله z-Sequence متناظر درخت تولید می‌کند و بر اساس رتبه داده شده z-Sequence متناظر را تولید می‌کند. برای بدست آوردن رتبه درخت از روی دنباله z متناظر با آن، از ارزش $a_{i,j,k}$ که به صورت زیر تعریف شده‌است، استفاده می‌شود.

تعریف ۲.۲ $a_{i,j,k}$ تعداد دنباله‌های z به طول $\lfloor\frac{i+k-۱}{k-۱}\rfloor$ است که حداقل $j - \lfloor\frac{i+k-۱}{k-۱}\rfloor$ عنصر آغازین آن، به ترتیب از یک تا $j - \lfloor\frac{i+k-۱}{k-۱}\rfloor$باشند. $\triangledown$

لم ۲.۲ برای محاسبه ارزش $a_{i,j,k}$ از روابط بازگشتی زیر می‌توان استفاده نمود:

$$a_{i,j,k} = ۱ , \qquad i = ۰, \qquad (٦)$$

$$a_{i,j,k} = ۰ , \qquad i > \frac{j-۱}{k-۱} + ۱, \qquad (۷)$$

$$a_{i,j,k} = a_{i,j-۱,k} + a_{i-۱,j,k} , \qquad \text{در غیر این صورت} . \qquad (۸)$$

برهان.

اثبات در [۲۲]. $\square$

لم ۳.۲ اگر $i, j \geqslant ۰$ و $k > ۱$، آنگاه:

$$a_{i,j,k} = \binom{i+j-۱}{i} - \sum_{t=۱}^{\lfloor\frac{i-۱}{k-۱}\rfloor} \binom{i+j-۱-kt}{j-t} \frac{۱}{(k-۱)t+۱} \binom{kt}{t}. \qquad (۹)$$

که در این رابطه $\lfloor x \rfloor$ بزرگ‌ترین عدد صحیح کوچکتر یا مساوی x است و همچنین $\sum_{t=۱}^{s}$ برای $s < ۱$ صفر در نظر گرفته می‌شود.

برهان.

برای اثبات نشان داده می‌شود که سمت چپ رابطه (۹) در روابط (٦)، (۷) و (۸) صدق می کند. روابط (٦) و (۷) به سادگی قابل محاسبه می‌باشند. همچنین برای (۸)

با کمک رابطه خیام پاسکال

$$\binom{m-۱}{r}+\binom{m-۱}{r-۱}=\binom{m}{r},$$

به سادگی می‌توان نوشت:

$$\begin{aligned}
a_{i,j-۱,k} &+ a_{i-۱,j,k} \\
&= \binom{i+j-۲}{i}-\sum_{t=۱}^{\lfloor\frac{i-۱}{k-۱}\rfloor}\binom{i+j-۲-kt}{j-t-۱}\frac{۱}{(k-۱)t+۱}\binom{kt}{t}+ \\
&\quad \binom{i+j-۲}{i-۱}-\sum_{t=۱}^{\lfloor\frac{i-۲}{k-۱}\rfloor}\binom{i+j-۲-kt}{j-t}\frac{۱}{(k-۱)t+۱}\binom{kt}{t} \\
&= \binom{i+j-۱}{i}-\sum_{t=۱}^{\lfloor\frac{i-۱}{k-۱}\rfloor}\binom{i+j-۱-kt}{j-t}\frac{۱}{(k-۱)t+۱}\binom{kt}{t} \\
&= a_{i,j,k}.
\end{aligned}$$

□

به عنوان نمونه، مقادیر $a_{i,j,k}$ برای $i=۰\cdots۷$، $j=۰\cdots۷$ و $k=۳$ در جدول ۲_۲ آمده است.

با استفاده از تعاریف بالا به سادگی می‌توان الگوریتم‌های رتبه‌گشایی و رتبه‌گذاری را ارایه نمود.

الگوریتم رتبه‌گذاری نشان داده شده در الگوریتم ۲—۲ به ازای هر برگ، مقدار $a_{l,v,k}$ را محاسبه و جمع می‌کند؛ که در آن، l تعداد برگ‌های مشاهده نشده بعد از برگ مورد نظر (بدون در نظر گرفتن $k-۱$ برگ انتهایی) در پیمایش پیش‌ترتیب درخت است و همچنین v تعداد گره‌های داخلی پیمایش نشده بعد از برگ مورد نظر می باشد.

جدول ۲ـ۲: مقادیر $a_{0\ldots7,0\ldots7,3}$ برای محاسبه رتبه از روی دنباله z

j / i	۰	۱	۲	۳	۴	۵	۶	۷
۰	۱	۱	۱	۱	۱	۱	۱	۱
۱	۰	۱	۲	۳	۴	۵	۶	۷
۲	۰	۰	۰	۳	۷	۱۲	۱۸	۲۵
۳	۰	۰	۰	۰	۰	۱۲	۳۰	۵۵
۴	۰	۰	۰	۰	۰	۰	۰	۵۵
۵	۰	۰	۰	۰	۰	۰	۰	۰
۶	۰	۰	۰	۰	۰	۰	۰	۰
۷	۰	۰	۰	۰	۰	۰	۰	۰

برای محاسبه پیچیدگی زمانی الگوریتم رتبه‌گذاری، کافی است دقت شود که در محاسبه مجموع $a_{l,v,k}$ برای هر برگ مقدار l و برای هر گره داخلی v کاهش می‌یابد. از این رو، رتبه یک دنباله داده شده در $O(kn)$ محاسبه می‌شود.

برای رتبه‌گشایی و تولید یک درخت براساس رتبه داده شده، r، عکس فرایند فوق در الگوریتم ۲—۳ انجام می‌شود. برای محاسبه هر z_i کافی است، تعداد برگ‌هایی را که پیش از آن گره داخلی در پیمایش پیش‌ترتیب مشاهده شده‌اند را شمرد. از این رو، مقادیر $a_{l,v,k}$ را با ثابت نگه داشتن v (تعداد گره‌های داخلی مشاهده نشده) و کاهش l (تعداد برگ‌های مشاهده نشده) از مقدار r کاسته می‌شوند. اگر دیگر برگی وجود نداشته باشد، $r > a_{l,v,k}$، در این صورت، z_i مقدارگذاری می‌شود و v کاهش می‌یابد.

```
function BOrderRank (Z: ZSeq; n, k : integer): integer;
begin
    l := (k − 1)n − k + 1;
    p := 1;    v := n − 2;    z_{n+1} := kn;
    for i := 2 to n + 1 do
        for j := p to z_i − 1 do
            r := r + a_{l,v,k}
            l := l − 1;
        end;
        v := v − 1;
        p := z_i;
    end;
    return r;
end;
```

الگوریتم ۲—۲: الگوریتم زکس برای رتبه‌گذاری درخت‌های k-تایی در B-Order

پیچیدگی زمانی الگوریتم رتبه‌گشایی، همانند الگوریتم رتبه‌گذاری، از در $O(kn)$ است. چرا که در کاستن مقادیر $a_{l,v,k}$ از رتبه داده شده، برای هر برگ مقدار l و برای هر گره داخلی v یک واحد کاهش می‌یابد. از این رو، رتبه یک دنباله داده شده محاسبه می‌شود.

```
function BOrderUnrank (r, n, k : integer): ZSeq;
begin
    l := (k − 1)n − k + 1;
    p := 1;    v := n − 2;
    for i := 1 to n do
        while r > a_{l,v,k} do
            p := p + 1;
            r := r − a_{l,v,k}
            l := l − 1;
        end;
        v := v − 1;
        p := p + 1;
        z_i := p;
    end;
    return (z_1, z_2, ···, z_n);
end;
```

الگوریتم ۲—۳: الگوریتم زکس برای رتبه‌گشایی درخت‌های k-تایی در B-Order

۳

تولید درخت‌هایی با n گره و m برگ در B-Order

در این فصل الگوریتم‌های ارایه شده توسط پالو[1] برای تولید درخت‌هایی با n گره داخلی و m گره خارجی ارایه خواهد شد[۱۴]. پالو برای تولید این درختان، از روش نشاندن یک درخت در یک درخت m-تایی منظم استفاده نموده است؛ لذا برای درک بیشتر، در بخش اول به کد گذاری درخت‌های k-تایی توسط P-Sequence پرداخته می‌شود[۱۴] و سپس در بخش ۲.۳ تولید درختان k-تایی منظم در ترتیب B-order ارایه می‌گردد[۱۴، ۱۷، ۱۸]. نشان داده می‌شود که الگوریتم تولید پالو دارای نرخ رشدی از $O(k)$ و مستقل از n می‌باشد.

[1] Jean-Marcel Pallo

در بخش سوم رتبه‌گذاری و رتبه‌گشایی درخت‌های مذکور به تفصیل ارایه می‌شود [۱۴، ۱۸].
بخش ۴.۳ به روش نشاندن درخت‌هایی با n گره و m برگ در یک درخت m-تایی می‌پردازد و بخش نهایی، الگوریتم‌های تولید این درختان را بررسی می‌کند. پالو الگوریتمی برای رتبه‌گذاری یا رتبه گشایی این درختان ارایه نکرده است.

۱.۳ کدگذاری درخت های k-تایی توسط P-Sequence

در این بخش دنباله P-Sequence برای کدگذاری یک درخت k-تایی منظم ارایه می‌شود. روابط، تعاریف و عملگرهای لازم برای دو بخش بعد ارایه می‌شود.

در این فصل نیز فرض خواهیم کرد که $T_{n,k}$ مجموعه تمام درخت‌های k-تایی منظم با n گره داخلی باشد. بنابراین

$$t_{n,k} = |T_{n,k}| = \frac{۱}{kn+۱}\binom{kn+۱}{n}$$

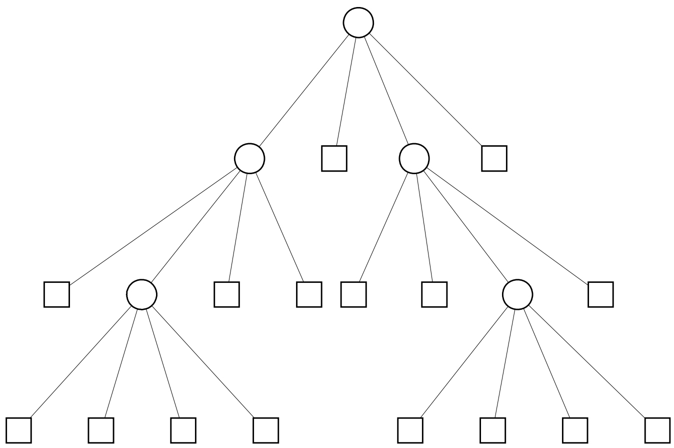
بدیهی است که اگر T_k عضوی از مجموعه $T_{n,k}$ باشد، تعداد برگ‌ها (گره‌های خارجی) T_k برابر $n(k-۱)+۱$ خواهد بود.

برای یک درخت k-تایی منظم مانند T_k، دنباله P-Sequence متناظر به صورت یک دنباله $(p_۱, \ldots, p_{n(k-۱)})$ می‌باشد که p_i برای هر برگ تعداد گره‌های داخلی مشاهده شده پیش از آن در پیمایش پیش‌ترتیب می‌باشد و با P_T نشان داده می‌شود. برای سادگی اگر گره‌های خارجی با ۱ و گره‌های داخلی با ۰ برچسب‌گذاری شود و درخت به صورت پیش‌ترتیب

پیمایش شود، در آن صورت، دنباله P-Sequence به راحتی با شمارش صفرهای ماقبل هر ۱ به دست می‌آید.

مثال ۱.۳. اگر $n = ۵$، $k = ۴$ و $T_k = ۰۰۱۰۱۱۱۱۱۱۱۰۱۱۰۱۱۱۱۱۱$ در آن صورت

$$P_T = (۲, ۳, ۳, ۳, ۳, ۳, ۳, ۳, ۴, ۴, ۵, ۵, ۵, ۵, ۵)$$



قضیه ۱.۳ یک دنباله عددی مانند $(p_۱, \ldots, p_{n(k-۱)})$ می‌تواند P-Sequence یک درخت k-تایی منظم با n گره داخلی باشد اگر و تنها اگر:

۱) برای هر $i \in [۱, n(k-۱) - ۱]$ وجود داشته باشد: $P_i \leqslant P_{i+۱}$

۲) $P_{n(k-۱)} = n$

۳) برای هر $i \in [۱, n(k-۱) - ۱]$ وجود داشته باشد: $P_i \geqslant ۱ + \lfloor \frac{i-۱}{k-۱} \rfloor$

در جدول ۳ـ۱، لیست ۱۲ دنباله P-Sequence متناظر با درختان $T_{3,3}$ به ترتیب B-Order نشان داده شده است.

قضیه ۲.۳ برای دو درخت k-تایی منظم T و T' از مجموعه $T_{n,k}$ رابطه $T \prec_{B} T'$ صحیح است اگر و تنها اگر دنباله P-Sequence متناظر درخت T (P_T) به صورت لغت‌نامه‌ایی از $P_{T'}$ کوچک‌تر باشد.

تعریف ۱.۳ برای یک P-Sequence به طول $n(k-1)$ عملگری به نام کاهش از چپ می‌توان تعریف نمود: با فرض اینکه اگر t کوچک‌ترین اندیسی باشد به طوری که:

$$p_{t-1} < \underbrace{p_t = p_{t+1} = \cdots = p_{t+k-1}}_{k} = q \quad 0 < q \leqslant n,$$

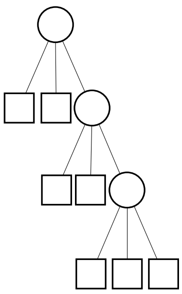
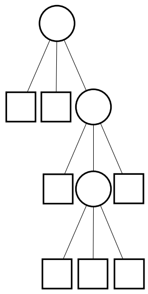
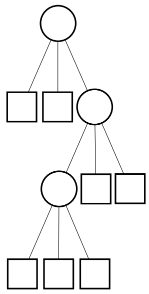
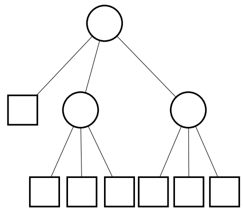
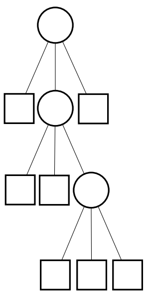
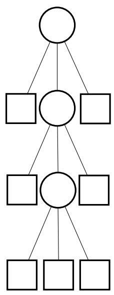
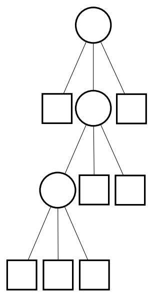
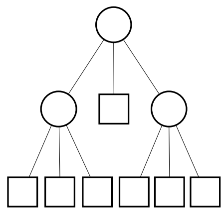
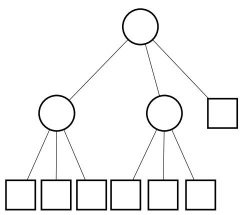
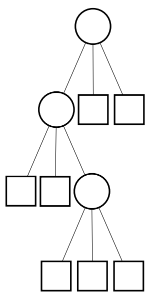
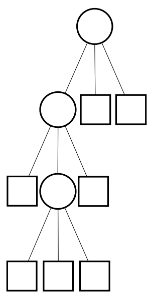
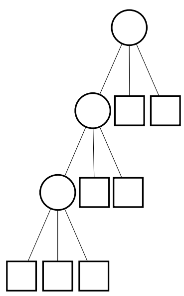
در آن صورت، عملگر کاهش از چپ به فرایند تعویض $p_t, p_{t+1}, \ldots, p_{t+k-1}$ با $q-1$ و کاهش یک واحد از ادامه دنباله و تولید دنباله زیر اطلاق می‌شود:

$$(p_1, p_2, \ldots, p_{t-1}, q-1, p_{t+k}-1, \ldots, p_{n(k-1)}-1).$$

اعمال عملگر کاهش از چپ بر روی یک درخت k-تایی منظم، چپ‌ترین گره داخلی درخت که همه فرزندان آن برگ می‌باشد را با یک برگ تعویض می‌کند.

