جزیره کیش، خلیج فارس، ایران ۱۹ الی ۲۱ اسفند ۱۳۸۶
انجمن کامپیوتر ایران
Computer Society of Iran

پوستر

تولید مجموعه کدهای متناظر با درخت‌های k-تایی

هایده اهرابیان
دانشکده علوم کامپیوتر، دانشگاه تهران
Ahrabian@ut.ac.ir

حسن علیزاده قادیکلایی
دانشکده علوم کامپیوتر، دانشگاه تهران
Alizadeh55@gmail.com

چکیده: در این مقاله الگوریتم جدیدی برای تولید کدهای متناظر با درختهای k-تایی ارائه می‌شود که از رسته الگوریتم‌های برنامه ریزی پویا است. این الگوریتم تمام z-دنباله‌های متناظر با درختهای k-تایی با n گره داخلی را در ترتیب قاموسی B-ترتیب تولید می‌کند. ثابت می‌شود هر دنباله در زمان ثابت O(1) تولید می‌شود. ایده اصلی در این الگوریتم تولید کدهای متناظر با درختهای k-تایی با n گره، از روی کدهای متناظر با درختهای k-تایی با n-1 گره است که مبتنی بر دو عمل افزایش و الحاق است.

واژه های کلیدی: درخت k-تایی، z-دنباله، B-ترتیب، برنامه‌ریزی پویا، زمان ثابت

۱- مقدمه

درختها به ویژه درختهای k-تایی[1] از مهمترین ساختارها در علوم کامپیوتر به شمار می روند. بین اینگونه درختان و اشیایی چون درختهای دودویی[2]، درختهای مرتب[3]، عبارات پرانتزی خوش فرم و دنباله های بالت[4] روابط هم ارزی جالبی وجود دارد. تکنیکهای نمایش، تولید ، رتبه گذاری[5] و بازیابی از رتبه[6] برای چنین درختانی یکی از مسائل قابل توجه در هر دو جنبه تئوری و کاربردی است. تا به حال الگوریتم های مختلفی برای نمایش و تولید کدهای متناظر برای این درختها پیشنهاد شده است که هم درختهای دودویی و هم درختهای k-تایی را شامل می شود[5, 8, 9, 11, 12, 13, 14, 16]. هر یک از این الگوریتم های تولید، ترتیبی را بر روی درختها تعریف می‌کنند، و بر اساس این ترتیب، اقدام به تولید کدهای متناظر با این درختها در آن ترتیب می‌نمایند. دو ترتیب A-ترتیب[7] و B-ترتیب[8] از جمله معروفترین این ترتیب ها هستند[7, 9]. این دو ترتیب توسط زکس[15, 16] و پالو[9, 10] معرفی شدند. از روشهای کدگذاری رایج می توان به x-دنباله، z-دنباله، بالت-دنباله و p-دنباله اشاره کرد. برخی از این دنباله ها مانند z-دنباله[9] شامل اعداد صحیح و برخی مانند x-دنباله، شامل رشته‌های بیتی[10] هستند. بسیاری از این روشهای کد گذاری، کدهای متناظر با درختهای k-تایی را در ترتیب های A-ترتیب یا B-ترتیب تولید می‌کنند. اولین بار، زکس با معرفی دنباله های صحیح و بیتی تمام کدهای متناظر با تمام درختهای k-تایی با n گره داخلی را بصورت B-ترتیب تولید کرد[15, 16]. اهرابیان-نوذری الگوریتم دیگری ارائه دادند که درختها را در ترتیب عکس B-ترتیب تولید می‌نماید[1]. بارونجیان و راسکی الگوریتمی پیشنهاد دادند که کدهای متناظر با این درختها را توسط pew-دنباله و در ترتیب A-ترتیب تولید می کند[11, 12]. اهرابیان-نوذری همچنین الگوریتم دیگری ارائه کردند که تمام کدهای متناظر با درحتهای k-تایی با n گره داخلی را توسط دنباله‌ای موسوم به بالت-دنباله تولید می کند[2]. کورش[8] و اهرابیان-نوذری[3] الگوریتمی بر اساس کدهای گری[11] ابداع نمودند که کدهای متناظر را بر اساس ترتیب کد گری، یعنی کدهایی که هریک با کد قبلی فقط در یک بیت اختلاف دارند، تولید می‌کند.

