کراص الگوریتم

در الگوریتم Insertion Sort، اگر $\overline{t_k}$ متوسط تعداد مقایسه های لازم برای درج عنصر k ام در یک آرایه n عنصری باشد، $\overline{t_k}$ برابر کدام گزینه است؟

۱. n ۲. k ۳. $\frac{k+1}{2}$ ۴. $\frac{n+1}{2}$

زمان اجرای الگوریتم زیر کدام است؟

```
=1;
vhile(i<=n){
i=i×2;
```

۱. $T(n)\in\theta(\log n)$ ۲. $T(n)\in\theta(n^2)$ ۳. $T(n)\in\theta(n)$ ۴. $T(n)\in\theta(n\log n)$

از بین سه مورد داده شده کدام موارد صحیح است؟

مورد ۱: $n!\in O(n^n)$

مورد ۲: $\frac{n^2}{n\log n}\in\Omega(n^2)$

مورد ۳: $n^{2^n}+6\times 2^n\in\Omega(2^n)$

۱. مورد ۱ و مورد ۲ ۲. مورد ۱ ۳. مورد ۱ و مورد ۳ ۴. موارد ۱ و ۲ و ۳

حاصل f(5) با توجه به الگوریتم زیر کدام است؟

```
nt f(n){
 if(n==1)
   return 1;
 else
   return f(n-1)+2n;
}
```

۱. 29 ۲. 19 ۳. 13 ۴. 23

کراص الگوریتم

تابع زیر بر روی درخت دودویی T چه کاری انجام می دهد؟

```
int test(node* tree){
if(tree==null)
  return 0;
else
  return 1+max(test(tree → left),test(tree → right));
```

۱. محاسبه تعداد گره های داخلی درخت
۲. محاسبه تعداد برگ های درخت
۳. محاسبه عمق درخت
۴. محاسبه تعداد گره های دوفرزندی درخت

مرتبه زمانی رابطه بازگشتی $T(n)=T(\frac{n}{2})+n\log n$ **کدام است؟**

۱. $\theta(n)$
۲. $\theta(n\log n)$
۳. $\theta(n^2\log n)$
۴. $\theta(n\log\log n)$

جواب کلی رابطه بازگشتی زیر کدام است؟ (c1 و c2 دو عدد حقیقی هستند.)

$$T(n)=2T(n-1)+3T(n-2)$$

۱. $T(n)=C_1(1)^n+C_2(3)^n$
۲. $T(n)=C_1(2)^n+C_2(3)^n$
۳. $T(n)=C_1(-2)^n+C_2(-3)^n$
۴. $T(n)=C_1(-1)^n+C_2(3)^n$

اگر 10 عنصر در یک لیست از اندیس 1 تا 10 به صورت مرتب شده قرار گرفته باشند، با توجه به درخت تصمیم دودوئی برای جستجوی دودوئی، میانگین تعداد مقایسه ها در جستجوی ناموفق کدام است؟

۱. 3.21
۲. 3.54
۳. 3.78
۴. 3.93

با توجه به الگوریتم مرتب سازی سریع، نتیجه اجرای تابع partition بر روی آرایه زیر کدام است؟

12	1	25	3	28	47	10	8	52

۱.

8	1	3	10	12	47	25	28	52

۲.

3	1	10	8	12	47	25	28	52

۳.

1	3	8	10	12	25	28	47	52

۴.

8	1	3	10	12	28	25	47	52

کراص الگوریتم

تعداد ضرب های انجام شده توسط الگوریتم استراسن برای ضرب دو ماتریس 4×4 کدام است؟

۱. 196 ۲. 49 ۳. 343 ۴. 56

برای یافتن ماکسیمم و مینیمم عناصر یک آرایه با استفاده از روش تقسیم و حل، پس از تقسیم مساله به دو زیرمساله مساوی و یافتن کوچکترین و بزرگترین عنصر در هر زیر لیست، عناصر بدست آمده از زیرلیست ها را برای یافتن بزرگترین و کوچکترین عناصر نهایی با هم مقایسه می نماییم. تعداد مقایسه های انجام شده در این الگوریتم کدام گزینه است؟

۱. $2n-1$ ۲. $n-1$ ۳. $\frac{3n}{2}-2$ ۴. $2n$

اگر مجموعه سکه های موجود در مسأله خرد کردن پول به صورت $\{1و2و5و10و13و12\}$ باشد و از هر سکه به تعداد دلخواه موجود باشد. در الگوریتم حریصانه برای خردکردن 17 ریال کدام مجموعه از سکه ها انتخاب می شود؟

۱. {15و2} ۲. {12و5} ۳. {15و1و1} ۴. {10و5و2}

کدام گزینه در مورد الگوریتم های پریم و کروسکال صحیح است؟

۱. الگوریتم پریم همواره از الگوریتم کروسکال سریع تر است.