عملگر کاهش از راست همانند عملگر کاهش از چپ تعریف می‌شود با این تفاوت که t کوچکترین اندیسی است که $P_t = n$ باشد. این عملگر راست‌ترین گره داخلی درخت که همه فرزندان آن برگ می‌باشد را با یک برگ جایگزین می‌کند. $\triangledown$

جدول ۳ـ۱: درختان $T_{۳,۳}$ به همراه دنباله‌های P-Sequence متناظر

 $P =$ (۱۱۲۲۳۳)	 $P =$ (۱۱۲۳۳۳)	 $P =$ (۱۱۳۳۳۳)	 $P =$ (۱۲۲۲۳۳)
 $P =$ (۱۲۲۳۳۳)	 $P =$ (۱۲۳۳۳۳)	 $P =$ (۱۳۳۳۳۳)	 $P =$ (۲۲۲۲۳۳)
 $P =$ (۲۲۲۳۳۳)	 $P =$ (۲۲۳۳۳۳)	 $P =$ (۲۳۳۳۳۳)	 $P =$ (۳۳۳۳۳۳)

۲.۳ الگوریتم تولید درخت k-تایی و تحلیل آن

پیشتر گفته شده است که ترتیب لغت‌نامه‌ایی بر روی P-Sequence با ترتیب B-Order بر روی درختان متناظر، منطبق است. از این رو یک روش برای تولید تمام درخت‌های k-تایی با n گره داخلی، براساس دنباله P-Sequence متناظر آن‌ها این است که ابتدا اولین دنباله P-Sequence در ترتیب لغت‌نامه‌ایی تولید می‌شود؛ سپس به کمک الگوریتمی که با دریافت یک دنباله ورودی، دقیقا دنباله بعدی را در ترتیب لغت‌نامه‌ایی تولید می‌کند، درخت‌های مجموعه $T_{n,k}$ یکی پس از دیگری تولید می‌گردند. این روش که در الگوریتم ۳—۱ نشان داده شده است، درخت‌های k-تایی بعدی در ترتیب B-Order را تولید می‌کند.

دنباله P-Sequence آغازین در ترتیب لغت‌نامه‌ایی به صورت زیر می‌باشد:

$$(p_۱, \ldots, p_{n(k-۱)}) = (\underbrace{۱, \ldots, ۱}_{k-۱}\ ,\ \underbrace{۲, \ldots, ۲}_{k-۱}\ ,\ \ldots,\ \underbrace{n, \ldots, n}_{k-۱}\ ),$$

و دنباله پایانی در این ترتیب اینگونه است:

$$(p_۱, \ldots, p_{n(k-۱)}) = (\underbrace{n, \ldots, n}_{n(k-۱)}).$$

الگوریتم ۳—۱ برای تولید دنباله بعدی، انتهایی ترین عنصر دنباله، که از n کوچکتر باشد، پیدا نموده و یک واحد افزایش می‌دهد. سپس، ادامه دنباله با کوچکترین مقادیر ممکن p_i، براساس قضیه ۱.۳، مقداردهی می‌شود.

پیچیدگی الگوریتم ۳—۱ وابسته به تعداد مقایسه‌های لازم جهت یافتن موقعیت انتهایی‌ترین عنصر کوچک‌تر از n در اولین حلقه for آن می‌باشد. از آنجا که $k - ۱$ عنصر انتهایی هر دنباله بر اساس تعریف برابر با n می‌باشد، تنها $(n - ۱)(k - ۱)$ عنصر ابتدایی دنباله از $n(k - ۱)$ عنصر آن جستجو می‌شود.

```
procedure BOrderNextPSeq (var P: PSeq; n, k: integer);
begin
   for i := (n − 1)(k − 1) downto 1 do
      if p_i < n then
         p_i := p_i + 1;
         for j := i + 1 to n(k − 1) do  p_j := max(p_i, 1 + [(j−1)/(k−1)]);
         return;
      end;
   end;
end;
```

الگوریتم ۳—۱: الگوریتم پالو برای تولید درختان k-تایی در B-Order

تعریف ۲.۳ $R_{n,k,c}$ مجموعه تمام دنباله‌های P-Sequence متناظر به درخت‌های $T_{n,k}$ می‌باشد که در الگوریتم ۳—۱ به c مقایسه نیاز دارند. تعداد اعضای این مجموعه با $r_{n,k,c}$ نشان داده می‌شود. $\triangledown$

پیشتر گفته شد که $k-۱$ عنصر انتهایی هر دنباله بر اساس تعریف برابر با n می باشند و جستجو نمی‌شوند. حال، اگر $c-۱$ عنصر از $(n-۱)(k-۱)$ عنصر مورد جستجو در یک دنباله، برابر با n باشد، آن دنباله تنها نیاز به c مقایسه خواهد داشت. بنابراین، مجموعه $R^p_{n,k,c}$ شامل تمام دنباله‌های P-Sequence است که تنها $(c-۱)+(k-۱)$ عنصر انتهایی آن دنباله‌ها برابر با n است.

اگر عملگر کاهش از راست را بر روی هر یک از دنباله‌های متعلق به $R^p_{n,k,c}$ اعمال شود، دنباله های عضو $R^p_{n,k,c}$، دقیقا از عنصر $t=n(k-۱)-(k+c-۲)$ اُم، کاهش می‌یابد. به عبارت دیگر، در صورتی که عملگر کاهش از راست بر روی دنباله‌های عضو $R^p_{n,k,c}$ اعمال

گردد، در درخت‌های متناظر آن‌ها، گره داخلی t اُم که k برگ دارد، با یک برگ جایگزین می‌شود.

محاسبه $r^p_{n,k,c}$ روشی برای یافتن نرخ رشد الگوریتم ۳—۱ است. بدین صورت که میانگین تعداد مقایسه های لازم برای تولید هر دنباله از رابطه زیر به دست می آید:

$$\frac{1}{t_{n,k}} \sum_{c=1}^{(n-1)(k-1)} c\, r^p_{n,k,c}. \qquad (1)$$

لم ۱.۳ برای محاسبه $r^p_{n,k,c}$ روابط زیر به صورت بازگشتی وجود دارد:

$$r^p_{n,k,c} = t_{n-1,k}\,, \quad 1 \leqslant c < k, \qquad (2)$$

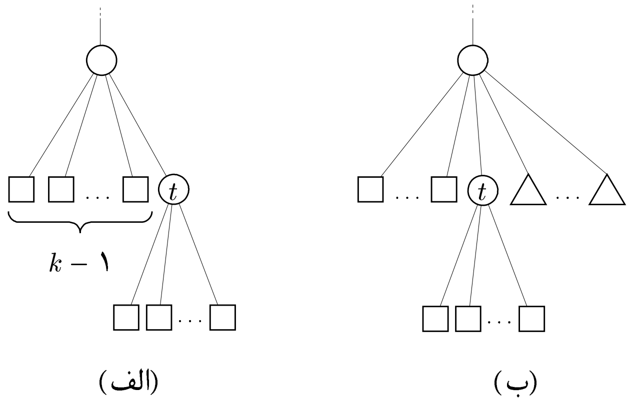
$$r^p_{n,k,c} = r^p_{n-1,k,c-k+1} + r^p_{n,k,c+1}\,, \quad k \leqslant c < (n-1)(k-1), \qquad (3)$$

$$r^p_{n,k,c} = 1\,, \quad c = (n-1)(k-1). \qquad (4)$$

برهان.

برای رابطه (۲)، به سادگی می‌توان نشان داد که رابطه‌ایی یک‌به‌یک بین اعضای $R^p_{n,k,c}$ و $T_{n-1,k}$ هنگامی که $1 \leqslant c < k$ وجود دارد: اگر بر روی یک درخت از اعضای مجموعه $R^p_{n,k,c}$ عملگر کاهش از راست اعمال شود، درختی از مجموعه $T_{n-1,k}$ به دست می آید. همچنین، بالعکس با تعویض $t = (n-1)(k-1) - c + 2$ اُمین برگ درختی از $T_{n-1,k}$ با گره‌ایی داخلی (که خود k برگ دارد)، درختی از $R^p_{n,k,c}$ به دست می آید.

اگر $c \geqslant k$ باشد، نمی‌توان مطمئن بود که درخت حاصل از اعمال عملگر کاهش عضوی از مجموعه $T_{n-1,k}$ باشد. برای اثبات رابطه (۳)، برای یک درخت از $R^p_{n,k,c}$ دو وضع



شکل ۳—۱: دو وضع ممکن برای حالت دوم محاسبه $r^p_{n,k,c}$

می‌تواند وجود داشته باشد: t اُمین برگ لیست شده در دنباله، (الف) فرزند k اُم پدر خود باشد یا (ب) فرزند k اُم پدر خود نباشد. این دو وضع در شکل ۳—۱ نشان داده شده‌اند. در وضع (الف)، اگر زیردنباله $p_t, p_{t+۱}, \ldots, p_{t+k-۱}$ را کاهش از راست داده شود، درختی از $R^p_{n-۱,k,c-k+۱}$ به دست می‌آید؛ چرا که درخت حاصل یک گره داخلی کمتر داشته و قابل کاهش از $t-k+۱$ اُمین برگ خود است. دروضع (ب) که کمی پیچیده‌تر است، زیردنباله $p_t, p_{t+۱}, \ldots, p_{t+k-۱}$ را کاهش از راست داده می‌شود و برگ p_{t+k} با یک گره داخلی (که خود k برگ دارد)، جایگزین می‌شود. درختی حاصل عضوی از $R^p_{n,k,c+۱}$ به دست می‌آید؛ چرا که درخت حاصل قابل کاهش از $t+۱$ اُمین برگ خود است.

□

قضیه ۳.۳ تعداد کل مقایسه‌ها در الگوریتم ۳—۱ برابر است با:

$$\sum_{c=۱}^{(n-۱)(k-۱)} c\, r^p_{n,k,c} = t_{n+۱,k} - (k-۱)t_{n,k}. \qquad (۵)$$

برهان.

با توجه به لم ۱.۳ می‌توان نوشت:

$$\begin{aligned} r^p_{n,k,c} &= r^p_{n-1,k,c-k+1} + r^p_{n,k,c+1} \\ &= r^p_{n-1,k,c-k+1} + r^p_{n-1,k,c-k} + r^p_{n,k,c+2} \\ &= r^p_{n-1,k,c-k+1} + \cdots + r^p_{n-1,k,(n-1)(k-1)} \\ &= \sum_{i=c-k+1}^{(n-1)(k-1)} r^p_{n-1,k,i}. \end{aligned}$$

یا به عبارت دیگر:

$$\sum_{i=c}^{(n-1)(k-1)} r^p_{n,k,i} = r^p_{n+1,k,k+c-1}.$$

حال با توجه به بسط داریم:

$$\begin{aligned} \sum_{c=1}^{(n-1)(k-1)} c r^p_{n,k,c} &= \sum_{j=0} r^p_{n,k,j} + \sum_{j=1} r^p_{n,k,j} + \cdots + \sum_{j=(n-1)(k-1)} r^p_{n,k,j} \\ &= r^p_{n+1,k,k-1} + r^p_{n+1,k,k+1-1} + \cdots + r^p_{n+1,k,n(k-1)} \\ &= \sum_{j=k-1} r^p_{n,k,j} \\ &= r^p_{n+2,k,2(k-1)}. \end{aligned}$$

در عین حال براساس روابط (۲)، (۳) و (۴):

$$\begin{aligned} r^p_{n+2,k,2(k-1)} &= r^p_{n+2,k,2(k-1)-1} - r^p_{n+1,k,k-2} \\ &= r^p_{n+2,k,2(k-1)-1} - t_{n,k} \\ &= r^p_{n+2,k,2(k-1)-2} - 2t_{n,k} \\ &= r^p_{n+2,k,k-1} - (k-1)t_{n,k} \\ &= t_{n+1,k} - (k-1)t_{n,k}. \end{aligned}$$

سرانجام

$$\sum_{c=1}^{(n-1)(k-1)} c\, r^{p}_{n,k,c} = t_{n+1,k} - (k-1)t_{n,k}.$$

□

بنابراین متوسط تعداد مقایسه‌های لازم برای هر دنباله درخت در الگوریتم ۳—۱ بر اساس رابطه (۱) و قضیه قبل دارای نرخ رشد $O((t_{n+1,k}/t_{n,k}) - k)$ می‌باشد. برای درک بیشتر تخمینی برای $t_{n,k}$ براساس $t_{n-1,k}$ آورده می‌شود:

$$\begin{aligned}
t_{n,k} &= \frac{1}{kn+1}\binom{kn+1}{n} \\
&= \frac{1}{kn+1}\,\frac{kn+1}{kn-n+1}\,\frac{(kn)!}{n!(kn-n)!} \\
&= \frac{1}{(k-1)n+1}\,\frac{(kn)!}{n!(kn-n)!} \\
&= \frac{(k-1)(n-1)+1}{(k-1)n+1}\;\frac{1}{n}\;\frac{(kn)\cdots(kn-(k-1))}{(kn-n)\cdots(kn-n-(k-2))}\;t_{n-1,k} \\
&\leqslant \frac{kn}{n}\,\frac{(kn)^{k-1}}{(kn-k-n+2)^{k-1}}t_{n-1,k} \\
&\leqslant k\left(\frac{(\kappa+1)(\eta+1)}{\kappa\eta}\right)^{\kappa} t_{n-1,k} \qquad \text{که} \quad \kappa = k-1 \quad \text{و} \quad \eta = n-1 \\
&\leqslant k\left(1+\frac{\kappa+\eta+1}{\kappa\eta}\right)^{\kappa} t_{n-1,k} \\
&\leqslant k e^{\frac{\kappa+\eta+1}{\eta}} t_{n-1,k} \\
&\leqslant k e^{3} t_{n-1,k} \qquad \text{اگر} \quad \kappa \geqslant \eta
\end{aligned}$$

بنابراین الگوریتم ۳—۱ بر اساس محاسبات این بخش دارای نرخ رشد متوسطی از $O(k)$ که مستقل از تعداد گره‌های داخلی درخت n است می‌باشد.