در این مقاله الگوریتم جدیدی ارائه می شود که کدهای متناظر با درختهای k-تایی با n گره داخلی را بر اساس شیوه برنامه ریزی پویا[12] تولید می کند. این روش می تواند تمام کدهای متناظر را به شکل قاموسی در هر یک از ترتیبهای B-ترتیب و B-ترتیب معکوس تولید نماید. ایده اصلی در این الگوریتم این است که کدهای متناظر با درختهای k-تایی با n گره داخلی، از کدهای تولید شده در مرحله قبل برای درختهای k-تایی با n-1 گره داخلی به دست آید. این روش مبتنی بر رو عمل افزایش و الحاق است و می تواند به سادگی به هر دو صورت تکراری و بازگشتی پیاده سازی شود. در پایان، ثابت می شود هر دنباله در این روش در زمان O(1) تولید می شود که مستقل از k و n است.

در بخش بعد از این مقاله تعاریف و نمادهای مورد استفاده در این روش بیان می‌شود. سپس در بخش 3 خود الگوریتم ارائه می‌شود. در بخش 4، پیچیدگی زمانی الگوریتم با محاسبه اثبات می‌شود. در بخش 5 نیز خلاصه و نتیجه گیری آمده است.

1- تعاریف

فرض می کنیم خواننده با مفاهیم درخت k-تایی، درخت k-تایی توسعه یافته، گره داخلی، برگ و زیردرخت آشنایی دارد[6]. فرض کنید T یک درخت k-تایی با n گره داخلی باشد. هر گره داخلی از T دارای k یا 0 فرزند است. تعداد تمام درختهای k-تایی با n گره داخلی از رابطه معروف زیر به دست می آید.

$$C_{n,k}=\frac{1}{(k-1)n+1}\binom{kn}{n} \qquad (1)$$

به منظور کدگذاری درخت k-تایی T با n گره داخلی، به هر گره داخلی برچسب 1 و به هر برگ برچسب 0 نسبت می دهیم. سپس این درخت را بصورت پیش‌ترتیب پیمایش می کنیم، در این صورت دنباله ای با n تا 1 و $(k-1)n+1$ تا 0 به دست می آید. با توجه به این که همیشه آخرین گره ملاقات شده 0 است، در این دنباله آخرین 0 را حذف می کنیم و این دنباله را با $x=\{x_i\}_1^{kn}$ نشان می دهیم. از دنباله x دنباله دیگری به نام $z=\{z_i\}_1^n$ می سازیم که در آن z_i موقعیت i امین یک در دنباله x را نشان می دهد. دنباله x دارای خاصیت k-غالب[13] است، اگر در هر زیر دنباله $\{x_i\}_1^l$ $(1\le l\le kn)$ از آن تعداد 1 ها حداقل برابر $\lceil l/k \rceil$ باشد. واضح است که تعداد 1 ها در هر پیشوند از دنباله x حداقل برابر تعداد 0 ها است. خصوصیت k-غالب برای دنباله z بصورت $z_i\le (i-1)k+1$ در می آید، $i=1,2,\mathbf{K},n$.

قضیه زیر رابطه بین درختهای k-تایی و دنباله های صحیح را بیان می‌کند[15].

قضیه 1: مجموعه های زیر در تناظر یک به یک با هم هستند:
تمام درختهای k-تایی با n گره داخلی،
تمام دنباله های صحیح بصورت $\{z_i\}_1^n$ بطوریکه داشته باشیم :
$0<z_1<z_2<\mathbf{K}<z_n$ و $z_i\le (i-1)k+1$ برای
$i=1,2,\mathbf{K},n$.

نتیجه 1: برای هر z-دنباله $z=\{z_i\}_1^n$ داریم :
$i\le z_i\le (i-1)k+1$ ، $i=1,2,\mathbf{K},n$

برای طراحی الگوریتمی که بتواند کدهای متناظر با درختها را تولید کند، لازم است ترتیبی برای این درختها تعریف کنیم. یکی از این ترتیب ها B-ترتیب است که بصورت زیر تعریف می شود[15].