۲. الگوریتم کروسکال با انتخاب نزدیکترین گره در هر مرحله، درخت پوشای کمینه را پیدا می کند.

۳. الگوریتم کروسکال در بدترین حالت دارای پیچیدگی زمانی $\theta(n \log n)$ است. (n=تعداد رئوس)

۴. الگوریتم کروسکال در گراف متراکم سریع تر از الگوریتم پریم است.

در کدگذاری رشته abaabacadcade با استفاده از روش هافمن، کد حاصل برای هر کدام از نویسه ها کدام است؟

۱. a=1, b=01, c=001, d=0001, e=0000 ۲. a=0, b=101, c=110, d=111, e=100

۳. a=000, b=001, c=010, d=011, e=100 ۴. a=00, b=01, c=10, d=11, e=100

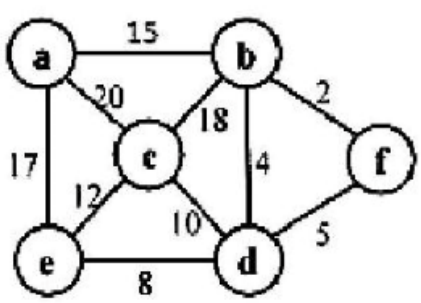
کراص الگوریتم

در مسأله کوله پشتی، اگر آیتم ها به صورت جدول زیر باشند و ظرفیت کوله پشتی 13 باشد، بیشترین ارزش به دست آمده به روش حریصانه کدام است؟

i	P_i	w_i
1	35	7
2	30	5
3	20	2
4	12	3
5	3	1

۱. 65 ۲. 68 ۳. 70 ۴. 80

برای یافتن درخت پوشای کمینه گراف زیر به کمک الگوریتم پریم، ترتیب انتخاب یال ها با شروع از رأس a کدام گزینه است؟(از چپ به راست)



۱. (b,f),(b,d),(e,d),(c,d),(a,b) ۲. (a,b),(b,f),(f,d),(d,e),(e,c)

۳. (a,b),(b,f),(b,d),(d,e),(d,c) ۴. (a,b),(b,f),(f,d),(d,e),(d,c)

در ضرب زنجیره ای ماتریس های $A_{10\times20}\times B_{20\times50}\times C_{50\times1}\times D_{1\times100}$، ترتیب پرانتز گذاری بهینه برای حداقل اعمال ضرب کدام است؟

۱. $((A\times(B\times C)\times D)$ ۲. $(((A\times B)\times C)\times D)$

۳. $(A((B\times C)\times D))$ ۴. $((A\times B)\times(C\times D))$

کراص الگوریتم

اگر یک مسئله هم به روش برنامه نویسی پویا و هم به روش تقسیم و حل قابل حل باشد، آنگاه کدام گزینه صحیح است

۱. استفاده از روش تقسیم و حل بهتر است چون پیاده سازی آن آسان است.

۲. استفاده از روش برنامه نویسی پویا بهتر است چون حافظه مصرفی آن کمتر است.

۳. روش برنامه نویسی پویا ممکن است نسبت به روش تقسیم و حل مسئله را در زمان کمتری حل کند.

۴. روش تقسیم و حل همواره نسبت به روش برنامه نویسی پویا مسئله را در زمان کمتری حل می کند.

