موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

درس و کنکور ارشد

ساختمان داده و طراحی الگوریتم

(حل تشریحی سوالات دولتی ۱۳۹۷)

ویژه‌ی داوطلبان کنکور کارشناسی ارشد مهندسی کامپیوتر و IT

براساس کتب مرجع

توماس اچ کورمن، چارلز ای لیزرسان، رونالد ال ریوست و کلیفورد استین

ابوالفضل گیلک

کد کنترل

260

E

260E

«اگر دانشگاه اصلاح شود مملکت اصلاح می‌شود»
امام خمینی (ره)

عصر پنج‌شنبه
۹۷/۲/۶

جمهوری اسلامی ایران
وزارت علوم، تحقیقات و فناوری
سازمان سنجش آموزش کشور

آزمون ورودی دوره‌های کارشناسی‌ارشد ناپیوستهٔ داخل ـ سال ۱۳۹۷

مهندسی فناوری اطلاعات (IT) ـ کد (۱۲۷۶)

تعداد سؤال: ۱۰۰

مدت پاسخگویی: ۲۱۰ دقیقه

عنوان مواد امتحانی، تعداد و شمارهٔ سؤالات

ردیف	مواد امتحانی	تعداد سؤال	از شمارهٔ	تا شمارهٔ
۱	زبان عمومی و تخصصی (انگلیسی)	۳۰	۱	۳۰
۲	دروس مشترک (ساختمان‌های گسسته، ساختمان داده‌ها، طراحی الگوریتم، مهندسی نرم‌افزار، شبکه‌های کامپیوتری)	۳۰	۳۱	۶۰
۳	اصول و مبانی مدیریت	۲۰	۶۱	۸۰
۴	مجموعه دروس تخصصی مشترک (اصول طراحی پایگاه داده‌ها، هوش مصنوعی، سیستم‌های عامل)	۲۰	۸۱	۱۰۰

این آزمون نمرهٔ منفی دارد.

استفاده از ماشین‌حساب مجاز نیست.

۱۳۹۷

۳۷- جواب رابطهٔ بازگشتی $T(n)=2T(\frac{n}{4})+\log n$، کدام است؟

۱) $O(\sqrt{n})$

۲) $O(\log n)$

۳) $O(\log^2 n)$

۴) $O(\sqrt{n}\log n)$

۳۸- فرض کنید n عدد داخل آرایهٔ A، به‌صورتی قرار گرفته است که به‌ازای هر اندیس $i \le n-k$ داریم $A[i]>A[i+k]$ (k مقداری ثابت است). این آرایه را در چه زمانی می‌توان مرتب کرد؟ (بهترین پاسخ را انتخاب کنید.)

۱) $O(n)$

۲) $O(n\log k)$

۳) $O(n\log n)$

۴) $O(n\log_k n)$

۳۹- یک داده‌ساختار در اختیار داریم که از یک هرم کمینه و یک پشته تشکیل شده است. در ابتدا هر دو خالی هستند. در هر مرحله یکی از کارهای زیر را می‌توان انجام داد.

- یک عدد از ورودی خواند و آن را داخل هرم کمینه ریخت.
- کوچک‌ترین عدد را از هرم کمینه استخراج کرد و داخل پشته **push** کرد.
- عمل **pop** را روی پشته اجرا و به عنوان خروجی گزارش کرد.

اگر ورودی از چپ به راست ۳،۲،۵،۴،۶،۱ باشد، کدام خروجی (به‌ترتیب از چپ به راست) را نمی‌توان تولید کرد؟

۱) هر جایگشت از اعداد ۱ تا ۶ را می‌توان تولید کرد.

۲) ۶،۵،۴،۳،۲،۱

۳) ۶،۱،۲،۳،۵،۴

۴) ۱،۲،۳،۴،۵،۶

۴۰- کدام تابع درهم‌ساز داده شده $h(x)$، یکنواخت است؟ (برد توابع اعداد طبیعی است.)

۱) $۲^x \bmod ۷$ ۲) $۳^x \bmod ۱۱$

۳) $x^۲ \bmod ۷$ ۴) $x^۲ \bmod ۱۱$

۴۱- یک آرایه با اندازهٔ n داریم که هر خانهٔ آن به یک گره دلخواه از یک لیست پیوندی دوسویه اشاره می‌کند. هیچ دو خانه‌ای از آرایه به یک گره اشاره نمی‌کند. لیست پیوندی دوسویه دقیقاً شامل n گره است و هر گره یک عدد متمایز در خود نگه داشته است. لیست پیوندی بر اساس این اعداد به صورت صعودی مرتب شده می‌باشد. (یعنی اگر لیست را از ابتدا تا انتها پیمایش کنیم اعداد به‌صورت صعودی مشاهده می‌شوند.) آیا عدد داده شدهٔ x در لیست پیوندی دوسویه موجود است و در بدترین حالت با چه مرتبهٔ زمانی می‌توان این موضوع را تشخیص داد؟

۱) $O(۱)$ ۲) $O(n)$

۳) $O(\sqrt{n})$ ۴) $O(\log n)$

۴۲- طول متن uselessness . با کدگذاری هافمن چند بیت می‌شود؟

۱) ۲۰ ۲) ۲۲

۳) ۲۵ ۴) ۲۸

۴۳- در یک گراف جهت‌دار و وزن‌دار (با وزن‌های مثبت) برای محاسبهٔ کوتاه‌ترین مسیر از مبدأ داده شده به بقیهٔ رأس‌ها از الگوریتم دایکسترا استفاده شده است. کدام مورد، همیشه درست است؟

منظور از relax یال (u,v) در الگوریتم دایکسترا تابع زیر است، که در آن $w(u,v)$ وزن یال (u,v) است.

```
relax(u,v):
    if d[v]>d[u]+w(u,v) then d[v]=d[u]+w(u,v)
```

۱) هر یال دقیقا یک بار relax می‌شود.

۲) هر یال حداکثر یک بار relax می‌شود.

۳) ممکن است بعضی از یال‌ها بیش از یک بار relax شوند.