تعریف 1: اگر T و T' دو درخت k-تایی باشند، گوییم T<T' ، اگر و فقط اگر:
T تهی و T ناتهی باشد، یا
T ناتهی باشد، وبرای هر i دلخواه $(1\le i\le k)$ داشته باشیم،
$T_j=T'_j$ برای $j=1,2,\mathbf{K},i-1$ و $T_i<T'_i$.

قضیه زیر رابطه بین z-دنباله و درختهای k-تایی را نشان می-دهد[15].

قضیه 2: فرض کنید T و T' دو درخت k-تایی با n گره داخلی باشند، و z و z' دو z-دنباله باشند. روابط زیر با هم هم ارزند:
$T<T'$ (در B-ترتیب)
z<z'

قضیه 3: اگر $\{z_i\}_1^n$ یک z-دنباله برای یک درخت k-تایی با n گره داخلی باشد، آنگاه دنباله $\{w_i\}_1^{n+1}$ که بصورت زیر تعریف می‌شود، یک z-دنباله برای درخت k-تایی با n+1 گره داخلی است.

$$i\le n:\ w_i=z_i\ ,\ z_n+1\le w_{n+1}\le kn+1 \qquad (2)$$

اثبات: با توجه به قضیه 1، کافیست ثابت کنیم $\{w_i\}_1^{n+1}$ یک z-دنباله است. چون $\{z_i\}_1^n$ یک z-دنباله است، از قضیه 1 داریم: $0<z_1<z_2<\mathbf{K}<z_n$ و $z_i\le (i-1)k+1$ برای $i=1,2,\mathbf{K},n$. همچنین با در نظر گرفتن فرض قضیه می توان نوشت : $0<w_1<w_2<\mathbf{K}<w_n<w_{n+1}$. حال می توان نوشت:

$$\forall i<n+1:\ w_i\le k(n+1-1)+1=kn+1 \qquad (3)$$

همچنین برای $i=n+1$ با توجه به فرض قضیه داریم : $w_{n+1}\le kn+1$.

نتیجه 2: اگر $\{z_i\}_1^n$ یک z-دنباله برای یک درخت k-تایی با n گره داخلی باشد، آنگاه هر پیشوند به طول l از آن مانند $(l\le n)$ $\{z_i\}_1^l$، یک z-دنباله برای درخت k-تایی با l گره داخلی است.

۲- الگوریتم تولید

در این بخش یک الگوریتم جدید برای تولید تمام درختهای k-تایی با n گره داخلی ارئه می‌شود که بر اساس مشی برنامه ریزی پویا است. الگوریتم Iter_GenZ در شکل (1) تمام z-دنباله های متناظر با درختهای k-تایی با n گره داخلی را به شکل قاموسی در ترتیب B-ترتیب تولید می‌کند. در این روش مجموعه کدهای تمام درختهای k-تایی با n گره داخلی را از روی مجموعه کدهای تمام درختهای k-تایی با n-1 گره داخلی بدست می آوریم.

فرض کنید $S_{n,k}$ مجموعه تمام z-دنباله های متناظر با درختهای k-تایی با n گره داخلی باشد. اگر تعداد عناصر مجموعه $S_{n,k}$ را با $|S_{n,k}|$ نشان دهیم، واضح است که رابطه $|S_{n,k}|=C_{n,k}$ را می‌توان نوشت. الگوریتم با $S_{1,k}=\{z=1\}$ شروع به کار می‌کند، بطوریکه می توان نوشت: $|z|=1$. $S_{1,k}$ مجموعه کدهای درخت k-تایی با یک گره داخلی است. الگوریتم دارای n-1 مرحله است. در هر مرحله $S_{m,k}$ از روی $S_{m-1,k}$ در ترتیب قاموسی تولید می شود، $m=2,3,\mathbf{K},n$. شکل چگونگی کار این الگوریتم را نشان می دهد. در این مثال نحوه تولید تمام z-دنباله های درختهای 3-تایی با

چهار گره داخلی نشان شده است. این الگوریتم به کمک قضیه 3 تمام کدهای متناظر با درختهای k-تایی را به روش برنامه ریزی پویا تولید می کند. هر کد بر اساس دو عمل افزایش و الحاق به دست می آید. به وضوح مشخص است که این روش، کدها را در ترتیب عکس B-ترتیب تولید می نماید. به عبارت دیگر، $\{1,2,3,\mathbf{K},n\}$ اولین کد تولید شده در مجموعه $S_{n,k}$، و $\{1,1k+1,2k+1,\mathbf{K},(n-1)k+1\}$ آخرین کد تولید شده است. اگر دستورات for داخلی دوم و سوم را معکوس نمائیم، می‌توانیم کدها را در ترتیب B-ترتیب نیز تولید کنیم.