۳.۳ رتبه گذاری و رتبه گشایی درخت‌های k-تایی

این بخش به الگوریتم هایی می‌پردازد که رتبه یک درخت k-تایی منظم را از روی دنباله P-Sequence متناظر درخت تولید می‌کند و بر اساس رتبه داده شده P-Sequence متناظر را تولید می‌کند. اگر مجموعه‌ایی از دنباله‌های عددی با طول N وجود داشته باشد که دنباله‌های عددی آن به ترتیب الفبایی تولید شده باشد، یک روش برای به دست آوردن رتبه یک دنباله مانند $a_۱, a_۲, \ldots, a_N$ به شرح زیر است: برای همه مقادیر $k = ۱, \ldots, N$، عددی مانند s_k محاسبه می‌شود که تعداد دنباله‌هایی است که $k - ۱$ عناصر ابتدایی آن $a_۱, a_۲, \ldots, a_{k-۱}$ و عنصر k اُم آن برابر با ۱، ۲، ... و $a_k - ۱$ باشد. در این صورت رتبه دنباله فوق برابر $s_۱ + s_۲ + \cdots + s_N$ می‌باشد.

در محاسبه رتبه یک دنباله P-Sequence از میان دنباله‌های درخت‌های $T_{n,k}$ از روش فوق استفاده می‌شود. برای این منظور تعریف ۳.۳، لم ۲.۳ و قضیه ۴.۳ پیش از ارایه الگوریتم‌های رتبه‌گذاری و رتبه گشایی بحث می‌گردد.

تعریف ۳.۳ $S_{n,v,i,k}$ مجموعه تمام دنباله‌های P-Sequence می‌باشد که i عنصر ابتدایی تمام دنباله‌های آن برابر با v است. تعداد اعضای این مجموعه با $s_{n,v,i,k}$ نشان داده می‌شود. $\triangledown$

مجموعه $S_{n,v,i,k}$ شامل دنباله‌های P-Sequence است که در درخت متناظر آنها اولین i برگ‌هایی که مشاهده می‌شوند در سطح v می‌باشند. محاسبه $s_{n,v,i,k}$ ابزاری برای رتبه‌گذاری و رتبه‌گشایی درخت‌های $T_{n,k}$ است. مقادیر $s_{n,v,i,k}$ در جدول ۳ـ۲ برای درخت‌های $T_{۳,۳}$ نمایش داده شده است. محاسبه $s_{n,v,i,k}$ از طریق اعمال عملگر کاهش بر روی درخت‌های

جدول ۳_۲: مقادیر $s_{۳,v,i,۳}$

$s_{۳,v,i,۳}$	$i=۱$	$i=۲$	$i=۳$	$i=۴$	$i=۵$	$i=٦$
$v=۱$	۷	۳				
$v=۲$	۴	۳	۲	۱		
$v=۳$	۱	۱	۱	۱	۱	۱

متناظر دنباله‌های مجموعه $S_{n,v,i,k}$ امکان‌پذیر می‌باشد.

لم ۲.۳ برای محاسبه $s_{n,v,i,k}$ روابط زیر به صورت بازگشتی وجود دارد:

$$s_{n,\circ,i,k} = \circ \quad (۱ \leqslant i < k), \tag{٦}$$

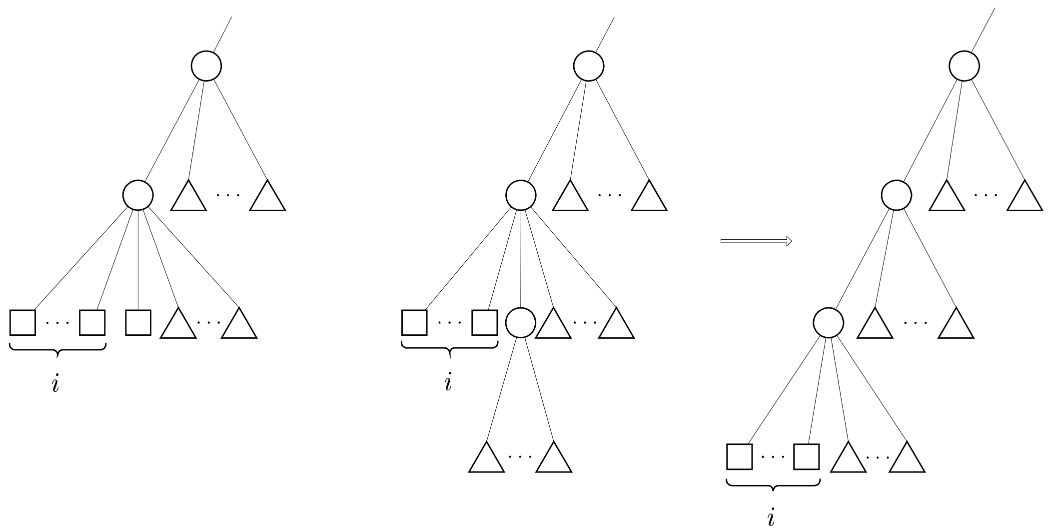
$$s_{n,n,i,k} = ۱ \quad (i \leqslant n(k-۱)), \tag{۷}$$

$$s_{n,v,k,k} = s_{n-۱,v-۱,۱,k} \quad (n > ۱, v > \circ), \tag{۸}$$

$$s_{n,v,i,k} = s_{n,v+۱,i,k} + s_{n,v,i+۱,k} \quad (۱ \leqslant i \leqslant v(k-۱), ۱ \leqslant v \leqslant n). \tag{۹}$$

برهان.

رابطه‌های (٦) و (۷) به صورت شهودی برقرار است. رابطه (۸) درخت‌هایی را در نظر می‌گیرد که k برگ اول آن در سطح v قرار دارد. بدیهی است که این k برگ دارای پدر یکسان بوده و در صورت کاهش از چپ و جایگزینی آن گره پدر با یک برگ، درختی به دست می آید که یک گره داخلی کمتر دارد و اولین برگ مشاهده شده در سطح $v-۱$ است. از این رو دنباله متناظر با درخت حاصل با $v-۱$ شروع شده و در مجموعه $S_{n-۱,v-۱,۱,k}$ قرار دارد.



شکل ۳—۲: دو وضع ممکن برای حالت چهارم محاسبه $s_{n,v,i,k}$

رابطه (۹) کمی پیچیده‌تر است. در این حالت دو وضعیت برای درختان متناظر مجموعه $S_{n-۱,v-۱,۱,k}$ می‌توان در نظر گرفت: گره مجاور i اُمین برگ لیست شده در دنباله، (الف) یک برگ در همان سطح v یا (ب) یک زیردرخت باشد. این دو وضع در شکل ۳—۲ نشان داده شده‌اند. در وضع (الف)، درخت به سادگی می‌تواند از مجموعه $S_{n,v,i+۱,k}$ باشد. دروضع (ب) که کمی پیچیده‌تر است، گره مجاور i اُمین برگ، با برگ اول جایگزین می‌شود. این تغییر و درخت حاصل در شکل ۳—۲(ب) نمایش داده شده است. بدیهی است که درخت در مجموعه $S_{n,v+۱,i,k}$ قرار دارد.

□

مقدار $s_{n,v,i,k}$ تنها تعداد دنباله‌هایی نیست که $p_۱ = \cdots = p_{i-۱} = p_i = v$ باشد. این مقدار، همچنین، دنباله‌هایی را می‌شمارد که در آن‌ها $p_i = v$ و مقادیر $p_۱, \ldots, p_{i-۱}$ ثابت

باشند. برای مثال، در درخت‌های $T_{۳,۳}$ تعداد دنباله‌هایی که با ۲,۲,۲ شروع می‌شوند، با تعداد دنباله‌هایی که با ۲,۱,۱ آغاز می‌گردند، مساوی و برابر با $s_{۳,۲,۳,۳}$ می‌باشد.

قضیه ۴.۳ اگر $۱ \leqslant v \leqslant n$ و $۱ \leqslant i \leqslant v(k-۱)$، آنگاه:

$$s_{n,v,i,k} = \frac{kv-v-i+۱}{kn-v-i+۱}\binom{kn-v-i+۱}{n-v}. \tag{۱۰}$$

برهان.

برای اثبات نشان داده می‌شود طرف چپ رابطه فوق در روابط (٦)، (٧)، (٨) و (٩) صدق می‌کند. روابط (٦) و (٧) به سادگی قابل محاسبه می‌باشند. همچنین برای رابطه (٨) به سادگی می توان نوشت:

$$\frac{kv-v-k+۱}{kn-v-k+۱}\binom{kn-v-k+۱}{n-v} = \frac{k(v-۱)-v+۱}{k(n-۱)-v+۱}\binom{k(n-۱)-v+۱}{n-v}.$$

اثبات رابطه (٩) نیز با به کار گیری

$$\frac{kv-v-i}{kn-v-i}\binom{kn-v-i}{n-v} = \binom{kn-v-i}{n-v} - k\binom{kn-v-i-۱}{n-v-۱},$$

و رابطه مثلث خیام پاسکال

$$\binom{m-۱}{r} + \binom{m-۱}{r-۱} = \binom{m}{r},$$

ساده خواهد بود:

$$\frac{k(v+۱)-(v+۱)-i+۱}{kn-(v+۱)-i+۱}\binom{kn-(v+۱)-i}{n-(v+۱)} + \frac{kv-v-(i+۱)+۱}{kn-v-(i+۱)+۱}\binom{kn-v-(i+۱)+۱}{n-v} =$$

$$\binom{kn-v-i}{n-v-۱} - k\binom{kn-v-i-۱}{n-v-۲} + \binom{kn-v-i}{n-v} - k\binom{kn-v-i-۱}{n-v-۱} =$$

$$\binom{kn-v-i+۱}{n-v} - k\binom{kn-v-i}{n-v-۱} =$$

$$\frac{kv-v-i+۱}{kn-v-i+۱}\binom{kn-v-i+۱}{n-v}.$$

□

تعداد دنباله‌های P-Sequence که در موقعیت i اُم مقدار p_i باشد، و زیردنباله متشکل از $i-۱$ عنصر ابتدایی آن ثابت باشد، برابر $s_{n,p_i,i,k}$ است. از این‌رو برای رتبه‌گذاری و رتبه‌گشایی درختان k-تایی منظم با n گره داخلی می‌توان از $s_{n,v,i,k}$ استفاده نمود. یک نمونه محاسبه رتبه یک دنباله P-Sequence در مثال ۲.۳ نشان داده شده است. همان گونه که در ابتدای این بخش گفته شده است، برای محاسبه رتبه براساس هر رقم در جای i اُم کافیست تعداد دنباله‌هایی را محاسبه کرد که زیر دنباله متشکل از $i-۱$ عنصر ابتدایی آن ثابت باشد و در محل i اُم آن مقادیر مجاز کمتر از p_i وجود داشته باشد. در ضمن، از آن رو که دنباله می بایست غیرنزولی باشد، باید در جایگاه i اُم مقادیری را لحاظ کرد که از $p_{i-۱}$ بزرگ‌تر باشد. بنابراین برای هر جایگاه i، برای شمارش تعداد دنباله‌های ماقبل، باید حاصل $\sum_{v=p_{i-۱}}^{p_i-۱} s_{n,v,i,k}$ را محاسبه کرد. این روش در نتیجه ۱.۳ به کار گرفته شده است.

نتیجه ۱.۳. در ترتیب لغت‌نامه‌ایی همه دنباله‌های P-Sequence با طول $n(k-۱)$ رتبه

یک دنباله مانند p به صورت زیر به دست می آید:

$$rank(p) = \sum_{i=۱}^{(n-۱)(k-۱)} \sum_{v=p_{i-۱}}^{p_i-۱} s_{n,v,i,k},$$

که در آن برای سادگی $p_۰ = ۱$ و در صورتی که $p < q$ آنگاه $\sum_{v=p}^{q} = ۰$.

مثال ۲.۳. برای محاسبه رتبه دنباله‌ایی مانند ۳ ۳ ۳ ۳ ۲ ۱ از درختان $T_{۳,۳}$ که در مثال ۱.۳ نشان داده شده است، می‌توان به صورت زیر عمل نمود:

۱) با آغاز از سمت چپ و مشاهده اولین رقم، ۱، باید تعداد دنباله‌هایی را شمرد، که با کمتر از ۱ شروع می‌شوند، که چنین چیزی امکان پذیر نیست.

۲) دومین رقم، ۲، یعنی دنباله‌هایی که با ۱ شروع می‌شوند و در جایگاه دوم مقدار ۱ دارند، یعنی $s_{۳,۱,۲,۳} = ۳$.

۳) سومین رقم، ۳ است؛ باید دنباله‌هایی شمرد که با ۲ ۱ شروع می‌شود و در جایگاه سوم، ۱ یا ۲ دارند. حالت اول امکان‌پذیر نمی‌باشد چرا که دنباله خاصیت غیرنزولی بودن را از دست می دهد؛ اما حالت دوم $s_{۳,۲,۳,۳} = ۲$ می‌شود.

۴) چهارمین رقم، ۳ است؛ باید دنباله‌هایی را شمرد که با ۳ ۲ ۲ شروع می‌شود و در جای چهارم ۱ یا ۲ دارد. هر دو حالت امکان پذیر نمی‌باشد.

بنابراین رتبه دنباله فوق برابر $s_{۳,۱,۲,۳} + s_{۳,۲,۳,۳} = ۵$ می‌باشد.

الگوریتم ۳—۲ با گرفتن رتبه یک درخت، r، دنباله P-Sequence متناظر آن را تولید می کند. روش تولید دنباله دقیقا عکس رتبه‌گذاری می‌باشد. از آنجا که برای n و k ثابت، مقادیر $s_{n,v,i,k}$ غیرصعودی است، از اولین عنصر دنباله شروع می‌کند و مقدار v را برای هر عنصر محاسبه می‌کند. سپس، هر عنصر را مقدارگذاری می کند.

function $BOrderUnrank\,(r,\, n,\, k$: **integer**): **PSeq**;
begin
  $p_0 := 1$;
  for $i := 1$ **to** $(n-1)(k-1)$ **do**
    find an integer like v such that
      $\sum_{u=p_i-1}^{v-1} s_{n,u,i,k} \leqslant r < \sum_{u=p_i-1}^{v} s_{n,u,i,k}$
    $r := r - \sum_{u=p_i-1}^{v-1} s_{n,u,i,k}$;
    $p_i := \max(v, 1 + [\frac{i-1}{k-1}])$;
  end;
  for $j := (n-1)(k-1)$ **to** $n(k-1)$ **do** $p_j := n$;
end;

الگوریتم ۳—۲: الگوریتم پالو برای رتبه‌گشایی درخت‌های k-تایی در B-Order

۴.۳ کدگذاری درخت‌هایی با n گره و m برگ

این بخش به روش نشاندن درخت‌هایی با n گره و m برگ در یک درخت m-تایی و تولید دنباله‌های متناظر آن می‌پردازد. از این رو روابط، تعاریف و عملگرهای لازم برای این بخش و بخش بعد در ابتدا می‌آید.