یافتن بزرگترین زیر رشته مشترک دو رشته X و Y که هر کدام دارای طول n هستند، دارای چه مرتبه زمانی است؟

۱. $\theta(n)$ ۲. $\theta(n^2)$ ۳. $\theta(n^2 \log n)$ ۴. $\theta(n^3)$

میزان حافظه مصرفی در روش برنامه نویسی پویا برای مسئله فروشنده دوره گرد بازای n راس کدام است؟

۱. $\theta(n)$ ۲. $\theta(n^2)$ ۳. $\theta(n\,2^n)$ ۴. $\theta(n^2\,2^n)$

تعداد کل گره های درخت فضای حالت در روش عقبگرد برای حل مسئله حاصل جمع زیرمجموعه ها بازای n عدد صحیح کدام است؟

۱. 2^n ۲. $2^n - 1$ ۳. 2^{n-1} ۴. $2^{n+1} - 1$

در چند مورد از مسائل زیر،جواب های مساله در گره های موجود در پایین ترین سطح درخت فضای حالت قرار دارند؟

مورد 1: حاصل جمع زیر مجموعه ها

مورد 2: مدارهای هامیلتونی

مورد 3: n-وزیر

۱. 2 ۲. 3 ۳. 1 ۴. 0

از بین موارد زیر کدام مورد یا موارد صحیح است؟

مورد1: در روش شاخه و حد جستجوی درخت فضای حالت به صورت عمقی انجام می شود.

مورد2: روش شاخه و حد برای مسائل بهینه سازی مورد استفاده قرار می گیرد.

۱. فقط مورد 1 ۲. فقط مورد 2 ۳. مورد 1 و مورد 2 ۴. هیچکدام

کراص الگوریتم

اگردر مسئله حاصل جمع زیر مجموعه ها داشته باشیم $S=\{5,10,12,13,15,18\}$ و $W=30$، انگاه چند راه حل وجود دارد؟

۱. 2 ۲. 3 ۳. 4 ۴. 1

کدام یک از موارد زیر به طور قطع صحیح است؟

۱. $P \subseteq NP$ ۲. $P = NP$ ۳. $P \neq NP$ ۴. $NP \subseteq P$

سوالات تشریحی

کراص الگوریتم

مثال ۲ : با مثال نشان دهید که :

$$\begin{cases} a_n = 2a_{n-1} + 1 \\ a_1 = 1 \end{cases} \Rightarrow a_n = 2^n - 1$$

حل : از روی رابطه مستقیم $a_n = 2^n - 1$ داریم :

$$a_1 = 2^1 - 1 = 1 \quad , \quad a_2 = 2^2 - 1 = 3 \quad , \quad a_3 = 2^3 - 1 = 7 \quad , \quad a_4 = 2^4 - 1 = 15$$

از روی روابط بازگشتی داریم :

$$a_2 = 2a_1 + 1 = 2 \times 1 + 1 = 3 \quad , \quad a_3 = 2 \times a_2 + 1 = 2 \times 3 + 1 = 7$$
$$a_4 = 2a_3 + 1 = 2 \times 7 + 1 = 15$$

مثال ۳۱ : مرتبه اجرایی الگوریتم بازگشتی برج‌های هانوی را بدست آورید:

```
oid Hanoi (int n, A, B, C)

if (n==1) Move a disk from A to C;
else {
        Hanoi (n - 1, A, C, B);
        Move a disk from A to C;
        Hanoi (n-1, B, A, C);
}
```

روش دوم : اگر $n = 1$ باشد فقط یک انتقال صورت می‌گیرد. در غیر اینصورت علاوه بر یک انتقال تابع دو بار با مقدار $n - 1$ فراخوانی می‌گردد. لذا :

$$T(n) = \begin{cases} 1 & \text{اگر } n = 1 \\ T(n-1) + T(n-1) + 1 & \text{اگر } n > 1 \end{cases}$$

حال با روش جایگذاری معادله بازگشتی فوق را حل می‌کنیم :

$$T(n) = 2T(n-1) + 1 = 2(2T(n-2) + 1) + 1 = 2^2 T(n-2) + 2 + 1$$
$$= 2^3 T(n-3) + 2^2 + 2 + 1 = \ldots$$
$$= 2^{n-1} T(1) + 2^{n-2} + 2^{n-3} + \ldots + 2 + 1 \qquad \left(\frac{t_1(q^n - 1)}{q - 1}\right) \text{ جمع تصاعد هندسی}$$

$$\frac{1 \times (2^n - 1)}{2 - 1} = 2^n - 1 \quad \Rightarrow \quad T(n) = O(2^n)$$

مثال ۱۹ : رابطه $a_n = 3a_{n-1} + 5 \times 7^n$ برای مقادیر $n \geq 1$ و با شرط اولیه $a_0 = \frac{39}{4}$ را حل کنید.