۴) مثالی وجود دارد که بعضی از یال‌ها بیش از یک بار relax می‌شوند، اما به‌طور متوسط تعداد relax ها $O(۱)$ است.

۴۴- در آرایهٔ مرتب شدهٔ A تمام عناصر به جز یک عنصر که تنها یک بار ظاهر شده است، دقیقاً دو بار ظاهر شده‌اند. عنصری که تنها یک بار ظاهر شده را در چه زمانی می‌توان به‌دست آورد؟

۱) $O(۱)$ ۲) $O(n)$

۳) $O(\log n)$ ۴) $O(\frac{n}{\log n})$

۴۵- برای پیدا کردن کوتاه‌ترین مسیر .ن تمام رأس‌های یک گراف وزن‌دار تنک (که تعداد یال‌های آن از مرتبهٔ تعداد رأس‌ها است)، استفاده از کدام الگوریتم از نظر مرتبهٔ زمانی بهتر است؟ (وزن یال‌ها می‌تواند منفی هم باشد.)

۱) الگوریتم جانسون ۲) الگوریتم فلوید ـ وارشال

۳) n بار اجرای الگوریتم دایکسترا برای هر رأس ۴) n بار اجرای الگوریتم بلمن ـ فورد برای هر رأس

۴۶- یک برنامه‌نویس تازه‌کار، تابع زیر را برای محاسبهٔ $\binom{n}{k}$ نوشته است. مدیر شرکت به‌عنوان تنبیه دستور داده است که این برنامه‌نویس تعداد بارهایی که این الگوریتم خودش را برای محاسبهٔ $\binom{۲۰}{۳}$ صدا می‌زند را به‌صورت دستی محاسبه کند. این تعداد چند مرتبه است؟

```
int comb (int n, int k) {
if ((k = 0) or ( n = k))
    return 1
else
   return comb (n-1 , k-1 ) + comb (n-1, k)
 }
```

۱) ۱۹۳۹
۲) ۱۹۵۷
۳) ۲۲۷۹
۴) ۲۲۹۱

۴۷- برای محاسبهٔ شار بیشینه در شبکه داده شده با n رأس و m یال و ظرفیت‌های صحیح، دو الگوریتم با زمان‌های (i) $O(mnC)$ و (ii) $O(m^۲ \log C)$ وجود دارد که C بیشترین ظرفیت یال‌ها است. زمان اجرای کدام الگوریتم برحسب اندازه ورودی، چند جمله‌ای است؟

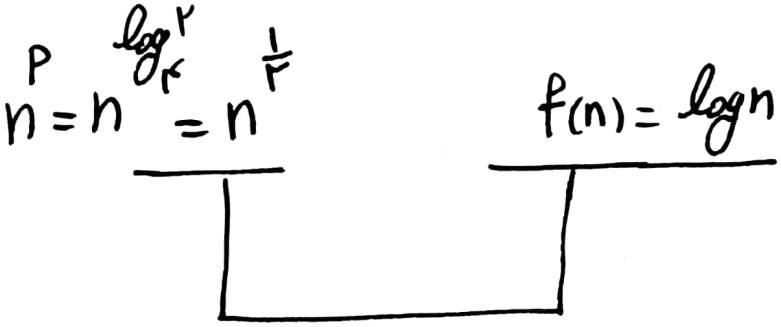
۱) (i) و (ii) ۲) فقط (i) ۳) فقط (ii) ۴) هیچ‌کدام

۴۸- فرض کنید یک گراف جهت‌دار G داده شده است. الگوریتم جستجوی عمق اول (DFS) را بر روی گراف G اجرا می‌کنیم تا همهٔ رئوس گراف را ملاقات کنیم. برای هر گره v، s_v و f_v را به‌ترتیب زمان قرار گرفتن v در پشته و زمان خارج شدن از پشته تعریف می‌کنیم. گراف G' را گرافی در نظر بگیرید که هر گره آن معادل یک مؤلفهٔ ماکزیمال همبند قوی از G است. از u' به v' در G' یال جهت‌دار وجود دارد. اگر از یک رأس از مؤلفه همبند معادل u' به یک رأس مؤلفهٔ همبند معادل v' یال وجود داشته باشد. برای هر گره u در G، فرض کنید u' گره‌ای در G' باشد که معادل مؤلفهٔ همبندی است که u متعلق به آن است. کدام مورد درست است؟

۱) اگر u رأسی با بیشترین s_u باشد، u' ورودی ندارد.
۲) اگر u رأسی با بیشترین s_u باشد، u' خروجی ندارد.
۳) اگر u رأسی با بیشترین f_u باشد، u' ورودی ندارد.
۴) اگر u رأسی با بیشترین f_u باشد، u' خروجی ندارد.

۳۷ پاسخ: گزینه (۱)

یک مسئله‌ی بسیار ساده از کاربرد قضیه‌ی اساسی:



ابتدا به درجه‌ی جمله‌ای اهمیت می‌دهیم.
اگر از نظر درجه‌ی جمله‌ای برابر بودند آنگاه به جملات لگاریتمی توجه می‌کنیم.

در اینجا $n^{\frac{1}{2}}$ از نظر درجه‌ی جمله‌ای، از $f(n)$ بزرگتر است پس

$$T(n) = \theta(n^{\frac{1}{2}}) = \theta(\sqrt{n})$$

وقتی θ باشد، O هم هست.

۳۸ گزینه (۲)

شرط $A[i] > A[i+K]$ ایجاب می‌کند که آرایه‌ی A به K تا زیرآرایه هرکدام به طول $\Theta(\frac{n}{K})$ تفکیک شده است. مرتب سازی وادغام K تا آرایه‌ی مرتب که هرکدام دارای m عضر باشند، اگر به بهترین شکل انجام شود، از مرتبه‌ی $O(mK\log K)$ است.