در این بخش و دو بخش بعد فرض خواهیم کرد که $\mathcal{T}_{n,m}$ مجموعه همه درخت‌های منظم با n گره داخلی و m برگ باشد. بنابراین

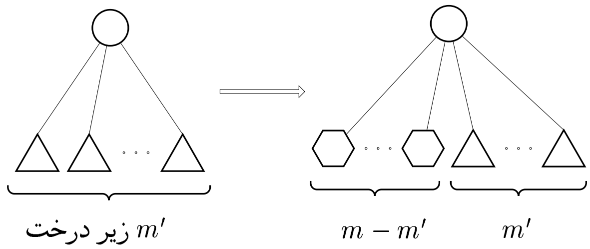
$$|\mathcal{T}_{n,m}| = \frac{1}{n+m-1}\binom{n+m-1}{m}\binom{n+m-1}{m-1}.$$

برای نشاندن هر یک از درخت‌های مجموعه $\mathcal{T}_{n,m}$ در یک درخت m-تایی به سبک زیر عمل می‌شود: به هر گره داخلی از یک درخت از $\mathcal{T}_{n,m}$ آن تعداد برگ اضافه می‌شود که درجه هر گره دقیقا برابر m گردد. این برگ‌ها که برگ مجازی می‌نامیده می‌شود، به گونه‌ایی به هر گره اضافه می‌گردد که پیش از زیردرخت‌های موجود آن قرار گیرند. نمونه‌ایی از این تغییر برای یک گره داخلی در شکل ۳—۳ آمده است.

درخت‌های حاصل، از آن رو که دارای n گره داخلی هستند و هر گره داخلی دارای درجه m است، در مجموعه‌ایی مانند $\widehat{T}_{n,m}$ از درخت‌های m-تایی منظم با n گره قرار می‌گیرند.

درختان این مجموعه دو گروه برگ دارند: m برگ واقعی و $(n-1)(m-1)$ برگ مجازی.

برای یک درخت مانند $\widehat{T}$ از مجموعه $\widehat{T}_{n,m}$، دنباله $\widehat{\mathrm{P}}$-Sequence متناظر به صورت یک دنباله $(p_1, \ldots, p_{n(m-1)})$ می باشد که به صورت زیر به‌دست می‌آید: اگر درخت به صورت پیش‌ترتیب پیمایش شود و n_i تعداد گره‌های داخلی مشاهده شده قبل از برگ i اُم در این پیمایش باشد، آنگاه در صورتی که برگ i اُم برگی واقعی باشد، $p_i = n_i$ و در غیر این صورت $p_i = \widehat{n}_i$. دنباله حاصل با $\widehat{P}_{\widehat{T}}$ نشان داده می‌شود. مثال ۳.۳ نمونه‌ایی از این تبدیل به همراه $\widehat{\mathrm{P}}$-Sequence متناظر ارایه می‌دهد. در این مثال، برای تمایز بین دو نوع

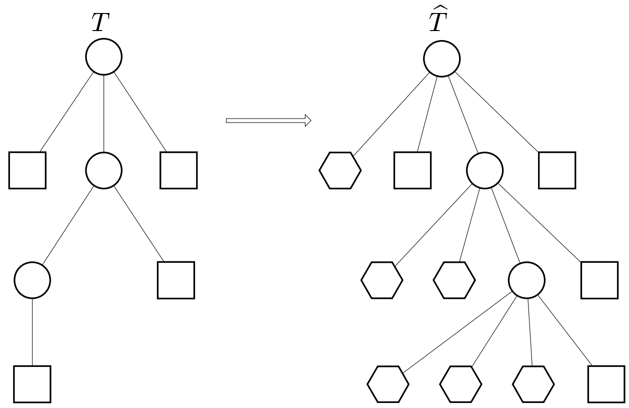


شکل ۳—۳: روش اضافه کردن گره مجازی به گره‌های داخلی یک درخت از $\mathcal{T}_{n,m}$

برگ درخت، برگ‌های واقعی با ۱ و برگ‌های مجازی با ۲ نشانه‌گذاری شده‌اند.

مثال ۳.۳. اگر $n = ۳$، $m = ۴$ و $\widehat{T} = ۰۲۱۰۲۲۰۲۲۲۱۱۱۰$ در آن صورت

$$\widehat{P}_{\widehat{T}} = (\widehat{۱}, ۱, \widehat{۲}, \widehat{۲}, \widehat{۳}, \widehat{۳}, \widehat{۳}, ۳, ۳)$$



برای کدگذاری هر درخت از مجموعه $\mathcal{T}_{n,m}$، می‌توان درخت معادل آن در $\widehat{T}_{n,m}$ را ایجاد نمود و $\widehat{\mathrm{P}}$-Sequence متناظر را به دست آورد. از آن رو که تبدیل فوق و تولید دنباله $\widehat{\mathrm{P}}$-Sequence از درخت m-تایی معادل منحصر به فرد است، می‌توان دانست که به ازای هر درخت از مجموعه $\mathcal{T}_{n,m}$، یک دنباله $\widehat{\mathrm{P}}$-Sequence منحصر به فرد وجود دارد. قضیه ۵.۳ شرط مجاز بودن یک دنباله عددی برای یک درخت از $\mathcal{T}_{n,m}$ را ارایه می‌کند.

از این پس، برای سادگی نمایش، N مجموعه $\{۱, \ldots, n\}$ و $\widehat{N}$ مجموعه $\{\widehat{۱}, \ldots, \widehat{n}\}$ فرض می‌شود.

قضیه ۵.۳ یک دنباله عددی مانند $(p_۱, \ldots, p_{n(m-۱)})$ که در آن $p_i \in N \cup \widehat{N}$ باشد، می‌تواند $\widehat{\text{P}}$-Sequence یک درخت از مجموعه $\mathcal{T}_{n,m}$ باشد اگر و تنها اگر:

۱) دنباله حاصل از حذف ^ها، P-Sequence حاصل، یک درخت از $T_{n,m}$ باشد.
۲) برای هر $i \in [۱, n(m-۱)]$، تعداد p_i هایی که $p_i \in \widehat{N}$ برابر $(n-۱)(m-۱)$ باشد.
۳) برای هر $\widehat{k} \in \widehat{N}$، تعداد $i \in [۱, n(m-۱)]$ هایی که $p_i = \widehat{k}$ کمتر از m است.
۴) برای $k \in [۱, n]$، اگر i و j وجود داشته باشد که $p_i = k$ و $p_j = \widehat{k}$ آنگاه $j < i$.

دنباله $\widehat{\text{P}}$-Sequence تنها روشی نیست که با آن می‌توان درختی از $\widehat{T}_{n,m}$ و در نتیجه درختی از $\mathcal{T}_{n,m}$ را کدگذاری کرد. بر اساس $\widehat{\text{P}}$-Sequence، دو دنباله دیگر ساخته می‌شود که در کنار هم، یک درخت از $\mathcal{T}_{n,m}$ را مشخص می‌کنند. یکی از این دو دنباله R-Sequence و دیگری S-Sequence نامیده می‌شود. هر دو دنباله به طول n می‌باشند و بر اساس روابط زیر از روی یک $\widehat{\text{P}}$-Sequence ساخته می‌شوند:

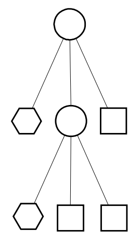
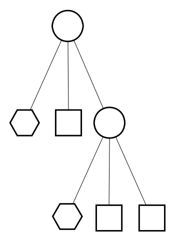
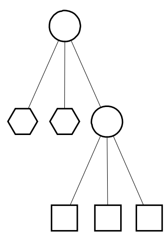
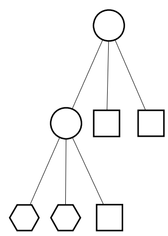
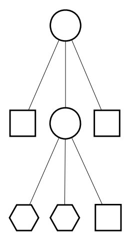
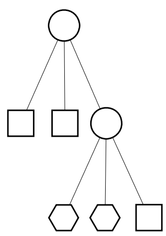
$$\begin{aligned} R_T &= (r_۱, r_۲, \cdots, r_n), \\ S_T &= (s_۱, s_۲, \cdots, s_n), \\ r_i &= \text{card}\{k \in [۱, n(m-۱)] |\ \ p_k = i \ \text{ or } \ p_k = \widehat{i}\ \}, \\ s_i &= \text{card}\{k \in [۱, n(m-۱)] |\ \ p_k = \widehat{i}\ \}. \end{aligned}$$

یک دنباله S-Sequence منتسب به یک دنباله R-Sequence شرایط زیر را داراست:

$$\sum_{i=۱}^{n} s_i = (n-۱)(m-۱),$$
$$\forall i \in [۱, n] : s_i \leqslant \min(m-۱, r_i).$$

یکی از مزیت‌های دنباله‌های R و S نسبت به $\widehat{\mathrm{P}}$-Sequence این است که طول آن دو برای $m > ۲$ از طول دنباله $\widehat{\mathrm{P}}$-Sequence کمتر است. مزیت مهم دیگر آن دو این است که آن‌ها دنباله‌هایی عددی هستند، در حالی که دنباله $\widehat{\mathrm{P}}$-Sequence بر مجموعه $N \cup \widehat{N}$ تعریف می‌شود. از این رو برای کدگذاری درخت‌های $\mathcal{T}_{n,m}$ از این دو دنباله استفاده می‌شود. جدول ۳ـ۳ درختان $\mathcal{T}_{۲,۳}$ که در درختان $\widehat{T}_{۲,۳}$ نشانده شده‌اند به همراه دنباله‌های $\widehat{\mathrm{P}}$-Sequence متناظر و دنباله‌های R-Sequence و S-Sequence نشان داده است.

جدول ۳ـ۳: درختان $\mathcal{T}_{۲,۳}$ که در درختان $\widehat{T}_{۲,۳}$ نشانده شده‌اند

		
$\widehat{P} = (\hat{۱}\hat{۲}۲۲)$	$\widehat{P} = (\hat{۱}۱\hat{۲}۲)$	$\widehat{P} = (\hat{۱}\hat{۱}۲۲)$
$R = (۱۳)\ ,\ S = (۱۱)$	$R = (۲۲)\ ,\ S = (۱۱)$	$R = (۲۲)\ ,\ S = (۲۰)$
		
$\widehat{P} = (\hat{۲}\hat{۲}۲۲)$	$\widehat{P} = (۱\hat{۲}\hat{۲}۲)$	$\widehat{P} = (۱۱\hat{۲}\hat{۲})$
$R = (۰۴)\ ,\ S = (۰۲)$	$R = (۱۳)\ ,\ S = (۰۲)$	$R = (۲۲)\ ,\ S = (۰۲)$

۵.۳ الگوریتم‌های تولید درخت‌هایی با n گره و m برگ

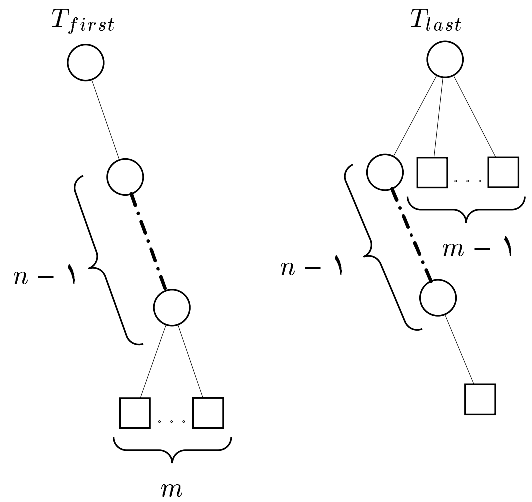
در این بخش دو روش برای تولید درخت‌هایی با n گره و m برگ ارایه می‌شود. این روش‌ها مستقل از یکدیگر می‌باشند و در هر یک از آنها، دنباله‌های R و S به دلایل مطرح شده قبل به عنوان خروجی تولید می‌شوند. بدیهی است که به سادگی می‌توان از روی این دنباله‌ها $\widehat{\text{P}}$-Sequence متناظر را ساخت. بخش‌های ۱.۵.۳ و ۲.۵.۳ این دو روش را مطرح می‌کنند. بخش نهایی به تحلیل زمانی این دو روش می‌پردازد.

۱.۵.۳ روش اول تولید درخت‌های $\mathcal{T}_{n,m}$

در روش اول تولید درخت‌هایی با n گره و m برگ، از الگوریتم ۳—۱ برای تولید درخت‌های مجموعه $\widehat{T}_{n,m}$ استفاده می‌شود. به این صورت که این الگوریتم بدون در نظر گرفتن خاصیت $\hat{}$عناصر بر روی $\widehat{\text{P}}$-Sequence عمل می‌کند. در عمل، ابتدا P-Sequence زیر مجموعه‌ایی از درخت‌های $T_{n,m}$ تولید می‌شوند. در گام دوم، دنباله R-Sequence متناظر هر P-Sequence ایجاد می‌شود. سرانجام، S-Sequence های متناظر به یک دنباله R توسط الگوریتم ۳—۳ تولید می‌گردند. شرح هر یک از گام‌های تولید در ادامه می‌آید.

در گام نخست، زیرمجموعه‌ایی از درخت‌های m-تایی منظم با n گره تولید می‌شود. نکته مهم یافتن زیرمجموعه مناسب از $T_{n,m}$ برای تولید این درخت‌هاست.