این الگوریتم به سادگی می تواند به صورت بازگشتی نیز نوشته شود. (شکل 2)

```
void  Iter_GenZ (int  n, int  k)
{
  S_{1,k} = {z = 1};  // |z| = 1
  for (m = 2; m ≤ n; m++) {
    S_{m,k} = f;
    for each z in S_{m-1,k} {
      w = z;   // |w| = m
      for (i = 1 + z_{m-1} ; i ≤ k(m-1)+1 ; i++) {
        w_m = i;
        S_{m,k} = S_{m,k} + w ; // Add w to S_{m,k}
      }
      S_{m-1,k} = S_{m-1,k} − z;   // Remove z from S_{m-1,k}
    }
  }
}
```

شکل (1) : الگوریتم تکراری تولید

3- آنالیز الگوریتم

لم 1 : برای $k \ge 2$, $n \ge 1$ داریم :

$$\sum_{z \in S_{n,k}} z_n = (kn+1)C_{n,k} - C_{n+1,k} \qquad (4)$$

قضیه 4 : الگوریتم Iter_GenZ تعداد $C_{n,k}$ دنباله متمایز به طول n تولید می کند.

اثبات : الگوریتم با مجموعه تک عضوی $S_{1,k} = \{z = 1\}$ شروع به کار می کند، بطوریکه $|z| = 1$ است. در هرمرحله، مجموعه $S_{m,k}$ از روی مجموعه $S_{m-1,k}$ ساخته می شود، $m = 2,3,\mathbf{K},n$. به کمک استقرا ثابت می کنیم که دنباله ها در ترتیب قاموسی تولید می-شوند.

با توجه به اینکه در این الگوریتم تمام دنباله های مجموعه $S_{m-1,k}$

```
void  GenZ (int  n, int  k)
{
  if (n = 1) {
    S_{1,k} = {z = 1};  // |z| = 1
    return ;
  }
  GenZ (n − 1, k) ; // Generate S_{n-1,k}
  S_{n,k} = f ;
  for each z in S_{n-1,k} {
    w = z;   // |w| = n
    for (i = 1 + z_{n-1} ; i ≤ k(n-1)+1 ; i++) {
      w_n = i;
      S_{n,k} = S_{n,k} + w ; // Add w to S_{n,k}
    }
    S_{n-1,k} = S_{n-1,k} − z;   // Remove z from S_{n-1,k}
  }
}
```

متمایز هستند و مجموعه $S_{m,k}$ نیز به کمک دو عمل افزایش و الحاق از هر دنباله در $S_{m-1,k}$ ساخته می‌شود، بنابراین تمام کدهای تولید شده در این الگوریتم متنایز هستند. اکنون باید ثابت کنیم تعداد

شکل (2) : نسخه بازگشتی الگوریتم

دنباله‌های تولید شده در این روش برابر $C_{n,k}$ است. با توجه به سه حلقه for تودر تو در این الگوریتم، حلقه دوم به تعداد $C_{m-1,k}$ بار انجام می شود. همچنین حلقه سوم از $1+z_{m-1}$ تا $k(m-1)+1$ اجرا می شود. بنابراین تعداد کل تکرارها و در نتیجه تعداد کل دنباله های تولید شده برابر است با :

$$\sum_{z\in S_{m-1,k}}(km-k+1-z_{m-1}-1+1)$$
$$=\sum_{z\in S_{m-1,k}}(km-k-z_{m-1}+1)$$
$$=\sum_{z\in S_{m-1,k}}(km-k+1)-\sum_{z\in S_{m-1,k}}z_{m-1}$$
$$=(km-k+1)C_{m-1,k}-((km-k+1)C_{m-1,k}-C_{m,k})$$
$$=C_{m,k}$$

با توجه به لم 1 و در نظر داشتن اینکه در انتها m=n است، نتیجه می‌شود، تعداد کل دنباله‌های تولید شده در این روش برابر $C_{n,k}$ است و اثبات پایان می‌یابد.