در اینجا $m=\frac{n}{K}$ است پس:

$$mK\log K = \frac{n}{K} K\log K = n\log K$$

توجه 1: این‌که هرکدام از لیست‌های به طول K، ترتیب نزولی دارند، اشکالی ایجاد نمی‌کند ما می‌توانیم ادغام را از انتهای لیست‌ها انجام دهیم نه از ابتدای آنها. یا می‌توانیم پس از ادغام معمولی و یا متن آرایه‌ی نزولی $A[1,2,\dots,n]$ به راحتی و از مرتبه‌ی $\Theta(n)$ این لیست را وارونه کنیم

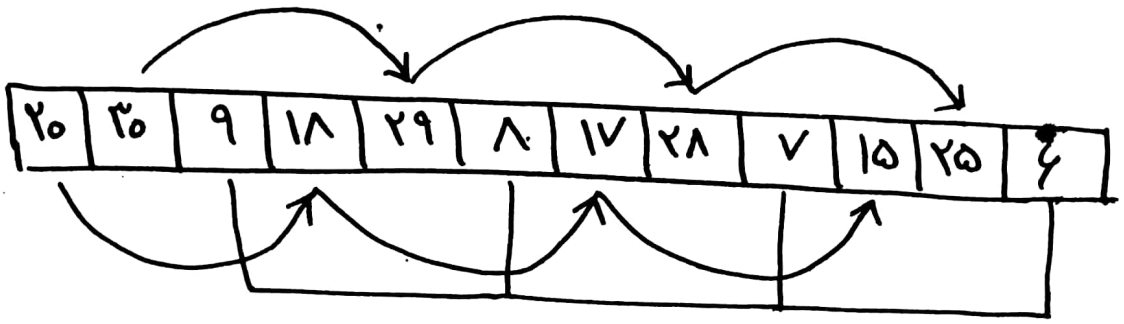
برای مثال با این کُد:

```
for  i=1  to  n
```

$$B[i] = A[n-i+1]$$

یک آرایه ترولی به آرایه صعودی تبدیل می‌شود.

توجه ۲ برای درک بهتر به این مثال برای $k=۳$ توجه کنید:



شرط $A[i] > A[i+۳]$ برقرار است. یعنی کل این آرایه، به ۳ تا زیرآرایه‌ی ترولی تفکیک می‌شود:

اندیس‌های $3k+1$: $۲۰ > ۱۸ > ۱۷ > ۱۵$

اندیس‌های $3k+2$: $۳۵ > ۲۹ > ۲۸ > ۲۵$

اندیس‌های $3k$: $۹ > ۸ > ۷ > ۶$

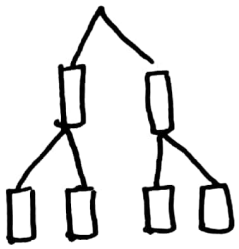
فقط می‌ماند ادغام این ۳ آرایه که طول هرکدام $\frac{n}{۳}$ است.

۲۸-۴

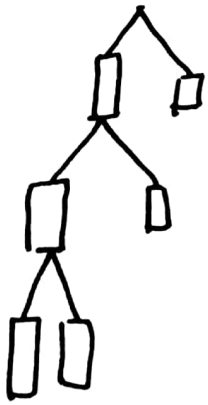
توجه ۳ (بیارمم)

فرض کنید k آرایه‌ی مرتب شده دارید و طول هرکدام از آن‌ها m است. همچنین فرض کنید تعداد کل اعداد $n = k \cdot m$ است.

الف) اگر ادغام این لیست‌ها به صورت بهینه (متوازن) انجام شود مرتبه‌ی زمانی آن $n \log k$ است.

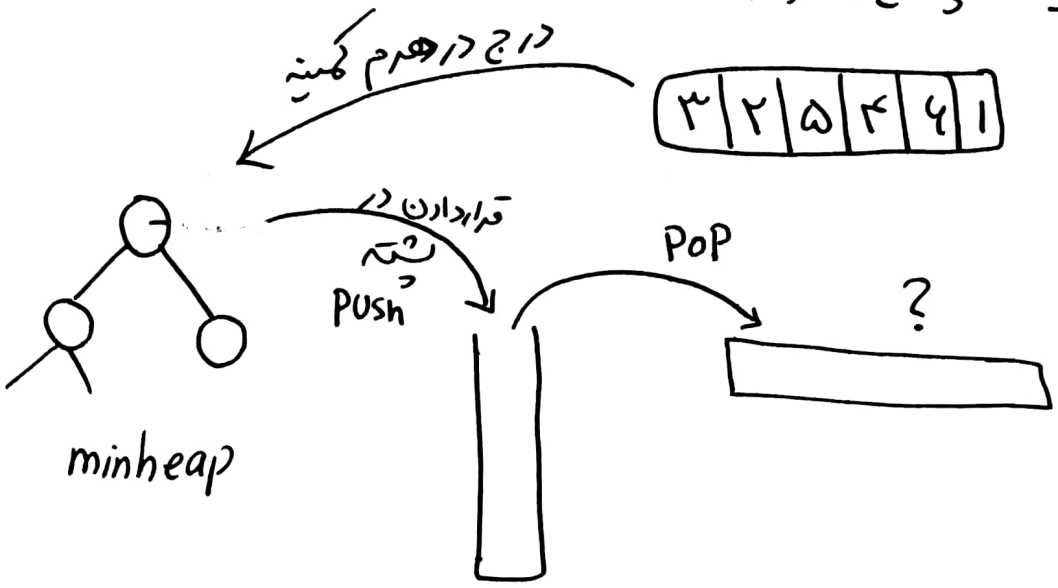


ب) اگر ادغام به این صورت انجام شود که:
لیست اول و دوم ادغام شوند، سپس نتیجه‌ی آن با لیست سوم ادغام شود. سپس نتیجه‌ی آن با لیست چهارم ادغام شود و ...
در این صورت، مرتبه زمانی آن nk است.



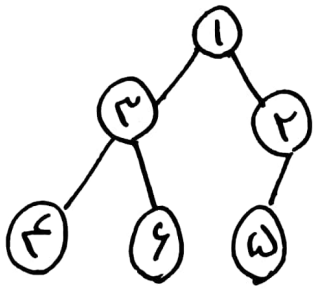
ج: یا باید همه‌ی لیست‌ها صعودی باشند،
یا باید همه‌ی لیست‌ها نزولی باشند.