بدیهی است که درخت آغازین $\mathcal{T}_{n,m}$ در ترتیب B-Order، درختی است که درجه گره‌های داخلی، غیر از گره آخر، همگی یک باشد و همه برگ‌ها فرزند آخرین گره باشند. این



شکل ۳—۴: درخت اول و آخر مجموعه $\mathcal{T}_{n,m}$ در ترتیب B-Order

درخت در شکل ۳—۴ با نام T_{first} نشان داده شده است. در تبدیل این درخت به درختی از $\widehat{T}_{n,m}$، درختی به نام $\widehat{T}_{first}$ بدست می‌آید که فرزندان همه گره‌های داخلی، غیر از گره آخر، شامل $m-۱$ برگ مجازی و یک گره داخلی است. همچنین، گره آخر m برگ واقعی دارد. از این رو، دنباله $\widehat{\text{P}}$-Sequence حاصل به صورت زیر می‌باشد:

$$\widehat{P}_{\widehat{T}_{first}} = (\ \underbrace{\widehat{۱},\ldots,\widehat{۱}}_{m-۱},\ \underbrace{\widehat{۲},\ldots,\widehat{۲}}_{m-۱},\ \ldots\ ,\ \underbrace{\widehat{n-۱},\ldots,\widehat{n-۱}}_{m-۱},\ \underbrace{n,\ldots,n}_{m-۱}\ )$$

درخت پایانی از مجموعه $\mathcal{T}_{n,m}$ در ترتیب B-Order، درختی است که درجه گره ریشه بیشترین درجه ممکن، m، باشد. این درخت در شکل ۳—۴ با نام T_{last} نشان داده شده است. در تبدیل این درخت به درختی از $\widehat{T}_{n,m}$ درختی به نام $\widehat{T}_{last}$ بدست می‌آید که فرزندان همه گره‌های داخلی، غیر از گره اول، شامل $m-۱$ برگ مجازی است. از این رو، دنباله $\widehat{\text{P}}$-Sequence حاصل این‌گونه است:

$$\widehat{P}_{\widehat{T}_{last}} = (\ \underbrace{\widehat{۲},\ldots,\widehat{۲}}_{m-۱},\ \underbrace{\widehat{۳},\ldots,\widehat{۳}}_{m-۱},\ \ldots\ ,\ \underbrace{\widehat{n},\ldots,\widehat{n}}_{m-۱},\ \underbrace{n,\ldots,n}_{m-۱}\ )$$

تعداد P-Sequence هایی که توسط الگوریتم ۳—۱ تولید می‌شود، برابر است با

$$\begin{aligned}
rank(T_{last}) &= rank(\underbrace{۲,\ldots,۲}_{m-۱},\ \underbrace{۳,\ldots,۳}_{m-۱},\ \ldots,\ \underbrace{n,\ldots,n}_{m-۱},\ \underbrace{n,\ldots,n}_{m-۱}\ ) \\
&= \sum_{i=۱}^{(n-۱)(m-۱)} \sum_{j=p_{i-۱}}^{p_i-۱} s_{n,j,i,m} \\
&= \sum_{i=m,۲m,\cdots,(n-۲)m} \sum_{j=p_{i-۱}}^{p_i-۱} s_{n,j,i,m} \\
&= \sum_{k=۱}^{n-۲} \sum_{j=p_{km-۱}}^{p_{km}-۱} s_{n,j,km,m} \qquad \text{که} \qquad p_{km-۱}=k+۱,\ p_{km}=k+۲ \\
&= \sum_{k=۱}^{n-۲} \sum_{j=k+۱}^{k+۱} s_{n,j,km,m} \\
&= \sum_{k=۱}^{n-۲} s_{n,k+۱,km,m}
\end{aligned}$$

در گام دوم، باید دنباله R متناظر با هر P-Sequence را تولید کرد. تولید R-Sequence بر اساس تعریف آن بسیار ساده است. هر r_i تعداد عناصری از دنباله P-Sequence می‌باشد که مقدار آن‌ها با i برابر است. این مرحله در $O(nk)$ امکان‌پذیر می‌باشد.

در گام نهایی، برای هر R-Sequence باید دنباله‌های S متناظر را تولید نمود. این گام را الگوریتم ۳—۳ انجام می‌دهد. هر s_i تعداد عناصری از P-Sequence است که مقدار آن برابر $\hat{i}$ می‌باشد؛ بنابراین برای هر $i \in [۱,n]$، عنصر s_i از r_i کوچکتر است. همچنین بر اساس قضیه ۵.۳ عنصر s_i از m کوچک‌ترند. و حاصل s_i ها برابر $(n-۱)(m-۱)$ است. الگوریتم ۳—۳ دو دنباله S و S' تشکیل می‌دهد که به ترتیب اولین و آخرین دنباله ممکن است. در تشکیل دنباله S، بزرگ‌ترین مقدار ممکن را در آخرین عنصر قرار می‌دهد. باقی‌مانده به ترتیب از انتها در دنباله قرار می‌گیرد. دنباله S' عکس دنباله S از ابتدا ساخته

1 **function** *GenSSequence*(R: **RSeq**; n, m: **integer**): **SSeq**;
2 **begin**
3 $s_n := \min(m-1, r_n)$; $t := (n-1)(m-1) - s_n$;
4 $s'_1 := \min(m-1, r_1)$; $t' := (n-1)(m-1) - s'_1$;
5 **for** $i := 2$ **to** n **do**
6 $s_{n+1-i} := \min(m-1, r^p_{n+1-i}, t)$; $t := t - s_{n+1-i}$;
7 $s'_i := \min(m-1, r_i, t')$; $t' := t' - s'_i$;
8 **end**;
9 **if** $t > 0$ **then** exit;
10 **output** $(s_1, \ldots, s_n)$;
11 **while exists** $i = \min\{k | s_k < s'_k\}$ **do**
12 $s_i := s_i + 1$;
13 $t := \sum_{j=i+1}^{n} s_j - 1$;
14 **for** $j := n$ **downto** $i+1$ **do**
15 $s_j := \min(m-1, r_j, t)$;
16 $t := t - s_j$;
17 **end**;
18 **output** $(s_1, \ldots, s_n)$;
19 **end**;
20 **end**;

الگوریتم ۳—۳: الگوریتم پالو برای تولید دنباله‌های S متناظر با یک R-Sequence

می‌شود. سپس در حلقه دستورالعمل خط ۱۱ دنباله‌های میانی تا آخرین دنباله ساخته می‌شود. روش اول، یک روش مستقیم نیست. در این روش سعی شده‌است دقیقا از الگوریتم تولید درختان m-تایی استفاده شود و پس از تولید هر درخت $T_{n,m}$، با جاگذاری گره‌های مجازی، درخت‌هایی از $\widehat{T}_{n,m}$ بدست آورد. بدیهی است که به ازای هر درخت m-تایی، یک یا چند درخت m-تایی با گره مجازی بدست می‌آید. در نتیجه، در این روش، درخت‌ها بر اساس ترتیب خاصی تولید نمی‌شوند.

۲.۵.۳ الگوریتم مستقیم تولید درخت‌های $\mathcal{T}_{n,m}$

الگوریتم ۳–۴ به طور مستقیم بر روی یک $\widehat{\text{P}}$-Sequence عمل می‌کند. این الگوریتم با دریافت یک $\widehat{\text{P}}$-Sequence معتبر، دنباله متناظر درخت بعدی در ترتیب B-Order را تولید می‌کند.

روش عملکرد این الگوریتم بر روی دنباله ورودی بسیار نزدیک به الگوریتم ۳–۱ است. بدین صورت که، انتهایی‌ترین جایگاه تفاوت دنباله ورودی با دنباله پایانی یافت می‌شود. عنصر مورد نظر بر اساس ترتیب زیر، یک واحد افزایش داده می‌شود.

$$\widehat{۱} < ۱ < \widehat{۲} < ۲ < \cdots < \widehat{n} < n$$

مابقی دنباله همانند یک P-Sequence ساده بدون در نظر گرفتن سمبل های^ساخته می‌شود. سرانجام، همه سمبل های^حذف شده از عناصر که تعدادشان k است، با توجه به اینکه هر گره داخلی حداکثر $m - ۱$ گره مجازی می‌تواند داشته باشد، به ادامه دنباله باز گردانده می‌شوند.

در این الگوریتم، دستور $p_i := \widehat{p_i}$ به معنای قرار دادن^بر روی عنصر i اُم دنباله به کار رفته است. همچنین، منظور از دستور $p_i := \widehat{p_i + ۱}$، افزایش یک واحدی عنصر i اُم و قرار دادن سمبل^بر روی آن است. در ضمن هدف از $|p_i|$، مقدار عنصر i اُم با حذف سمبل^ آن می‌باشد.

function $BOrderNext\widehat{P}Seq$ ($\widehat{P}$: $\widehat{\mathbf{P}}$**Seq**; n, m: **integer**): $\widehat{\mathbf{P}}$**Seq**;
begin
    $i := \min\{j | (p_{j+1}, \cdots, p_{n(m-1)})$ subsequence of $\widehat{T}_{last}\}$;
    $k := \text{card}\{j \in [i, n(m-1)] | \; p_j \in \widehat{N} \; \}$;
    if $p_i \in N$ **then**
        $p_i := \widehat{p_i + 1}$; $q := 1$; $k := k - 1$;
    else
        $p_i := |p_i|$; $q := m - 1$;
    for $j := i + 1$ **downto** $n(m-1)$ **do**
        $p_j := \max(|p_i|, 1 + [\frac{j-1}{m-1}])$;
    for $j := i + 1$ **downto** $n(m-1)$ **do**
        if $k = 0$ **then return**$(\widehat{P})$;
        else
        begin
            $p_j := \widehat{p_j}$; $k := k - 1$;
            if $p_j \neq |p_{j-1}|$ **then** $q := 1$;
            else if $q < m - 1$ **then** $q := q + 1$;
        end
end;

الگوریتم ۳—۴: الگوریتم پالو برای تولید درختان $\mathcal{T}_{n,m}$ در B-Order

۳.۵.۳ تحلیل زمانی و مقایسه الگوریتم‌های تولید $\mathcal{T}_{n,m}$

روش اول ارایه شده برای تولید درختان $\mathcal{T}_{n,m}$ روشی مستقیم نیست. ابتدا الگویتم ۳—۱ برای تولید P-Sequence هایی در بازه درختان T_{first} تا T_{last} اجرا می‌شود که دارای پیچیدگی زمانی متوسط $O(m)$ برای هر دنباله P-Sequence است. تعداد اجرای این الگوریتم و دنباله‌های P-Sequence تولید شده به وضوح $rank(T_{last})$ است. سپس، برای هر یک از دنباله‌های تولید شده، توسط الگوریتم ۳—۳ دنباله های S مناسب تولید می‌شود. این فرایند به طور مستقیم در زمان $O(n)$ انجام می‌شود. اگر شرط $|\mathcal{T}_{n,m}| < rank(T_{last})$ برقرار باشد، به طور میانگین کل این روش را می‌توان با پیچیدگی زمانی $O(n)$ دانست.

در روش دوم الگوریتم ۳—۴ برای تولید هر $\widehat{\text{P}}$-Sequence دارای پیچیدگی زمانی $O(nm)$ می‌باشد.

۴

تولید درخت‌هایی با n گره و m برگ در A-Order

در این فصل الگوریتم جدیدی برای تولید درخت‌هایی با n گره داخلی و m گره خارجی ارایه خواهد شد. در بخش اول این فصل به کد گذاری این درخت‌های توسط دنباله جدیدی به نام E-Sequence پرداخته می‌شود و سپس در بخش ۲.۴ تولید این درخت ها در ترتیب A-order ارایه می‌گردد. ترتیب A-Order برای این نوع درخت‌ها که فاقد گره‌های خارجی می‌باشند، با تعریف ارایه شده در فصل اول متفاوت است. نشان داده می‌شود که الگوریتم ارایه شده دارای نرخ رشدی از $O(n+m)$ می‌باشد. در بخش سوم رتبه‌گذاری و رتبه‌گشایی درختان مذکور به تفصیل ارایه می‌شود.

۱.۴ کدگذاری درخت‌هایی با n گره داخلی و m برگ

در این بخش نیز بر اساس نشان‌گذاری ارایه شده در فصل قبل، مجموعه درخت‌هایی که n گره و m برگ دارند با $\mathcal{T}_{n,m}$ نشان داده می‌شود. تعداد این درخت‌ها همچنان که در بخش ۴.۳ آمده است برابر است با

$$|\mathcal{T}_{n,m}| = \frac{1}{n+m-1}\binom{n+m-1}{m}\binom{n+m-1}{m-1} \qquad (1)$$

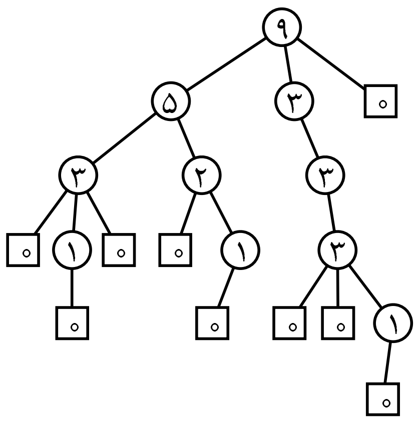
بدیهی است که اگر T عضوی از مجموعه $\mathcal{T}_{n,m}$ باشد، تعداد کل گره‌های آن برابر $n+m$ خواهد بود.

برای کدگذاری درخت‌هایی با n گره و m از یک دنباله از اعداد صحیح استفاده می‌شود. این دنباله پیشتر در متون مربوط به تئوری علوم کامپیوتر استفاده نشده است. این دنباله جدید E-Sequence نامیده خواهد شد. E-Sequence با تعریف ارایه شده زیر قابل تعمیم به همه درختان مرتب و ریشه دار می‌باشد.

تعریف ۱.۴ برای یک درخت از مجموعه $\mathcal{T}_{n,m}$، مانند T، دنباله E-Sequence متناظر که با E_T نشان داده می‌شود، یک دنباله صحیح به صورت $(e_1, \ldots, e_{n+m})$ است که e_i ها برچسب گره‌هادر پیمایش پیش‌ترتیب هستند. برچسب هر گره، تعداد برگ‌های موجود در زیر درختی است که آن گره ریشه آن باشد. برچسب هر برگ براساس تعریف صفر می‌باشد. $\triangledown$

به عبارت دیگر، به هر گره درخت T عددی متناظر می‌شود که این عدد، تعداد برگ‌هایی است که آن گره، جد کامل آن‌هاست. سپس، برچسب‌های عددی برگ‌های درخت، به صورت پیش‌ترتیب پیمایش می‌شود که دنباله حاصل E-Sequence متناظر درخت T است.

مثال ۱.۴. اگر $n = ۱۰$، $m = ۹$ و در آن صورت برای درخت زیر دنباله E-Sequence به صورت زیر است



$$E_T = (۹, ۵, ۳, ۰, ۱, ۰, ۰, ۲, ۰, ۱, ۰, ۳, ۳, ۳, ۰, ۰, ۱, ۰, ۰)$$

واضح است به شرطی که برچسب متناظر برگ‌ها هم عدد یک در نظر گرفته شوند، برچسب هر گره، برابر با جمع برچسب‌های فرزندان آن است. از این رو، مساله تولید درخت‌هایی با n گره داخلی و m برگ، معادل پارتیشن نمودن عدد m به صورت بازگشتی است که در نهایت $n + m$ پارتیشن به دست بیاید. این خاصیت در یکی از روش‌های رتبه‌گشایی و رتبه‌گذاری درختان $\mathcal{T}_{n,m}$ به کار رفته است.

قضیه ۱.۴ شرط معتبر بودن یک دنباله E-Sequence را ارایه می‌دهد. پیش از آن می‌بایست $z_{i,j}$ را تعریف نمود.

$z_{i,j}$ بر روی یک دنباله E-Sequence تعداد صفرهایی است که از از اندیس $i+۱$ تا j قرار دارد. به عبارت دیگر تعداد برگ‌هایی از درخت است که در پیمایش پیش‌ترتیب، در بین مکان‌های $i+۱$ اُم و j اُم قرار دارند. برای سادگی z_i را همان $z_{i,n+m}$ در نظر گرفته می‌شود. بدیهی است که مقدار $z_{i,j}$ برابر $z_i - z_j$ است.