لم 2 :برای $k\ge 2$, $n\ge 1$، داریم:

$$C_{n+1,k}\ge 2C_{n,k} \qquad (5)$$

اثبات : از تعریف $C_{n,k}$، می توان نوشت:

$$C_{n+1,k}=\frac{1}{(k-1)(n+1)+1}\binom{kn+k}{n+1}$$
$$=\frac{1}{(k-1)(n+1)+1}\ \frac{(kn+k)!}{(n+1)!\ (kn+k-n-1)!}$$
$$C_{n,k}=\frac{1}{(k-1)n+1}\binom{kn}{n}=\frac{1}{(k-1)n+1}\ \frac{(kn)!}{n!(kn-n)!}$$
$$\frac{C_{n+1,k}}{C_{n,k}}=\frac{(kn+k-1)(kn+k-2)\ \mathbf{K}\ (kn+k-n+1)}{n(kn-1)\ \mathbf{K}\ (kn-n+2)}$$
$$=(k+\underbrace{\frac{k-1}{n}}_{\ge 2})(1+\underbrace{\frac{k-1}{kn-1}}_{\ge 1})\ \mathbf{K}\ (1+\underbrace{\frac{k-1}{kn-n+2}}_{\ge 1}),\ (k\ge 2)$$
$$\Rightarrow\quad \frac{C_{n+1,k}}{C_{n,k}}\ge 2\quad or\quad C_{n+1,k}\ge 2C_{n,k}$$

لم 3 : برای $k\ge 2$, $n\ge 2$، داریم :

$$\sum_{i=1}^{n-1}C_{i,k}\le C_{n,k} \qquad (6)$$

اثبات : رابطه برای n=2 برقرار است. به کمک استقرا فرض کنید، رابطه برای n=m برقرار باشد، ثابت می‌کنیم برای n=m+1 نیز صحیح است.

$$\sum_{i=1}^{m-1}C_{i,k}\le C_{m,k} \qquad (7)$$

باید نشان دهیم :

$$\sum_{i=1}^{m}C_{i,k}\le C_{m+1,k} \qquad (8)$$

با افزودن $C_{m,k}$ به دو طرف رابطه (7)، می‌توان نوشت :

$$C_{m,k}+\sum_{i=1}^{m-1}C_{i,k}\le C_{m,k}+C_{m,k} \qquad (9)$$

$$\sum_{i=1}^{m}C_{i,k}\le 2C_{m,k}\le C_{m+1,k} \qquad (10)$$

با توجه به لم 2، قضیه اثبات می شود.

قضیه 5 : الگوریتم Iter_GenZ هر دنباله را در زمان ثابت O(1) تولید می کند.

اثبات : از قضیه 4 می دانیم، الگوریتم تعداد $C_{n,k}$ دنباله متمایز به طول n تولید می‌کند. در این روش هر مجموعه $S_{m,k}$، از روی مجموعه $S_{m-1,k}$ تولید می‌شود، $m=2,3,\mathbf{K},n$. بنابراین تمام مجموعه‌های $S_{1,k},S_{2,k},\mathbf{K},S_{n,k}$ تولید می‌شوند. کل زمان لازم برای تولید تمام $C_{n,k}$ دنباله برابر است با :

$$T_{n,k}=C_{1,k}+C_{2,k}+\mathbf{K}+C_{n,k}=\sum_{m=1}^{n}C_{m,k} \qquad (11)$$

زمان لازم برای تولید یک دنباله برابر است با :

$$\frac{\sum_{m=1}^{n}C_{m,k}}{C_{n,k}}=\frac{C_{n,k}+\sum_{m=1}^{n-1}C_{m,k}}{C_{n,k}}=1+\frac{\sum_{m=1}^{n-1}C_{m,k}}{C_{n,k}} \qquad (12)$$

به کمک لم 3، می توان نوشت: $\frac{\sum_{m=1}^{n-1}C_{m,k}}{C_{n,k}}\le 1$. یعنی زمان تولید هر دنباله برابر O(1) است.