۳۹ پاسخ: گزینه (۳)



گزینه های (۲) و (۴) که خروجی های نزولی و صعودی هستند به وضوح قابل دست یابی اند.

(گزینه ۲) اگر ابتدا همه ی اعداد را در هرم کمینه درج کنیم، خواهیم داشت:



(شکل نهایی هرم، فقط یکی از حالت هایی است که همه در آن درج شده باشند.)

حالا عملیات Push را برای همه اعداد انجام دهیم:

به ترتیب این ها وارد پشته می شوند:

اول: 1 چهارم: 4
دوم: 2 پنجم: 5
سوم: 3 ششم: 6

۳۹/۲

حالا عملیات pop را تا انتهای پشته انجام می‌دهیم:

6 5 4 3 2 1

پس ترتیب نزولی به دست خواهد آمد.

(گزینه ۴): ابتدا هم را در هدم کمتر درج کنیم.

حالا اعمال push و pop را پشت سر هم انجام دهیم.

1 از هدم وارد پشته می‌شود و بلافاصله از آن خارج می‌شود.
2 از هدم وارد پشته می‌شود و بلافاصله از آن خارج می‌شود.
⋮

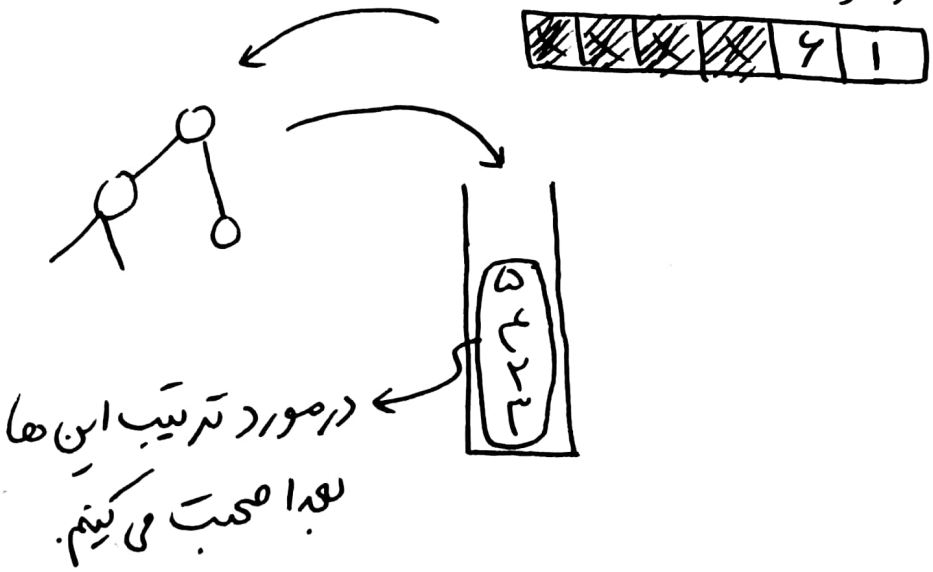
به این ترتیب این خروجی هم به دست خواهد آمد:

1 2 3 4 5 6

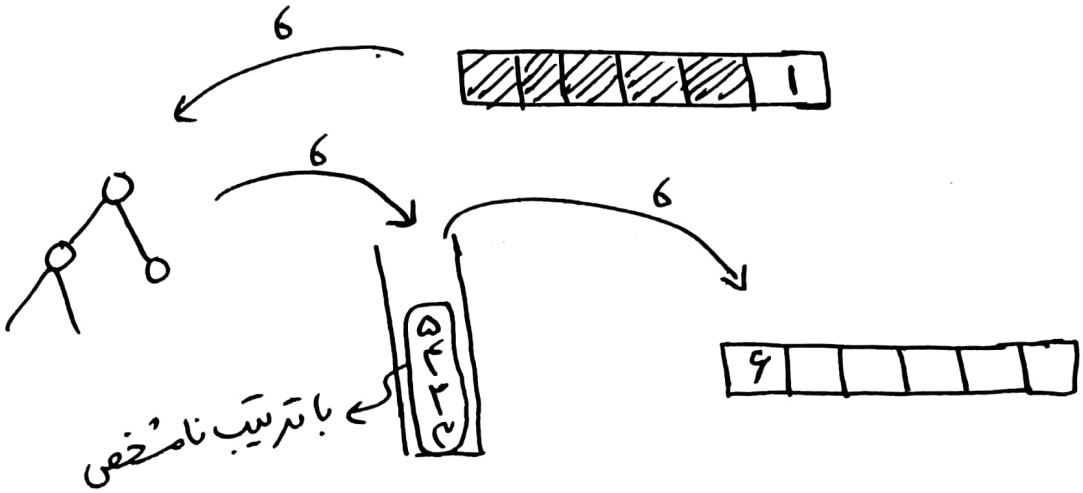
۳۹/۴

(گزینه ۳): هدف ما: ایجاد خروجی ۶۱۲۳۵۴ است.