قضیه ۱.۴ یک دنباله عددی مانند $(e_۱, \ldots, e_{n+m})$ می‌تواند E-Sequence برای یک درخت با n گره داخلی و m برگ باشد اگر و تنها اگر:

۱) $e_۱ = z_۱ = m$.

۲) برای هر $i \in [۱, n+m)$ که $e_i > ۰$ باید اندیسی مانند $j \in (i, n+m]$ وجود داشته باشد که $e_j = ۰$ و $z_{i,j} = e_i$.

۳) اگر برای یک i و j شرط قبل صدق کند، آنگاه برای هر $k \in (i,j)$ وجود دارد: $e_i - e_k \geqslant z_{i,k}$.

برهان.

به سادگی می‌توان نشان داد که برای دنباله E-Sequence هر درخت مانند T از مجموعه $\mathcal{T}_{n,m}$ شرایط فوق برقرار است و بالعکس. شرط اول بدیهی است: تعداد صفرهای دنباله که برابر تعداد برگ‌های درخت است در طول دنباله $z_{۱,n+m} = m$ است. این مقدار گره ریشه است که در پیمایش پیش‌ترتیب پیش از همه دیده می‌شود. شرط (۲) حکم می‌کند که به ازای هر عدد غیر صفر در دنباله، e_i، مکانی پس از آن در دنباله وجود دارد، j، که تعداد صفرها از i تا j برابر مقدار e_i باشد. هر عدد غیر صفر نمایانگر یک گره داخلی است. گره داخلی i زیر درخت‌هایی دارد که مجموع تعداد برگ‌های آن زیر درخت‌ها، مقدار ریشه، e_i، است. در پیمایش پیش‌ترتیب، آن گره داخلی قبل از زیر درختانش مشاهده می‌شود. همچنین آخرین گره مشاهده شده از مجموعه زیر درختان گره i یک برگ با مقدار صفر

است. بنابراین، در فاصله گره i تا برگ پایانی (با اندیس j) دقیقا e_i برگ وجود دارد. به عبارت دیگر $z_{i,j} = e_i$. شرط (۳) وجود و صحت زیردرخت‌ها را برای گره داخلی i تضمین می‌نماید. یک گره داخلی مانند i با مقدار e_i در نظر گرفته شود. بر اساس شرط قبل، همه زیر درخت‌های این گره قبل از عنصر j اُم قرار می‌گیرد. بدیهی است برای هر گره از نوادگان i مانند k رابطه $e_i \geqslant e_k$ برقرار است. همچنین میزان فاصله مقادیر گره i و k بیشتر از تعداد برگ‌هایی است که در فاصله این دو گره مشاهده شده است؛ یعنی $e_i - e_k \geqslant z_{i,k}$. □

در تعریف A-Order ارایه شده در فصل اول، معیار مقایسه درخت‌ها، تعداد گره‌های آن است. به عبارت دیگر، ارزش هر گره مانند x، تعداد گره‌های داخلی در زیر درختی با ریشه x است. در این فصل، برای مقایسه درختان مجموعه $\mathcal{T}_{n,k}$ از تعداد برگ‌های درخت‌ها برای مقایسه آن‌ها استفاده می‌شود. در صورتی که $\mathcal{L}(T)$ تعداد برگ‌های زیر درخت T باشد، تعریف ۲.۴ ترتیب A-Order را برای درخت‌های این مجموعه ارایه می‌کند.

تعریف ۲.۴ برای دو درخت از مجموعه $\mathcal{T}_{n,k}$ مانند T و T' گویند $T \prec_{A} T'$ اگر:

۱) $\mathcal{L}(T) < \mathcal{L}(T')$، یا

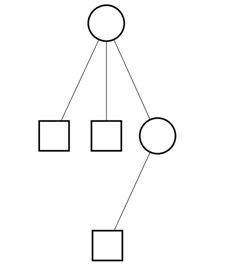
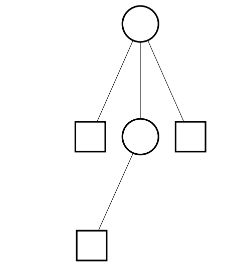
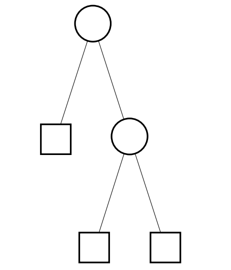
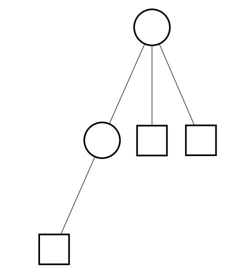
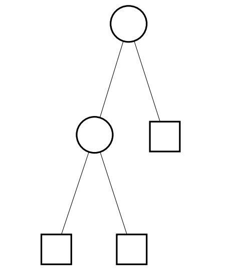
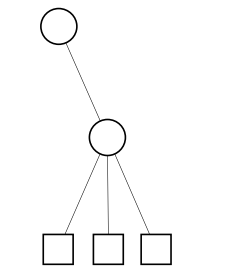
۲) $\mathcal{L}(T) = \mathcal{L}(T')$ و با فرض i بزرگترین مقداری که برای $j = ۱, \ldots, i$ وجود داشته باشد $T_j = T'_j$، آنگاه باید:

(a) $i = deg(T)$، یا

(b) $T_{i+۱} \prec_{A} T'_{i+۱}$

▽

جدول ۴ـ۱: درختان $\mathcal{T}_{۲,۳}$ به همراه E-Sequence متناظر

		
$E =$ (۳۰۰۱۰)	$E =$ (۳۰۱۰۰)	$E =$ (۳۰۲۰۰)
		
$E =$ (۳۱۰۰۰)	$E =$ (۳۲۰۰۰)	$E =$ (۳۳۰۰۰)

در جدول ۴ـ۱ درختان $\mathcal{T}_{۲,۳}$ را به همراه دنباله‌های E-Sequence متناظر نشان داده شده است. چنانکه مشهود است درخت های نمایش داده شده به ترتیب A-Order مرتب شده‌اند. این مساله در قضیه ۲.۴ مورد بررسی قرار گرفته است. در این قضیه اثبات خواهد شد که ترتیب A-Order درخت‌های $\mathcal{T}_{n,m}$ با ترتیب لغت‌نامه‌ایی E-Sequence های متناظر مطابق است.

قضیه ۲.۴ برای دو درخت T و T' از مجموعه $\mathcal{T}_{n,k}$ رابطه $T \prec_{A} T'$ صحیح است اگر و تنها اگر دنباله E-Sequence متناظر درخت T (E_T) به صورت لغت‌نامه‌ایی از $E_{T'}$ کوچک‌تر باشد.

برهان.

فرض می‌شود که دنباله‌های E_T و $E_{T'}$ به ترتیب، دنباله‌های متناظر دو درخت T و T' باشند و i کوچک‌ترین اندیسی است که در آن جایگاه $E_T(i) < E_{T'}(i)$ باشد و برای همه مقادیر $j = ۱, ۲, \dots, i - ۱$، عناصر دنباله‌ها $E_T(j) = E_{T'}(j)$ باشند. این تنها در صورتی امکان‌پذیر است که تعداد برگ‌های گره i اُم در پیمایش پیش‌ترتیب، در درخت T از درخت T' کمتر باشد؛ و یا به عبارت دیگر، درخت T پیش از درخت T' در ترتیب A-Order باشد.

□

۲.۴ الگوریتم تولید درخت‌های $\mathcal{T}_{n,m}$

الگوریتم ۴—۱ درخت‌هایی با n گره داخلی و m برگ را در ترتیب A-Order را تولید می‌کند. چنانچه پیشتر دیده شد، در صورتی که درخت‌های در ترتیب A-Order باشند، E-Sequence های متناظر در ترتیب لغت‌نامه‌ایی قرار می‌گیرند.

دنباله E-Sequence آغازین در ترتیب لغت‌نامه‌ایی به صورت زیر می‌باشد:

$$E_{T_f} = (m, \underbrace{۰, \dots, ۰}_{m-۱}, \underbrace{۱, \dots, ۱}_{n-۱}, ۰)$$

و دنباله پایانی اینگونه است:

$$E_{T_l} = (\underbrace{m, m, \dots, m}_{n}, \underbrace{۰, \dots, ۰, ۰}_{m})$$

الگوریتم ۴—۱ یک E-Sequence معتبر و متناظر به یک درخت از مجموعه $\mathcal{T}_{n,m}$ را به عنوان ورودی دریافت می‌کند و دنباله بعدی را در صورت وجود تولید می‌کند.

```
1   function AOrederNextESeq(var E: ESeq; n, m: integer);
2   begin
3      zreo := E_1;
4      for i := 2 to n + m do
5         if E_i = 0 then zero := zero − 1;
6         Z_i := zero;
7      end;
8      P := ProduceParentSequence(E);
9      for i := n + m − 1 downto 2 do
10        if (E_i = ۰ and n+m−i > Z_i) or (E_i ≠ ۰ and E_i < E_{P_i} −(Z_{P_i} −Z_i))
11        then
12           if E_i = 0 then Z_i := Z_i + 1;
13           E_i := E_i + 1;
14           for j := 1 to Z_i − 1 do   E_{i+j} = 0;
15           for j := 1 to n + m − i − Z_i do   E_{i+Z_i+j−1} = 1;
16           return E;
17        end;
18  end;
```

الگوریتم ۴−۱: الگوریتم تولید درختان $\mathcal{T}_{n,m}$ در A-Order

در الگوریتم ۴−۱ دو آرایه کمکی به نام‌های Z و P به کار رفته است که هر کدام به طول $n+m$ می‌باشند و در ابتدا محاسبه می‌شوند. اولین آرایه، Z، ارزشی است که پیشتر معرفی شده است؛ Z_i نمایانگر تعداد صفرهای دنباله از موقعیت $i+۱$ اُم تا انتهای دنباله می‌باشد. این آرایه به سادگی در $O(n+m)$ در اولین حلقه الگوریتم ۴−۱ (دستورالعمل‌های سوم تا هفتم) بدست می‌آید. آرایه بعدی، P، نیز یک آرایه عددی برای $i = ۱, \ldots, n+m$ است که حاوی لیست پدرهای درخت متناظر با دنباله ورودی است. به عبارت دیگر، P_i برابر اندیس پدر گره i اُم است. این آرایه توسط الگوریتم ۴−۲ بدست می‌آید که بعدا

```
function ProduceParentSequence(E: ESeq): ParentList;
begin
    stack.Push(1);
    P_1 := λ;
    for i := 2 to n + m do
        if E_i = 0 then E_{P_i} := E_{P_i} − 1;
        else E_{P_i} := E_{P_i} − E_i;
        if E_{P_i} = 0 then stack.Pop();
        if E_i ≠ 0 then stack.Push(i);
    end;
    return P;
end;
```

الگوریتم ۴—۲: الگوریتم تولید لیست پدرهای یک E-Sequence

شرح داده می‌شود.

در الگوریتم ۴—۱ برای تولید دنباله بعدی، دنباله ورودی از انتها پیمایش می‌شود و راست‌ترین عنصر قابل افزایش معین می‌شود. اگر i اندیس راست ترین عنصر قابل افزایش باشد، عنصر e_i دارای این مشخصات است: اگر $e_i = 0$ باشد، در آن صورت یک صفر از دنباله حذف خواهد شد. پس ادامه دنباله باید جای کافی برای جای دادن یک صفر داشته باشد یا $z_i < n + m - i$. در غیر این صورت، اگر $e_i \neq 0$ باشد، در آن صورت گره i اُم، ریشه زیردرختی است که e_i برگ دارد؛ افزایش آن به معنی قرار دادن یک برگ بیشتر در آن زیردرخت است. این عمل در صورتی ممکن است که مقدار گره i، از مقدار پدر گره i (با در نظر نگرفتنِ برگ‌هایی که بین پدر i تا i مشاهده شده‌اند) بیشتر نباشد یا $e_i < e_{p_i} - z_{p_i,i}$.

پس از یافتن عنصر مورد نظردر اندیس i و افزایش مقدار آن، $n + m - i$ عنصر ادامه با کوچک‌ترین دنباله ممکن با z_i صفر پر می‌شود. یعنی، $z_i - 1$ مقدار صفر و $n+m-i-z_i$

مقدار یک در ادامه دنباله قرار می‌گیرد.

الگوریتم ۴—۲ برای تولید آرایه P، لیست اندیس پدرها، در الگوریتم تولید بکار می‌رود. بر اساس تعریف، i اُمین گره در پیمایش پیش‌ترتیب، i اُمین عنصر E-Sequence (یا e_i) می‌باشد و فرض می‌شود که p_i اُمین گره مشاهده شده، در پیمایش پیش‌ترتیب (یا عنصر e_{p_i}) پدر آن باشد. در این صورت، e_{p_i} نزدیک‌ترین گره داخلی ماقبل e_i است که در فاصله این دو گره، تعداد برگ‌ها (صفرها) کمتر از $e_{p_i} - e_i$ باشد. این الگوریتم به کمک یک پشته، آرایه P را با پیچیدگی زمانی $O(n+m)$ بدست می آورد.

پیچیدگی زمانی الگوریتم ۴—۱ براساس زمان محاسبه دو آرایه P و Z و حلقه اصلی که در آن انتهایی‌ترین عنصر قابل افزایش پیدا می‌شود، برابر $O(n+m)$ است.

۳.۴ رتبه‌گذاری و رتبه‌گشایی درخت‌های $\mathcal{T}_{n,m}$

در این بخش به الگوریتم هایی پرداخته می‌شود که رتبه یک درخت از مجموعه $\mathcal{T}_{n,m}$ را از روی دنباله E-Sequence متناظر آن تولید می‌کند و بر اساس رتبه داده شده E-Sequence متناظر را تولید می‌کند. روش ارایه شده در این بخش، مبتنی بر مباحث ابتدایی بخش ۳.۳ است. بدین معنی که، اگر مجموعه‌ایی از دنباله‌های عددی با طول N وجود داشته باشد که به ترتیب الفبایی تولید شده باشند، برای به دست آوردن رتبه یک دنباله مانند $a_۱, a_۲, \ldots, a_N$ در آن مجموعه ابتدا برای همه مقادیر $k = ۱, \ldots, N$، عددی مانند s_k محاسبه می‌شود که تعداد دنباله‌هایی است که $k - ۱$ عناصر ابتدایی آن $a_۱, a_۲, \ldots, a_{k-۱}$

```
function AOrderRank(E: ESeq; n, m: integer): integer;
begin
    r := 0;
    for k := 2 to n + m − 1 do
        for c := 0 to E_k − 1 do
            r := r+CSW(E_1, E_2, ···, E_{k−1}, c);
        end;
    end;
    return r;
end;
```

الگوریتم ۴—۳: الگوریتم رتبه‌گذاری درختان $\mathcal{T}_{n,m}$ در A-Order

و عنصر k اُم آن برابر با ۰، ۱، ... و $a_k - ۱$ باشد؛ سپس، رتبه دنباله فوق برابر $\sum_{k=۱}^{N} s_k$ می‌باشد.