حل :

$$a_n - 3a_{n-1} = 0 \Rightarrow r - 3 = 0 \Rightarrow r = 3$$

جواب عمومی : $U_n = \alpha_1 3^n$

با توجه به اینکه $f(n) = 5 \times 7^n$ می‌باشد، جواب خصوصی را به صورت $P_n = C7^n$ حدس می‌زنیم:

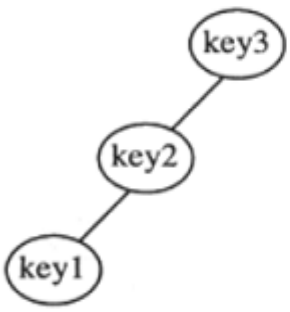
$$C7^n = 3C7^{n-1} + 5 \times 7^n \Rightarrow 7C - 3C = 5 \times 7 \Rightarrow C = \frac{35}{4}$$

$$a_n = (\alpha_1 3^n) + (\frac{35}{4} 7^n)$$

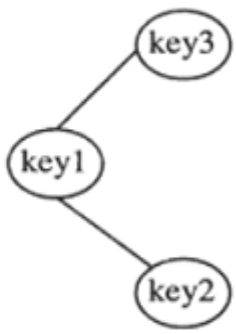
$$a_0 = \frac{39}{4} \Rightarrow \frac{39}{4} = \alpha_1 3^0 + \frac{35}{4} 7^0 \Rightarrow \frac{39}{4} = \alpha_1 + \frac{35}{4} \Rightarrow \alpha_1 = 1$$

$$\Rightarrow a_n = 3^n + \frac{5}{4} 7^{n+1}$$

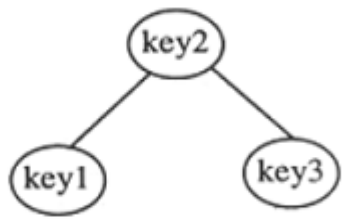
مثال : سه کلید Key1 < key2 < key3 با احتمال‌های $P_1 = 0.7$ ، $P_2 = 0.2$ و $P_3 = 0.1$ در ۵ درخت مختلف BST ذخیره شده‌اند. زمان جستجوی میانگین را برای هر یک از درخت‌ها محاسبه کنید :



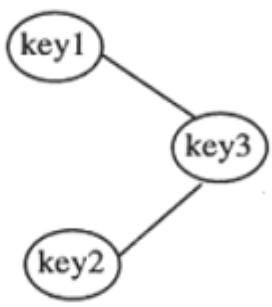
(درخت ۱)



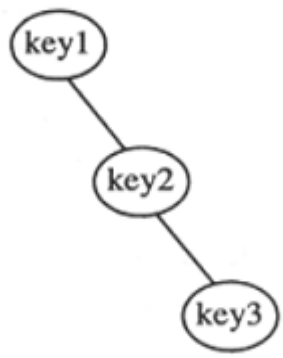
(درخت ۲)



(درخت ۳)



(درخت ٤)



(درخت ٥)

حل :

درخت ۱ $\Rightarrow 3 \times 0.7 + 2 \times 0.2 + 1 \times 0.1 = 2.6$

درخت ۲ $\Rightarrow 2 \times 0.7 + 3 \times 0.2 + 1 \times 0.1 = 2.1$

درخت ۳ $\Rightarrow 2 \times 0.7 + 1 \times 0.2 + 2 \times 0.1 = 1.8$

درخت ٤ $\Rightarrow 1 \times 0.7 + 3 \times 0.2 + 2 \times 0.1 = 1.5$

درخت ٥ $\Rightarrow 1 \times 0.7 + 2 \times 0.2 + 3 \times 0.1 = 1.4$

همان‌طور که مشاهده می‌شود درخت ۵، حداقل تعداد مقایسه را نیاز داشته و لذا از همه بهتر است. هدف ما این است که با توجه به احتمالات داده شده‌ی هر کلید شکل بهینه فوق را به دست آوریم. توجه کنید برای ۳ کلید فقط همان ۵ حالت شکل فوق امکان‌پذیر است.