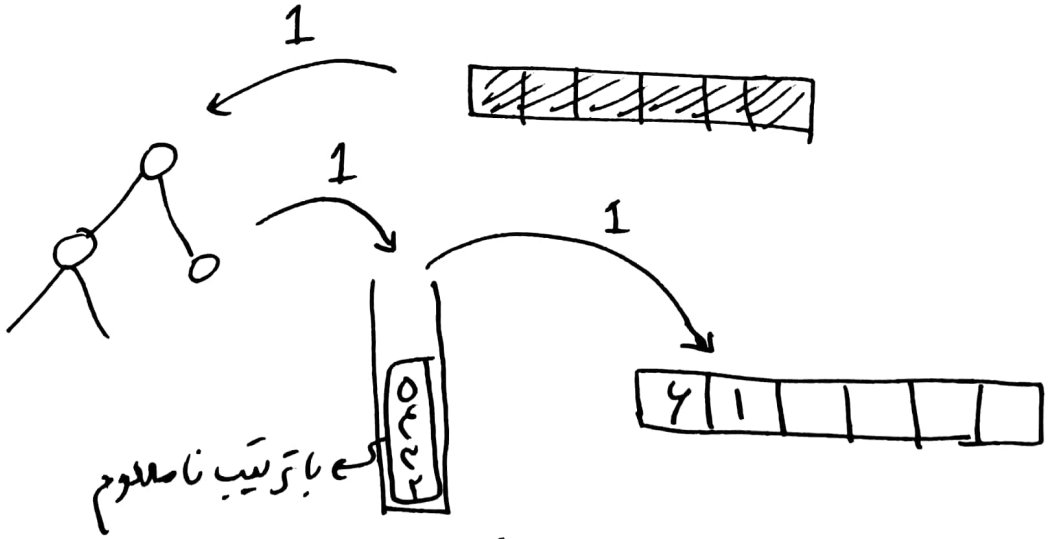
به محل ۶ در ورودی برنامه توجه کنید. می بینید که عددهای ۳,۲,۵,۴ قبل از ۶ در هدم درج می شوند. اگر می خواهیم ۶ اولین عدد در لیست خروجی باشد، ابتدا باید ۳,۲,۵,۴ را به ترتیبی که فعلاً به ترتیب آن کاری نداریم، وارد هدم کرده و سپس از آن خارج کنید و در پشته ذخیره کنید.



حالا عدد ۶ را در هدم درج می کنیم، بلافاصله آن را خارج کرده در پشته قرار می دهیم. بلافاصله از پشته خارج کرده و در خروجی قرار می دهیم:

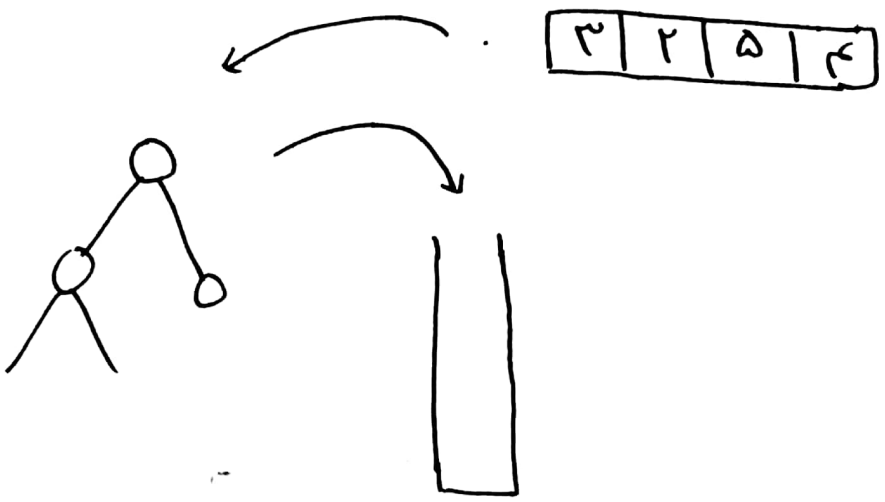


در ادامه می خواهیم رقم 1 بعد از 6 در خروجی آمده باشه
پس برای 1 هم مانند 6، این ۳ عملیات را پشت سر هم
انجام می دهیم: درج در هرم کمینه
خروج از هرم و درج در پشته
خروج از پشته و قرار گرفتن در خروجی.



تا اینجا موفق شدیم 1 و 6 را ایجاد کنیم. حالا دوست داریم که
عدد بعدی که از پشته خارج می شود، ۲ باشه.

۳۹/۵ اما این امکان ندارد زیرا ترتیب داده های ورودی چنین است:



درضمن اگر یادتان باشه، قرار شد که این ۴ عدد، زودتر از ۶ از پشته خارج نشوند. از آنجاکه فعلاً فقط قصد ذخیره کردن این ۴ عدد در پشته را داریم و هر بار عضو min از هردم خارج می شود، امکان ندارد که ۲ آخرین ورودی به پشته باشه. پس گزینه (۳) یعنی ترتیب خروجی 6 1 2 3 5 4 → را نمی توان ایجاد کرد. خود به خود گزینه (۱) هم نادرست است.

۴۵ پاسخ: گزینه‌ی (۲)

مقدمه: آنچه ما از یک تابع درهم ساز می‌خواهیم انتظار داریم آن است که تا حد امکان ۲ کلید متفاوت را به یک اندیس مربوط نکند.

یکی از راه‌های ایجاد یک تابع درهم ساز، استفاده از روش تقسیم یعنی استفاده از رده‌های هم‌نهشتی به پیمانه‌ی m است.

انتخاب یک پیمانه‌ی مناسب، بسیار مهم است. برای مثال معروف است که انتخاب $m=2^p$ و $m=2^p-1$ انتخاب خوبی نیست. مثلاً اگر از $m=2^2=4$ به عنوان پیمانه استفاده کنیم در تابع درهم ساز زیر می‌بینید که خروجی تابع تنها، حالت‌های تکراری بسیار زیادی دارد:

$$h(x)=(x^2 \bmod 4)$$

به چند مقدار آن توجه کنید:

$$x=1 \Rightarrow h(1)=1$$
$$x=2 \Rightarrow h(2)=0$$
$$x=3 \Rightarrow h(3)=1 \qquad x=4 \Rightarrow h(4)=0$$
$$x=5 \Rightarrow h(5)=1$$
$$\vdots \qquad\qquad \vdots$$

یعنی تابع درهم ساز شما، کلیدهای $x=1$ و $x=3$، $x=5$، ... را به خانه‌ی با اندیس $i=1$ می‌برد.

و کلیدهای فرد را هم به خانه‌ی با اندیس $i=0$ می‌برد،
درضمن خانه‌های با اندیس $i=2,4,6,\dots$ هم بلا استفاده و
خالی مانده‌اند.
پس این تابع درهم‌ساز، به هیچ وجه، یکنواخت نیست و
درضمن مشکل دیگری هم که دارد خروجی $h(x)=0$ است
که طبق صورت سؤال، ایراد دارد چون نُبرد و خروجی
توابع، قرار است مجموعه اعداد طبیعی باشد.