در صورتی که `CSW()` تابعی باشد که با دریافت یک زیردنباله آغازین، تعداد دنباله‌های E-Sequence که با آن زیردنباله شروع می‌شود را محاسبه می‌کند، الگوریتم ۴—۳ رتبه یک دنباله ورودی را محاسبه می‌کند. این الگوریتم به سادگی از روش توضیح داده شده بالا استفاده می‌کند. بدیهی است، از آن رو که همه E-Sequence های متناظر به درختان $\mathcal{T}_{n,m}$ با عنصر $E_۱ = m$ شروع می‌شوند، و عنصر انتهایی آن‌ها همواره $E_{n+m} = ۰$ است، مقادیر k از ۲ تا $n + m - ۱$ در نظر گرفته می‌شود.

چنانچه تابع `CSW()` دارای پیچیدگی زمانی $O(C)$ باشد، بدیهی است که الگوریتم رتبه‌گذاری ۴—۳ در بدترین حالت دارای پیچیدگی زمانی $O(nmC)$ خواهد بود؛ چرا که m عنصر هر E-Sequence متناظر به هر درخت از $\mathcal{T}_{n,m}$ برابر صفر است که حلقه داخلی برای آن اجرا نمی‌شود و همچنین m حد بالایی برای مابقی عناصر دنباله است. بنابراین تابع `CSW()`

```
function AOrderUnrank (r, n, m : integer): ESeq;
begin
   E_1 := m;
   for i := 2 to n + m do
      find an integer like c such that
         CSW(E_1, ..., E_{i-1}, c) ⩽ r < CSW(E_1, ..., E_{i-1}, c + 1)
      r := r − CSW(E_1, ..., E_{i-1}, c);
      E_i := c;
   end;
end;
```

الگوریتم ۴—۴: الگوریتم رتبه‌گشایی درختان $\mathcal{T}_{n,m}$ در A-Order

حداکثر nm مرتبه فراخوانی می‌شود.

الگوریتم ۴—۴، به کمک تابع `CSW()` براساس رتبه داده شده، دنباله E-Sequence متناظر درختی را تولید می‌کند که در ترتیب A-Order در آن جایگاه باشد. پیچیدگی زمانی این الگوریتم همانند الگوریتم رتبه‌گذاری $O(nmC)$ خواهد بود.

روش‌های مختلفی برای تولید تابع `CSW()` وجود دارد. این بخش دو روش مختلف از این روش‌ها را مورد بررسی قرار می‌دهد. روش اول که مبتنی بر پارتیشن نمودن اعداد صحیح است، از خاصیت دنباله‌های E-Sequence، که پیشتر در مثال ۱.۴ مطرح شده است، استفاده می‌کند. این روش به طور خلاصه با یک مثال در ۱.۳.۴ ارایه می‌شود. روش دوم یک ساختار داده‌ایی برای ذخیره اطلاعات تابع `CSW()` تولید می‌کند که به تفصیل در ۲.۳.۴ شرح داده می‌شود.

۱.۳.۴ محاسبه تابع $\mathrm{CSW}()$ به کمک پارتیشن اعداد صحیح

یکی از روش‌های محاسبه تابع `CSW()` استفاده از الگوریتم‌های پارتیشن اعداد صحیح است. چنانچه پیشتر در مثال ۱.۴ گفته شده است، دنباله‌های E-Sequence متناظر به درختان $\mathcal{T}_{n,m}$ به صورت بازگشتی آنقدر به پارتیشن کردن عدد m می‌پردازند تا در نهایت $n+m$ بخش بدست آید. از این رو، تعداد کل درختان $\mathcal{T}_{n,m}$، برابر تعداد پارتیشن‌های ممکن عدد m تا $n+m$ بخش است. در این بخش نشان داده می‌شود که می‌توان تعداد حالت‌هایی را که بخش‌بندی یک زیردنباله آغازین داده شده تکمیل می‌شود، شمارش نمود.

این بخش، الگوریتمی ارایه نمی‌کند. تنها روش شمارش برای یک دنباله نمونه شرح داده می‌شود. مجموعه درختان مورد بررسی $\mathcal{T}_{۳,۴}$ است و تعداد دنباله‌هایی از E-Sequence متناظر به این درختان را خواهیم شمرد که با زیر دنباله نمونه ۱ ۴ آغاز می‌شوند. طول این دنباله‌ها ۷ می‌باشد که دو عنصر اولیه آن ۴ و ۱ است. یعنی $\underbrace{۴\ ۱\ -\ -\ -\ -\ -}_{۷}$. این شمارش در چهار گام انجام می‌شود.

در **گام اول**، برگ‌های باقیمانده دنباله تقسیم بندی می‌شود. عنصر $E_۲ = ۱$ است. این بدین معناست که زیر درختی وجود دارد که تنها یک برگ دارد. و در مورد ۳ برگ باقیمانده اطلاعی در دست نیست. بنابراین باقیمانده دنباله بر اساس برگ‌های آن حداقل به دو قسمت تقسیم شود که در قسمت اول یک برگ و در قسمت بعد سه برگ جای بگیرد.

در حالت کلی اگر زیر دنباله داده شده $(E_۱\ E_۲\ \cdots\ E_k)$ باشد، برگ‌ها را می‌توان دقیقا به k قسمت تقسیم کرد که تعداد برگ‌ها در بخش $k-i+۱$اُم برابر $E_i - \sum_{j|P_j=i} E_j$ است. (در این رابطه، P_j اندیس پدر عنصر j است و مقدار E_j اگر صفر باشد، یک در نظر گرفته

می‌شود.) قسمت‌هایی با تعداد برگ صفر حذف می شوند و مجموعه‌ایی به دست می آید که W نامیده می‌شود.

برای مثال، زیردنباله ۳ ۳ ۰ ۵ چهار صفر باقیمانده را به ترتیب به قسمت‌هایی با تعداد یک و سه برگ تقسیم می‌کند؛ چرا که از چهار قسمت تشکیل شده، قسمت اول سه برگ، قسمت بعد ۰ = ۳ − ۳ برگ، قسمت سوم صفر برگ و قسمت آخر ۱ = ۳ − ۱ − ۵ برگ خواهد داشت. بنابراین، برای این زیردنباله $W = \{۱, ۳\}$ خواهد بود.

در **گام دوم**، چیدمان‌های ممکن برگ‌ها از روی اعضای مجموعه W تولید می‌شود و مجموعه‌ایی بدست می آید که در نهایت U نامیده می‌شود.

برای مثال، اگر یکی از اعضای W، عدد ۳ باشد، یعنی سه برگ در یک زیر درخت وجود دارد، این برگ‌ها می‌توانند هر سه، پدر یکسان داشته باشند؛ یا دو برگ، برادر باشند و برگ دیگر در همان زیردرخت، پدر دیگری داشته باشد؛ یا هر سه، فرزند گره‌هایی مجزا باشند. این حالات، پارتیشن‌های عدد سه را می‌سازد. به عبارت دیگر، سه برگ را می‌توان به صورت اعضای $U_۳ = \{۳,\ ۱ + ۲,\ ۱ + ۱ + ۱\}$ در نظر گرفت.

تبدیل عدد سه به ۱ + ۲ به این معناست که ابتدا یک تک‌برگ، سپس آن دو برگ در ادامه دنباله خواهد بود؛ بنابراین، می‌بایست حالت ۲ + ۱ را نیز لحاظ کرد. به عبارت دیگر، باید جایگشت‌های هر یک از پارتیشن‌های تولید شده را هم در نظر گرفت. از این رو، مجموعه فوق به شکل $U^p_۳ = \{۳,\ ۱ + ۲,\ ۲ + ۱,\ ۱ + ۱ + ۱\}$ تغییر می‌یابد.

برای مثال مورد بررسی باید جایگشت پارتیشن‌های ۱ و ۳ را تولید و به هم پیوند داد. جایگشت‌های پارتیشن‌های عدد یک به سادگی مجموعه $U^p_۱\{۱\}$ خواهد بود. در نتیجه با اضافه کردن آن به ابتدای مجموعه قبل، مجموعه نهایی به صورت زیر خواهد بود:

$$U^p = U^p_{۱} \cdot U^p_{۳} = \{۱ + ۳,\ \ ۱ + ۱ + ۲,\ \ ۱ + ۲ + ۱,\ \ ۱ + ۱ + ۱ + ۱\}.$$

در **گام سوم**، باید این تقسیم‌های برگ‌ها را در ۵ جایگاه باقی‌مانده قرار داد. بدیهی است که ابتدا باید عدد ۵ را پارتیشن نمود و جایگشت‌های آن پارتیشن‌ها را بدست آورد و جایگشت‌های مناسب را انتخاب نمود. مجموعه جایگشت‌های مناسب فضای باقی‌مانده V^p نامیده می‌شود.

جایگشتی از پارتیشن‌های ۵ مناسب است که اولا هم طول یکی از پارتیشن‌های U^p باشد و ثانیا، هر عنصر آن از عنصر پارتیشن مورد نظر از U^p کوچکتر نباشد. برای مثال ۱ + ۱ + ۳ مناسب نیست؛ زیرا با این که هم طول ۲ + ۱ + ۱ و ۱ + ۲ + ۱ است، هر قسمت آن، از هر قسمت دو دنباله بزرگتر یا مساوی نیست. همچنین، ۱ + ۱ + ۱ + ۱ + ۱ مناسب نیست؛ چرا که پارتیشنی با اندازه ۵ در مجموعه U^p وجود ندارد. جدول ۴ـ۲ اعضای U^p و پارتیشن‌های مناسب ۵ را نشان می‌دهد.

در **گام چهارم**، به ازای هر جفت پارتیشن متناظر از دو مجموعه U^p و V^p تعداد زیر درختان ممکن شمارش شده و جمع می‌شوند.

برای مثال، انتخاب ۴ + ۱ از مجموعه V^p و ۳ + ۱ از مجموعه U^p به این معناست که پنج جای باقیمانده به دو بخش با اندازه های ۱ و ۴ تقسیم شود؛ که در قسمت اول یک برگ و در قسمت دوم سه برگ باشد. این حالت فقط به صورت $\underbrace{\circ\ \ \circ\ \ \circ\ \ ۳\ \mid\ \circ}_{۵}$ امکان پذیر است. در ضمن، مشهود است که داشتن ۳ برگ در ۴ جایگاه برابر با تعداد درختان $\mathcal{T}_{۱,۳}$ است و یک برگ در یک جایگاه حالت خاص یک درخت با یک گره می‌باشد.

در حالت کلی، اگر $v_۱ + v_۲ + \ldots + v_s$ و $u_۱ + u_۲ + \ldots + u_s$ یک زوج متناظر از عناصر

جدول ۴_۲: اعضای V^p به همراه U^p های متناظر

اعضای مناسب از V^p	U^p
$۱+۴$ $۲+۳$	$۱+۳$
$۱+۲+۲$ $۱+۳+۱$ $۲+۲+۱$	$۱+۲+۱$
$۱+۱+۳$ $۱+۲+۲$ $۲+۱+۲$	$۱+۱+۲$
$۱+۱+۱+۲$ $۱+۱+۲+۱$ $۱+۲+۱+۱$ $۲+۱+۱+۱$	$۱+۱+۱+۱$

V^p و U^p باشند که اندازه آن‌ها s است، با بکارگیری رابطه (۱) می‌توان تعداد درخت‌های ممکن را برای این ترکیب به صورت زیر به دست آورد:

$$\text{Trees for}\left(\sum_{r=۱}^{s} v_r,\ \sum_{r=۱}^{s} u_r\right) = \prod_{r=۱}^{s} |\mathcal{T}_{v_r-u_r,u_r}| = \prod_{r=۱}^{s} \frac{۱}{v_r-۱}\binom{v_r-۱}{u_r}\binom{v_r-۱}{u_r-۱}.$$

(مقادیر $|\mathcal{T}_{n,m}|$ برای $m=۱$، برابر یک و برای $n \leqslant ۰$، همواره صفر در نظر گرفته می‌شود.)

سرانجام باید همه حالت‌های ممکن برای هر جفت زوج متناظر از مجموعه‌های U^p و V^p شمارش نموده و با هم جمع نمود. به عبارت دیگر:

$$\texttt{CSW}(E_۱, E_۲, \cdots, E_k) = \sum_{V^p \Leftrightarrow U^p}\left(\prod_{r=۱}^{s} |\mathcal{T}_{v_r-u_r,u_r}|\right).$$

۲.۳.۴ محاسبه و ذخیره اطلاعات تابع CSW()

در این بخش رویکرد دیگری برای محاسبه تابع `CSW()` ارایه می‌شود که نتیجه آن تولید یک ساختار اطلاعاتی برای ذخیره اطلاعات تابع `CSW()` است. این ساختار اطلاعاتی می‌تواند یک ساختمان داده‌ایی ساده مانند یک جدول دوستونی یا یک trie باشد. انتخاب مناسب این ساختمان داده، تاثیر بسزایی در بازیابی اطلاعات از آن دارد.

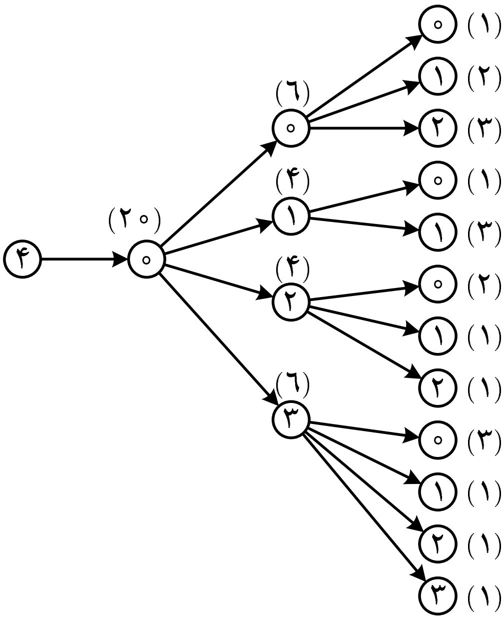
بهتر است ابتدا نوع اطلاعات تابع `CSW()` مورد بررسی قرار گیرد: این تابع برای یک مجموعه از درختان مشخص $\mathcal{T}_{n,m}$ تعین می‌کند که چه تعداد از این درختان با یک زیردنباله آغازین شروع می‌شوند. برای مثال، اگر درختان $\mathcal{T}_{۳,۴}$ در نظر گرفته شوند، در آن صورت:

$$\mathtt{CSW}(۴\ ۰) = ۲۰,$$

$$\mathtt{CSW}(۴\ ۰\ ۰) = ۶,\quad \mathtt{CSW}(۴\ ۰\ ۱) = ۴,\quad \mathtt{CSW}(۴\ ۰\ ۲) = ۴,\quad \mathtt{CSW}(۴\ ۰\ ۳) = ۶,$$

$$\mathtt{CSW}(۴\ ۰\ ۰\ ۰) = ۱,\quad \mathtt{CSW}(۴\ ۰\ ۰\ ۱) = ۲,\quad \mathtt{CSW}(۴\ ۰\ ۰\ ۲) = ۳,\quad \cdots$$

چنانچه مشهود است حجم بالایی از این اطلاعات دارای هم‌پوشانی می‌باشند. برای مثال، دنباله‌هایی که با ۴ ۰ شروع می‌شوند، مجموع دنباله‌هایی هستند که با ۰ ۰ ۴، ۱ ۰ ۴، ۲ ۰ ۴ و ۳ ۰ ۴ آغاز می‌گردند. برای سادگی بیشتر، می‌توان این اطلاعات را به صورت درختی نیز مشاهده و بررسی نمود. درخت شکل ۴−۱ این اطلاعات را برای قسمتی از درختان مجموعه $\mathcal{T}_{۳,۴}$ که با ۴ ۰ شروع می‌شوند، نشان می‌دهد.