4- نتیجه گیری

در این مقاله، الگوریتم تکراری جدیدی برای تولید z-دنباله های متناظر با درختهای k-تایی با n گره ارائه شده است که دارای زمان O(1) به ازای هر دنباله است. نسخه بازگشتی الگوریتم نیز ارائه شده است. این الگوریتم، اولین الگوریتمی است که کدهای متناظر برای درختهای k-تایی را به روش برنامه ریزی پویا تولید می‌کند. این روش چند مزیت خوب دارد: یکی اینکه می تواند به هر دو شکل تکراری و بازگشتی پیاده سازی شود. دیگر اینکه می تواند کدها را در هر دو ترتیب مستقیم و معکوس B-ترتیب تولید نماید. همچنین الگوریتم مناسبی برای موازی سازی است. مهمترین عیب این روش این است که برخلاف الگوریتمهایی مانند الگوریتم زکس، نمی‌توان از یک دنباله، دنباله بعدی را به دست آورد. همچنین الگوریتمهای رتبه گذاری و بازیابی از رتبه در این روش نامشخص است.

پیاده سازی این روش ساده است. می توان برای این منظور از ساختار لیست پیوندی یکطرفه استفاده کرد. در هر گره از این لیست،

دو فیلد ذخیره می شود، یکی برای z-دنباله و دیگری اشاره گری برای نگهداری آدرس گره بعدی. با توجه به الگوریتم ارائه شده ، دو عمل درج و حذف در این لیست باید انجام شود که هر کدام در زمان ثابت $O(1)$ قابل انجام است. حافظه مورد استفاده برای پیاده سازی این روش، $O(C_{n,k})$ است.

مراجع

[1] H.Ahrabian and A.Nowzari-Dalini, Adaptive generation of t-ary trees in parallel, Adv. Modeling. Optim., **8,** (2006), 16-26.

[2] H.Ahrabian and A.Nowzari-Dalini, Generation of t-ary trees with Ballot sequences, Intern. J. Comput. Math., **80** (2003), 1243-1249.

[3] H.Ahrabian and A.Nowzari-Dalini, Gray code algorithm for listing k-ary trees, Studies in Informatics and Control, **13** (2004), 243-251.

[4] H.Ahrabian and A.Nowzari-Dalini, On the generation of P-sequences, Adv. Modeling. Optim., **5,** (2003), 27-38.

[5] M.C. Er, Efficience generation of k-ary trees in natural order, Comput. J., **35**, (1992), 306-308.

[6] D. E. Knuth, The art of computer programming, Vol. 1: Fundemental algorithms, 2nd. Ed., Addison-Wesley, Reading, 1973.

[7] J. F. Korsh, A-order generation of k-ary trees with 4k-4 letter alphabet, J. Inform. Optim. Sci., **16** (1995), 557-567.

[8] J. F. Korsh and P. Lafollette, Loopless generation of Gray code for k-ary trees , Inform. Process. Lett., **70** (1999), 7-11.

[9] J. Pallo and R. Racca, A note on Generating binary tree in A-order and B-order, Intern. J. Comput. Math., **18** (1985), 27-39.

[10] J. Pallo, Generating trees with n nodes and m leaves, Intern. J. Comput. Math., **21** (1987), 133-144.

[11] D. Roelants Van Baronaigien and F.Ruskey, Generating t-ary trees in A-order, Inform. Process. Lett., **27** (1988), 205-213.

[12] F. Ruskey, Generating t-ary trees lexicographically, SIAM J. Comput., **7** (1978), 424-439.

[13] E. Trojanowski, Ranking and listing algorithm for k-ary trees, SIAM J. Comput., **7**, (1978), 492-509.

[14] Z. Yongjin and W. Jianfang, Generating k-ary trees in lexicographical order, Sci. Sin., **23,** (1980), 1219-1225.

[15] S. Zaks, Lexicographic generation of ordered tree, Theoret. Comput. Sci., **10** (1980), 63-82.

[16] S. Zaks, Generating and ranking t-ary trees, Inform. Process. Lett., **14** (1982), 44-48.

[1] k-ary Trees
[2] Binary Trees
[3] Ordered Trees
[4] Ballot sequence
[5] Ranking
[6] Unranking
[7] A-order
[8] B-order
[9] Z-sequence
[10] Bit string
[11] Gray code
[12] Dynamic programming
[13] K-dominant