یک راه ساده ولی بسیار کند مشابه فوق است یعنی تمامی حالات ممکن را ترسیم و زمان جستجوی میانگین را برای همه آنها محاسبه کنیم و در آخر حداقل آنها را انتخاب نمائیم. ولی تعداد این درخت‌ها مانند اکثر مسائل بهینه‌سازی حداقل از مرتبه نمایی است. برای اثبات فقط درخت‌هایی با عمق $n - 1$ را در نظر می‌گیریم که زیر مجموعه‌ای از کل حالات مورد نظر است یعنی هر گره فقط یک بچه داشته باشد. در این وضعیت گره فرزند

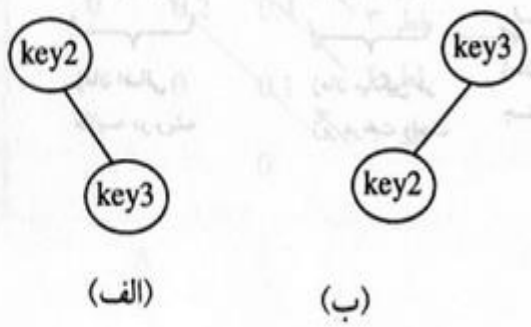
می‌تواند چپ یا سمت راست پدر خود قرار گیرد یعنی برای قرار دهی هر گره (به جز ریشه) 2 حالت انتخاب وجود دارد و لذا تعداد کل انتخاب‌ها 2^{n-1} می‌باشد یعنی تعداد درخت‌های جستجوی متفاوت با عمق n-1 برابر 2^{n-1} است که بسیار زیاد است. پس باید الگوریتمی کارآمد با برنامه‌نویسی پویا بنا کنیم.

تذکر : در کتاب هورویتز اثبات شده که با n کلید مجزا می‌توان $\frac{1}{n+1}\binom{2n}{2}$ درخت BST مجزا ساخت مثلاً برابر $n = 3$ به تعداد ۵ درخت BST می‌توان ساخت :

$$\frac{1}{3+1}\binom{6}{3}=$$ Personal_Office

این الگوریتم شباهت زیادی به الگوریتم ضرب زنجیری ماتریس‌ها دارد. اصل بهینگی برای این مسأله نیز صادق است. چرا که مثلاً در شکل قبلی که درخت شماره ۵ بهینه بود، زیر درخت مربوط به ریشه نیز می‌بایست بهینه باشد.

شکل‌های ممکن برای دو کلید key 2 < key 3 با احتمالات $P_3 = 0.1$ و $P_2 = 0.2$ به دو صورت زیر است:



(الف) ⇒ $1 \times 0.2 + 2 \times 0.1 = 0.4$

(ب) ⇒ $2 \times 0.2 + 1 \times 0.1 = 0.5$

(الف) (ب)

پس شکل (الف) شکل بهینه است.

چون اصل بهینگی در اینجا صادق است یک الگوریتم بازگشتی برای حل مسأله بنا می‌کنیم.

مشابه مسأله ضرب ماتریس‌ها از یک ماتریس A استفاده می‌کنیم که مفهوم عناصر آن به صورت زیر است:

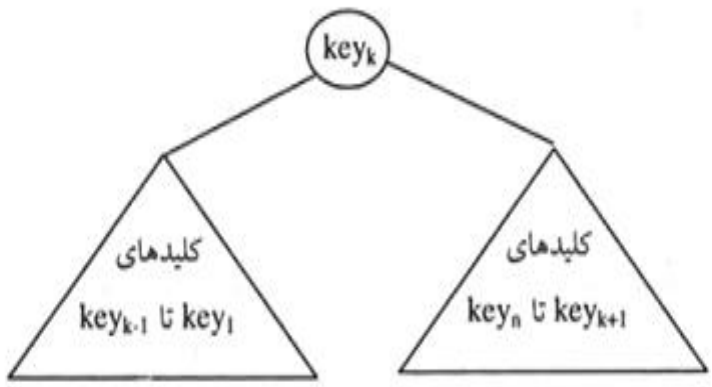
A[i] [j] =	مینیمم میانگین زمان جستجو برای یک درخت BST
$1 \le i \le j \le n$, $key_i \le key_j$	با کلیدهای key_i تا key_j که

بدیهی است که $A[i][i] = P_i$ است چرا که در این حالت درخت BST فقط یک کلید key_i به صورت روبه‌رو دارد :