شکل ۴—۱: نمایش درختی اطلاعات CSW برای درختان $\mathcal{T}_{3,4}$ و زیردنباله‌های ۴ ۰

عمق این درخت برابر $n+m-3$ می‌باشد و تعداد گره‌های آن از $O((n+m)|\mathcal{T}_{n,m}|)$ می‌باشد. مدیریت یک ساختار اطلاعاتی درختی پیچیده‌تر از یک جدول دوستونی است، ولی حجم ذخیره سازی و سرعت جستجو در این ساختار بسیار بیشتر است.

الگوریتم ۴—۵ در تولید اطلاعات CSW مستقل از ساختار داده‌ایی ذخیره آن عمل می‌کند. این الگوریتم، به صورت بازگشتی دنباله‌های مجاز آغازین و تعداد تکرار آن‌ها را مانند شکل ۴—۱ تولید می‌کند. با فراخواندن دستور `Store`، این مقادیر علاوه بر ذخیره شدن در ساختار داده‌ایی، بازگردانده می‌شوند.

در این الگوریتم، E یک زیردنباله آغازین است که در یک E-Sequence نگهداری می‌شود و متغیر i طول این زیردنباله را معین می‌کند. آرایه‌های Z و P متغیرهایی هستند که پیشتر در بخش ۲.۴ شرح داده شده‌اند. عنصر j اُم آرایه Z نمایانگر تعداد صفرهای زیردنباله از

function $ProduceCSW(E$: **ESeq**; Z,P:**array**; i,n,m: **integer**);
begin
  if $Z_i = 0$ **then** `Store(`E, i, 0`)`;
  if $Z_i = 1$ **then** $\mu := 1$; **else** $\mu := 0$;
  $E_{i+1} := \mu$;
  $P_{i+1} := \max\{j \leqslant i | E_j - E_{i+1} \geqslant Z_j - Z_{i+1} + 1 - \mu\}$;
  $\Theta := E_{P_{i+1}} - (Z_{P_{i+1}} - Z_{i+1})$;
  if $n + m - i - 1 \leqslant Z_i$ **then** $\Theta := 0$;
  if $i \geqslant n + m - 3$ **then** `Store(`E, i, $\Theta - \mu + 1$`)`;
  else
    for $E_{i+1} := \mu$ **to** Θ **do**
      if $E_{i+1} = 0$ **then** $Z_{i+1} = Z_i - 1$; **else** $Z_{i+1} := Z_i$;
      $s := s +$`ProduceCSW(`E, Z, P, $i+1$, n, m`)`;
    end;
    `Store(`E, i, s`)`;
  end;
end;

الگوریتم ۴—۵: الگوریتم تولید مقادیر تابع CSW

موقعیت $j+1$اُم تا انتهای دنباله می‌باشد. این مقدار به سادگی برابر m منهای تعداد صفرها از ابتدای دنباله تا موقعیت jاُم می‌باشد. همچنین، P_j برابر اندیس پدر گره jاُم است. در این الگوریتم، این دو آرایه با ساخته شدن هر زیردنباله، براساس تعاریفشان تکمیل می‌شوند.

برای آغاز محاسبه مقادیر تابع CSW برای مجموعه درختان $\mathcal{T}_{n,m}$ کافی است الگوریتم ۴—۵ با مقادیر اولیه زیر فراخوانی شود:

$$\texttt{ProduceCSW}([m, \cdots],\ [m, \cdots],\ [\lambda, \cdots],\ 1,\ n,\ m).$$

برای تولید درخت شکل ۴—۱ باید از دستور زیر استفاده کرد:

$$\texttt{ProduceCSW}([4, 0, \cdots],\ [4, 3, \cdots],\ [\lambda, 1, \cdots],\ 2,\ 3,\ 4).$$

الگوریتم ۴—۵ برای تولید اطلاعات تابع `CSW` از یک دنباله آغازین شروع می‌کند. سپس، بازه معتبر برای عنصر بعدی را، $[\mu, \Theta]$، محاسبه می‌کند. آنگاه به صورت بازگشتی ProduceCSW را، برای زیردنباله‌ایی با طول یکی بیشتر، صدا می‌زند.

این الگوریتم در هر بار فراخوانی گره‌ایی از درخت اطلاعات تابع `CSW` را ایجاد می‌کند. حجم گره‌های این درخت از $O((n+m)|\mathcal{T}_{n,m}|)$ می‌باشد. از این رو، پیچیدگی زمانی الگوریتم ۴—۵ برای ساختن آن نیز از $O((n+m)|\mathcal{T}_{n,m}|)$ می‌باشد. تولید اطلاعات مورد نیاز هر زیردنباله، برای رتبه‌گشایی یا رتبه‌گذاری، به طور متوسط دارای پیچیدگی زمانی $O(n+m)$ می‌باشد.

تنها نکته باقیمانده انتخاب نوع ساختمان داده‌ایی مناسب برای بازیابی اطلاعات تابع `CSW` است. در صورت استفاده از یک جدول دوستونی مرتب شده، پیچیدگی زمانی این جستجو برابر $O(log((n+m)|\mathcal{T}_{n,m}|))$ خواهد بود. استفاده از ساختمان داده پیچیده‌تری مانند یک trie این زمان را به $O(n+m)$ کاهش خواهد داد. در این صورت، پیچیدگی زمانی الگوریتم‌های رتبه‌گذاری و رتبه‌گشایی به صورت $O(nm(n+m))$ خواهد بود.

واژه‌نامه فارسی به انگلیسی

ارتفاع .... height
اشیا ترکیبیاتی .... combinatorial objects
الگوریتم .... algorithm
بدون‌دور .... assyclic
برادر .... sibling
برگ .... leaf
پدر .... parent
پیش‌ترتیب .... preorder
پس‌ترتیب .... postorder
پیمایش .... traverse
پیمایش معکوس .... reverse traverse
تابع پیچیدگی .... complexity function
ترتیب .... order

تولید .... generation

جد .... ancestor

جد مناسب .... proper ancestor

جنگل .... forest

حلقه .... self-loop

خارج شده .... incident from

درجه .... degree

درخت آزاد .... free tree

درخت .... tree

درخت دودویی .... binary tree

درخت دودویی منظم .... regular binary tree

درخت ریشه دار .... rooted/oriented tree

درخت مرتب .... ordered tree

درخت k-تایی .... k-array tree

دنباله .... Sequence

دور .... Cycle

رأس .... vertice

رتبه .... Rank

رتبه‌گذاری .... Ranking

رتبه‌گشایی .... Unranking

ریشه .... Root

زمان اجرا .... execution time

زیردرخت .... subtree

ساختار ترکیبیاتی .... combinatorial structure

ساختار داده‌ای .... Data Structure

علوم کامپیوتر .... Computer Science

عمق .... depth

فرزند .... child

قاموسی .... Lexicographic

قدم زدن بر درخت .... walk the tree

قویا همبند .... strongly connected

گراف جهت‌دار .... directed graph

گره .... node

گره خارجی .... externel node

گره داخلی .... internal node

گری کد .... Gray Code

لغت‌نامه‌ایی .... Lexicographic

لیست .... List

مبدا .... source

متصل .... connect

مجاور .... adjacent

محلی .... Local

مسیر .... path

مشاهده .... visit

مقصد .... target

موقعیت .... Position

میان‌ترتیب .... inorder

میزان حافظه .... memory cost

نواده .... descendent

نواده مناسب .... proper descendant

وارد .... inceident to

وزن .... Weight

همسایه .... neighbour

همبند .... connected

یال .... edge

۵

واژه‌نامه انگلیسی به فارسی

- مجاور ... adjacent
- الگوریتم ... algorithm
- جد ... ancestor
- بدون‌دور ... assyclic
- درخت دودویی ... binary tree
- فرزند ... child
- اشیا ترکیبیاتی ... combinatorial objects
- ساختار ترکیبیاتی ... combinatorial structure
- تابع پیچیدگی ... complexity function
- علوم کامپیوتر ... Computer Science

متصل ... connect

همبند ... connected

دور ... Cycle

ساختار داده‌ای ... Data Structure

درجه ... degree

عمق ... depth

نواده ... descendent

گراف جهت‌دار ... directed graph

یال ... edge

زمان اجرا ... execution time

گره خارجی ... externel node

جنگل ... forest

درخت آزاد ... free tree

تولید ... generation

گری کد ... Gray Code

ارتفاع ... height

خارج شده ... incident from

وارد شده به ... inceident to

میان‌ترتیب ... inorder

گره داخلی ... internal node

درخت k-تایی ... k-array tree

برگ .... leaf

قاموسی .... Lexicographic

لغت‌نامه‌ایی .... Lexicographic

لیست .... List

محلی .... Local

میزان حافظه .... memory cost

همسایه .... neighbour

گره .... node

ترتیب .... order

درخت مرتب .... ordered tree

پدر .... parent

مسیر .... path

موقعیت .... Position

پس‌ترتیب .... postorder

پیش‌ترتیب .... preorder

جد مناسب .... proper ancestor

نواده مناسب .... proper descendant

رتبه .... Rank

رتبه‌گذاری .... Ranking

درخت دودویی منظم .... regular binary tree

پیمایش معکوس .... reverse traverse

ریشه .... Root

درخت ریشه دار .... rooted/oriented tree

حلقه .... self-loop

دنباله .... Sequence

برادر .... sibling

مبدا .... source

قویا همبند .... strongly connected

زیردرخت .... subtree

مقصد .... target

پیمایش .... traverse

درخت .... tree

رتبه‌گشایی .... Unranking

رأس .... vertice

مشاهده .... visit

قدم زدن بر درخت .... walk the tree

وزن .... Weight

منابع و ماخذ

[۱] H. Ahrabian and A. Nowzari-Dalini, *Generation of t-ary Trees with Ballot-Sequences*, ————, x-x

[۲] H. Ahrabian and A. Nowzari-Dalini, *On the Generation of Binary Trees in A-order* , Intern. J. Computer Math. 00 (), 1-7

[۳] H. Ahrabian and A. Nowzari-Dalini, *Parallel Generation of t-ary Trees* , —— , X-X

[۴] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction To Algorithms*, The MIT Press, 2001.

[۵] M. Gaetler, *Clustering with spectral methods*, Ms thesis, Konstanz University, 2002.

[٦] U. Gupta, D.T.Lee and C.K.Wong, *Ranking and Unranking of 2-3 Trees*, SIAM J. Comput. 11 No. 3 (1982), 582-590

[۷] U. Gupta, D.T.Lee and C.K. Wong, *Ranking and Unranking of B-Trees*, Journal of Algorithms 4 (1983), 51-60

[۸] D.E. Knuth, *Art of Computer Programming vol. 1*, Addison-Wesley, 1997.

[۹] J.F. Korsh, *A-Order generation of k-ary trees with a 4k-4 letter alphabet*, Journal of Information and Optimization Sciences, 16 (1995), 557-564.

[۱۰] D.L. Kreher and D.R. Stinson, *Combinatorial Algorithms. Generation, Enumeration and Search*, CRC Press, 1999.

[۱۱] Liwu Li, *Ranking and Unranking of AVL Trees*, SIAM J. Comput. 15 No. 4 (1986), 1025-1035

[۱۲] E. Makinen, *A Survey on Binary Tree Coding*, The Comput. J. 34 No. 5 (1991), 438-443

[۱۳] J. Pallo and R. Racca, *A Note on Generating Binary Trees in A-order and B-order*, Intern. J. Computer Math. 18 (1985), 27-39

[۱۴] J. Pallo, *Generating Trees with n Nodes and m Leaves*, Intern. J. Computer Math. 21 (1987), 133-144

[۱۵] J. Pallo, *Coding binary trees by words over a alphabet with four letters*, Journal of Information and Optimization Sciences, 13 (1992), 257-266.

[۱۶] D. Roelants Van Baronaigien, A loopless algorithm for generating binary trees sequences, *Inform. Process. Lett.* **39** (1991), 189–194.

[۱۷] F. Ruskey and T.C.Hu, *Generating Binary trees lexicographically*, SIAM J. Comput. 6 No. 4 (1977), 745-758

[۱۸] F. Ruskey, *Generating t-ary trees lexicographically*, SIAM J. Comput. 7 No. 4 (1978), 424-439

[۱۹] V. Vajnovszki and J. Pallo, *Ranking and Unranking k-ary trees with a 4k-4 letter alphabet*, Journal of Information and Optimization Sciences, 18 (1997), 271-279.

[۲۰] M.S. Waterman, *Introduction to Computational Biology: Maps, Sequences and genomes*, Chapman and Hall, 1995.

[۲۱] S. Zaks, *Lexicographic Generation of Ordered Trees*, Theoritical Computer Science 10, (1980), 63-82.

[۲۲] S. Zaks, *Generation k-ary Trees Lexicographically*, Theoritical Computer Science 10, (1980), 63-82.

[۲۳] الهه ادریسی، تولید کدهای متناظر با درختان دودویی، پایان نامه برای دریافت درجه کارشناسی ارشد دانشکده علوم دانشگاه تهران، بهمن ۱۳۸۰.

Abstract

The main concern of this thesis is trees with n internal nodes and m external nodes (leaves) denoted as $\mathcal{T}_{n,m}$. New algorithms for generation, ranking and unranking of these trees in A-order are introduced; So, a new integer sequence codeword, called E-sequence, is presented and shown that A-order over the set of $\mathcal{T}_{n,m}$ matches lexicographic order over the set of corresponding E-sequences.

One important application of trees with n nodes and m leaves is in generating secondary structures of RNAs with $2n+m-2$ nucleotides and $n-1$ basepairs.

Time complexity of generation algorithm is $O(n+m)$ whereas the only existing generation algorithm is of $O(nm)$. No other rank nor unrank algorithms are known in the literature.