توجه: پیمانه‌ی مناسب برای تابع درهم‌ساز، معمولاً یک عدد
اول است که تا حد امکان از توان‌های ۲ فاصله
داشته باشد.
مثلاً در این تست، پیمانه‌ی $m=7$ مناسب نیست
چون $7=2^3-1$ اما پیمانه‌ی ۱۱ مناسب
است زیرا با توان‌های ۲ تا حد امکان فاصله دارد:

۲ ۴ ۸ ۱۶ ۳۲

مکان $m=11$

۴۵/۴

پاسخ سؤال ۱:

می‌دانیم که پیمانه‌ی $m=2^p-1$ مناسب نیست پس گزینه‌های (۱) و (۳) نادرست هستند. [بعداً آنها را دقیق‌تر بررسی می‌کنیم]

گزینه‌ی (۴) هم یکنواخت نیست و دو ایراد دارد:

اولاً: خروجی آن ممکن است صفر شود. برای مثال وقتی $x=11$ یا $x=22$ یا ... مضرب‌های ۱۱ باشند، داریم:

$$h(11k) = \left[(11k)^2 \bmod 11 \right] = 0$$

در حالی که طبق صورت سؤال خروجی باید عدد طبیعی باشد.

ثانیاً: تعداد بسیار زیادی از کلیدها به یک اندیس مشترک می‌روند. در واقع اگر بخواهیم $h(x)=h(y)$ باشد داریم:

$$x^2 \equiv y^2 \pmod{11}$$

$$\Rightarrow \quad x^2 - y^2 \equiv 0 \pmod{11}$$

$\Rightarrow$ $x^2 - y^2$ مضرب ۱۱ است

$\Rightarrow$ $(x-y)(x+y)$ مضرب ۱۱ است.

این تابع برای خیلی از عددها رخ می‌دهد. مثلاً برای اعداد ۱, ۱۲, ۲۳, ۳۴, ... که اختلاف آنها 11 تا است، همه‌ی این اعداد به یک اندیس مشترک می‌روند.

بررسی گزینه (۳) می‌دانیم که تنها گزینه‌ی قابل قبول، این گزینه است.

$$h(x) = (2^x \bmod 11)$$

اولاً هیچ‌گاه 2^x مضرب ۱۱ نمی‌شود پس هیچ‌گاه $h(x) = 0$ نمی‌شود.

تابعاً هیچ‌گاه دو کلید مختلف به یک انسان نمی‌رونه

زیرا اگه $h(x)$ و $h(y)$ بخواهند باهم برابر شوند

باید: $2^x \equiv 2^y$ (در پیمانه‌ی $m=11$).

پس باید $2^x - 2^y$ به ۱۱ بخش‌پذیر شود که

این هم رخ نمی‌دهد.

۴۵/۶

راه حل عددی

فقط به عنوان یک روش برای رد گزینه‌ها نه روشی برای اثبات درستی جواب، می‌توانید خروجی $h(x)$ را برای $x=1,2,3,\dots$ بررسی کنید.

گزینه (۱) $h(x)=2^x \bmod 7$

توان‌های ۲	۲	۴	۸	۱۶	۳۲	۶۴	...
باقی‌مانده تقسیم بر ۷	۲	۴	۱	۲	۴	۱	...

خروجی $h(x)$ مدام تکرار می‌شود.
پس یکنواخت نیست و بسیاری از اعداد طبیعی هم اصلاً به دست نمی‌آیند.

گزینه (۳) $h(x)=x^2 \bmod 7$

مقادیر x^2	۱	۴	۹	۱۶	۲۵	۳۶	...
باقی‌مانده تقسیم بر ۷	۱	۴	۲	۲	۴	۱	...

باز هم خروجی $h(x)$ تکراری است.

گزینه (۲)

$$h(x) = 2^x \bmod 11$$

۴۵/۷

توان‌های ۲	۲	۴	۸	۱۶	۳۲	۶۴	...
باقی‌مانده تقسیم بر ۱۱	۲	۴	۸	۵	۱۰	۹	...

به نظر می‌رسد خروجی $h(x)$
تکراری نیست و هر عدد طبیعی فقط یک بار ظاهر می‌شود.

گزینه (۴)

$$h(x) = x^2 \bmod 11$$

مقادیر x^2	۱	۴	۹	۱۶	۲۵	۳۶	۴۹	...
باقی‌مانده تقسیم بر ۱۱	۱	۴	۹	۵	۳	۳	۵	...

در خروجی، تکرار دیده می‌شود.

۴۱/۱

۴۱ پاسخ: گزینه (۲)

مساله بیار واضح است. با توجه به آن که اشاره‌گرها لزوماً صعودی نیستند پس مرتب بودن اعداد در لیست پیوندی قابل استفاده نیست و نمی‌توانیم به مرتبه‌ی $O(\log n)$ برای جستجوی x برسیم.

بنابراین این مساله معادل است با جستجوی کلید x در یک لیست نامرتب به طول n و مرتبه‌ی اجرای آن $O(n)$ است.

۴۲ پاسخ: گزینه (۲) ۴۲/۱

حروف	u	s	e	l	n
فرکانس (فراوانی) f_i	۱	۵	۳	۱	۱

به ترتیب صعودی مرتب می‌کنیم:

حروف	u	l	n	e	s
f_i	۱	۱	۱	۳	۵

ابتدا u و l برگ‌ها هستند. ($f_u + f_l = ۲$

پس n با این دو گره، یک زیردرخت می‌سازد:

$f_u + f_l + f_n = ۳$

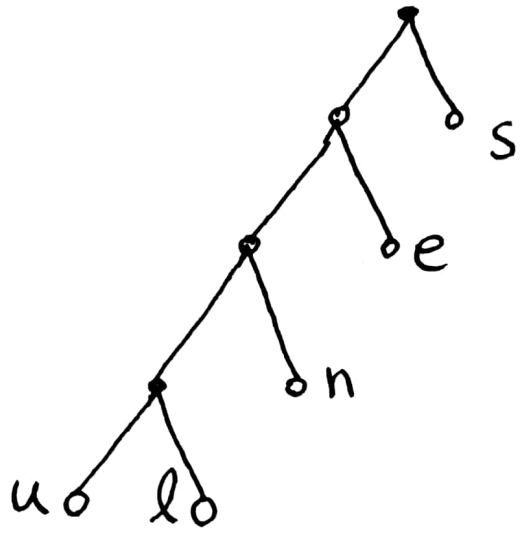
حالا این اعداد را داریم:

uln	e	s
۳	۳	۵

و e و uln یک زیردرخت می‌سازند.