که میانگین زمان جستجوی آن برابر است با :

$$A[i][i] = 1 \times P_i = P_i$$

بنابر اصل بهینگی اگر درخت بهینه مورد نظر شامل ریشه key_k باشد انگاه دو زیر درخت چپ و راست این ریشه نیز باید BST بهینه باشند :



اگر درخت فوق، درخت بهینه BST باشد آنگاه A[1][n] با فرض آنکه key_k ریشه باشد برابر خواهد بود با:

$$\underbrace{A[1][k-1]}_{\text{زمان میانگین در زیر درخت چپ}} + \underbrace{P_1 + \ldots + P_{k-1}}_{\text{زمان اضافی مقایسه در ریشه}} + \underbrace{P_k}_{\text{زمان میانگین جستجو برای ریشه}} + \underbrace{A[k+1][n]}_{\text{زمان میانگین در زیر درخت راست}} + \underbrace{P_{k+1} + \ldots P_n}_{\text{زمان اضافی مقایسه در ریشه}}$$

Personal_Office

$$= A[1][k-1] + A[k+1][n] + \sum_{m=1}^{n} P_m$$

توجه کنید فرمول فوق برای وقتی است که فرض کرده‌ایم key_k ریشه باشد ولی هر یک از key ها (key_1 تا key_n) می‌توانند در ریشه باشند که آنگاه جواب نهایی مینیمم آنها خواهد بود، لذا :

$$A[1][n] = \min_{1 \le k \le n} (A[1][k-1] + A[k+1][n]) + \sum_{m=1}^{n} P_m$$

در حالت کلی برای هر key_i تا key_j داریم :

$$A[i][j] = \min_{i \le k \le j} (A[i][k-1] + A[k+1][j]) + \sum_{m=i}^{j} P_m \qquad (i < j)$$

$$A[i][i] = P_i \qquad (i = j)$$

فرمول اصلی فوق برای ساخت ماتریس A استفاده می‌شود. جهت سادگی ماتریس A را به صورت $(n+1)\times(n+1)$ در نظر می‌گیریم که قطر اصلی آن صفر است. همگام با ساختن ماتریس A ، ماتریس کمکی R

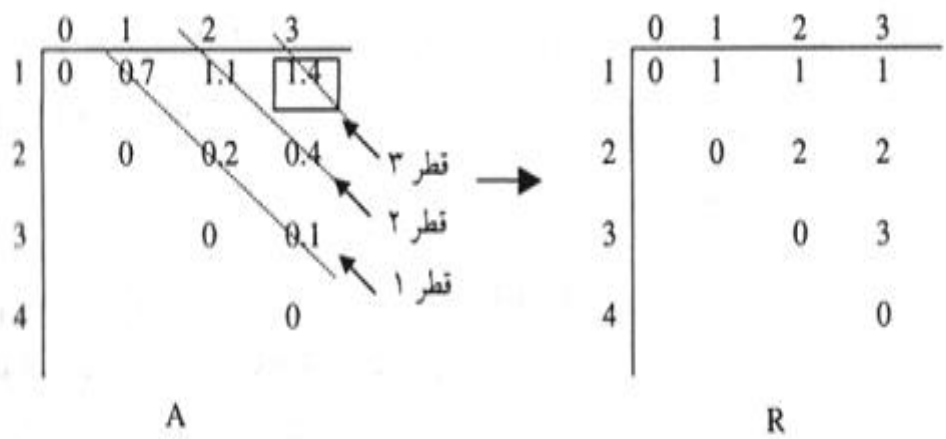
را نیز کنار آن پدید می‌آوریم. محتوای R[i][j] نمایانگر اندیس کلیدی است که به عنوان ریشه انتخاب می‌شود. مثلاً R[2][4] اندیس کلید ریشه درخت BST بهینه‌ای است که از کلیدهای key_2 ، key_3 و key_4 تشکیل شده است.

با یک مثال نحوه محاسبه ماتریس‌های A و R را نشان می‌دهیم. این ماتریس‌ها که $(n+1) \times (n+1)$ هستند اندیس سطرهایشان از 1 تا $n+1$ و اندیس ستون‌هایشان از 0 تا n است که n تعداد کلیدهاست. قطر اصلی هر دو ماتریس صفر است. این ماتریس‌ها را مشابه بحث ضرب ماتریس‌ها به صورت قطر به قطر پر می‌کنیم :

مثال : درخت BST بهینه مربوط به کلیدهای زیر را با احتمالات داده شده ترسیم کنید :

key_1	key_2	key_3
$P_1=0.7$	$P_2=0.2$	$P_3=0.1$

حل :



قطر ۱ : از آنجا که در قطر 1 داریم $A[i][i] = P_i$ به راحتی احتمالات داده شده را در قطر مربوطه قرار می‌دهیم. به همین ترتیب R[i][i]=i، می‌باشد چرا که در درخت BST بهینه از کلید key_i تا key_i بدیهی است که اندیس ریشه همان i است (چرا که این درخت، فقط یک گره دارد)

قطر ۲ :

$$A[1][2] = \min_{1\le k\le 2}(A[1][k-1]+A[k+1][2]) + \sum_{m=1}^{2} P_m$$

$$= \min(A[1][0]+A[2][2], A[1][1]+A[3][2]) + (P_1+P_2)$$

$$= \min(\underbrace{0+0.2}_{k=1}, \underbrace{0.7+0}_{k=2}) + (0.7+0.2) = 1.1$$

پس در خانه R[1][2] عدد $k = 1$ را قرار می‌دهیم.

$$A[2][3] = \min_{2\le k\le 3}(A[2][k-1]+A[k+1][3]) + \sum_{m=2}^{3} P_m$$

$$= \text{Min}(A[2][1]+A[3][3], A[2][2]+A[4][3]) + P_2 + P_3$$
$$= \text{Min}(\underbrace{0+0.1}_{k=2}, \underbrace{0.2+0}_{k=3}) + 0.2 + 0.1 = 0.4$$

و در خانه R[2][3] عدد k = 2 را قرار می‌دهیم .

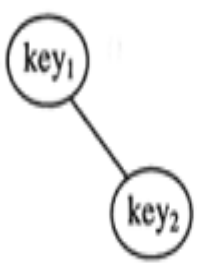
قطر ۳ :

$$A[1][3] = \min_{1 \le k \le 3} (A[1][k-1] + A[k+1][3]) + \sum_{m=1}^{3} P_m$$
$$= \text{Min}(A[1][0]+A[2][3], A[1][1]+A[3][3], A[1][2]+A[4][3]) + P_1 + P_2 + P_3$$
$$= \min(\underbrace{0+0.4}_{k=1}, \underbrace{0.7+0.1}_{k=2}, \underbrace{1.1+0}_{k=3}) + 0.7 + 0.2 + 0.1 = 1.4$$

و در خانه R[1][3] عدد k = 1 را قرار می‌دهیم.

پس درخت BST بهینه با کلیدهای key_1 تا key_3 در کل به زمان جستجوی میانگین 1.4 نیاز دارد. برای ترسیم شکل به ماتریس R نگاه می‌کنیم. برای key_1 تا key_3 ریشه key_1 می‌باشد (R[1][3] = 1) پس آن را در ریشه قرار می‌دهیم :

برای key_2 تا key_3 ریشه key_2 است چرا که R[2][3]=2 می‌باشد :



توجه کنید چون می‌دانیم $key_2 > key_1$ است آن را در سمت راست key_1 قرار داده‌ایم.

در آخر هم key_3 را در سمت راست key_2 رسم می‌کنیم.

پیاده‌سازی الگوریتم فوق را به عنوان تمرین برعهده دانشجویان قرار می‌دهیم. البته کدهای آن را به زبان C می‌توانید در کتاب نیپولیتان مشاهده کنید.

مشابه الگوریتم ضرب ماتریس‌ها این الگوریتم نیز نیاز به سه حلقه تودرتو دارد و لذا مرتبه اجرایی آن $\theta(n^3)$ است.

وضعیت کلید	پاسخ صحیح	شماره سوال
عادي	ج	1
عادي	الف	2
عادي	ج	3
عادي	الف	4
عادي	ج	5
عادي	ب	6
عادي	د	7
عادي	ب	8
عادي	الف	9
عادي	د	10
عادي	ج	11
عادي	الف	12
عادي	د	13
عادي	ب	14
عادي	د	15
عادي	ج	16
عادي	الف	17
عادي	ج	18
عادي	ب	19
عادي	ج	20
عادي	د	21
عادي	الف	22
عادي	ب	23
عادي	ب	24
عادي	الف	25