در پایان هم s را به آن‌ها ملحق می‌کنیم.

۴۲/۲



$$\text{طول متن (هزینه درخت)} = \sum (\text{عمق برگ}) \times (\text{فراوانی})$$

$$= ۱(۵) + ۲(۳) + ۳(۱) + ۴(۱) + ۴(۱)$$

$$= ۲۲$$

۴۳ پاسخ: گزینه‌های (۱) و (۲)

۴۳/۱

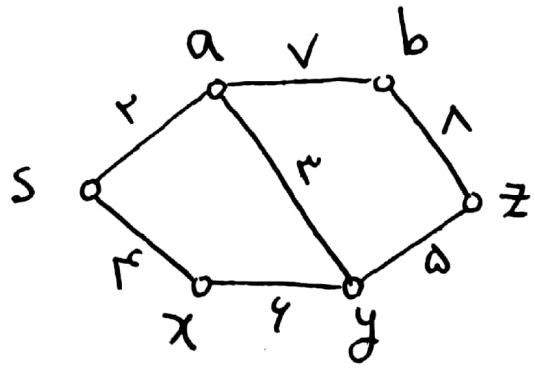
در الگوریتم دکسترا، برای هر یال $(u,v) \in E$ دقیقاً یک بار $d(v)$ و $d(u)+w(u,v)$ را مقایسه می‌کنیم.

در ضمن اگر نامساوی $d(v) > d(u)+w(u,v)$ برقرار باشد آنگاه به روزرسانی $d(v)=d(u)+w(u,v)$ اجرا خواهد شد.

بنابراین به ازای هر یال $(u,v) \in E$، این مقایسه دقیقاً یک بار و به‌روزرسانی، حداکثر یک بار انجام می‌شود.

در ضمن وقتی جمله‌ی "دقیقاً یک بار" صحیح باشد، جمله‌ی "حداکثر یک بار هم" صحیح خواهد بود. در واقع وقتی $t=1$ باشد آنگاه $t \leq 1$ هم صحیح است. پس گزینه‌های (۱) و (۲) هردو صحیح هستند البته گزینه (۱) دقیق‌تر است.

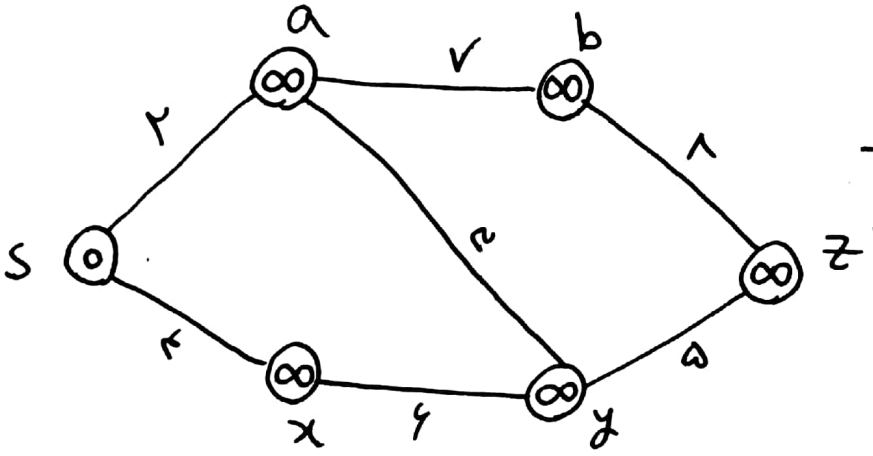
یادآوری: برای آن که پاسخ را درک کنید دکسترا را روی یک مثال اجرا می‌کنیم.



باید وزن‌ها نامنفی باشند.

هدف: کوتاه‌ترین مسیر از گره مبدا به سایر گره‌ها.

0) مقداردهی اولیه: $d(s)=0$ و برای سایر گره‌ها $d(u)=\infty$:



$S=\{\}$

$V-S=\{s,a,b,x,y,z\}$

1) کمترین مقدار d مربوط به $d(s)=0$ است پس این گره را به S اضافه کنیم:

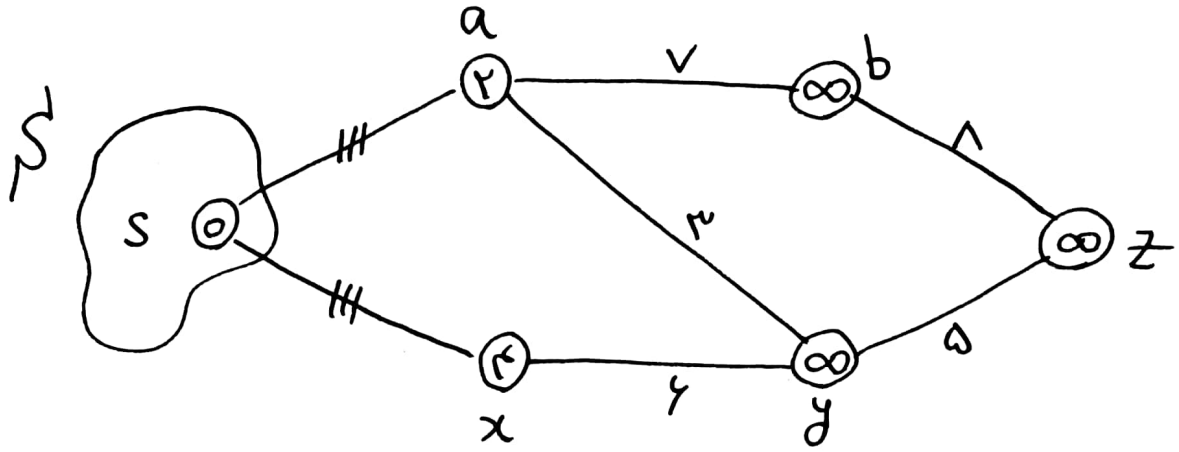
$S=\{s\}$ $\quad V-S=\{a,b,x,y,z\}$

همسایه‌های تازه وارد گره مبدا s است. یال‌های (s,a) و (s,x) را relax می‌کنیم:

۴۳/۴

مثلاً: $d(a) > d(s)+2$ است پس به روز رسانی می‌شود: $d(a)=2$

$d(x) > d(s)+4$ است پس به روز رسانی می‌شود. $d(x)=4$



(2

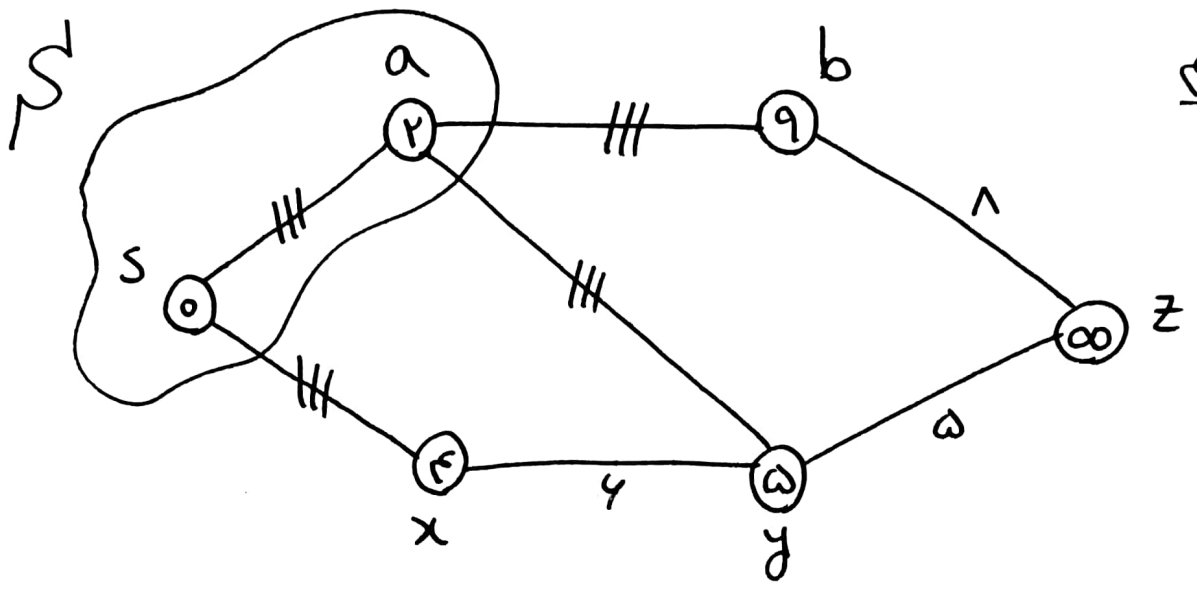
حالا کمترین مقدار d در بین گره‌های خارج از S مربوط به a است: $d(a)=2$ پس این گره را به S اضافه کنیم:

$$S=\{s,a\} \qquad V-S=\{b,x,y,z\}$$

عضو تازه وارد S همان a است پس یال‌های (a,b) و (a,y) باید relax شوند:

$d(b) > 2+7$ پس $d(b)=2+7$ می‌شود.

$d(y) > 2+3$ پس $d(y)=2+3$ می‌شود.

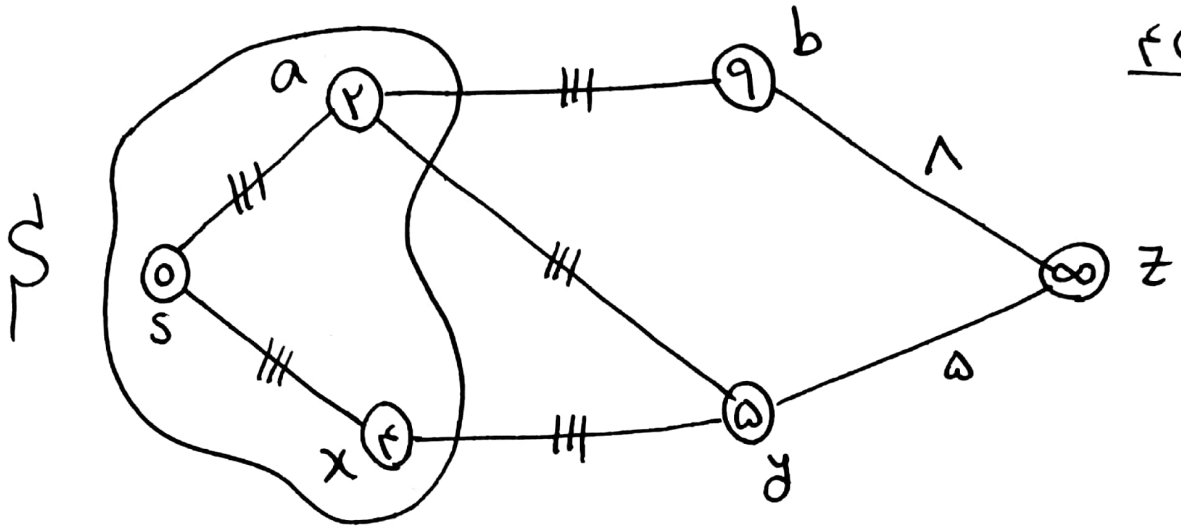


3) در بین گره های خارج از S کمترین مقدار d مربوط به x است. x را به S اضافه می کنیم.

$S = \{s, a, x\}$ $\quad V - S = \{b, y, z\}$

عضو تازه وارد S همان x است. یال هایی که از x به خارج از S می روند relax می شوند. پس یال (x,y) را relax می کنیم. اما d(y) از ۶+۴ بزرگتر نیست پس به روز رسانی نمی شود.

۴۴/۵

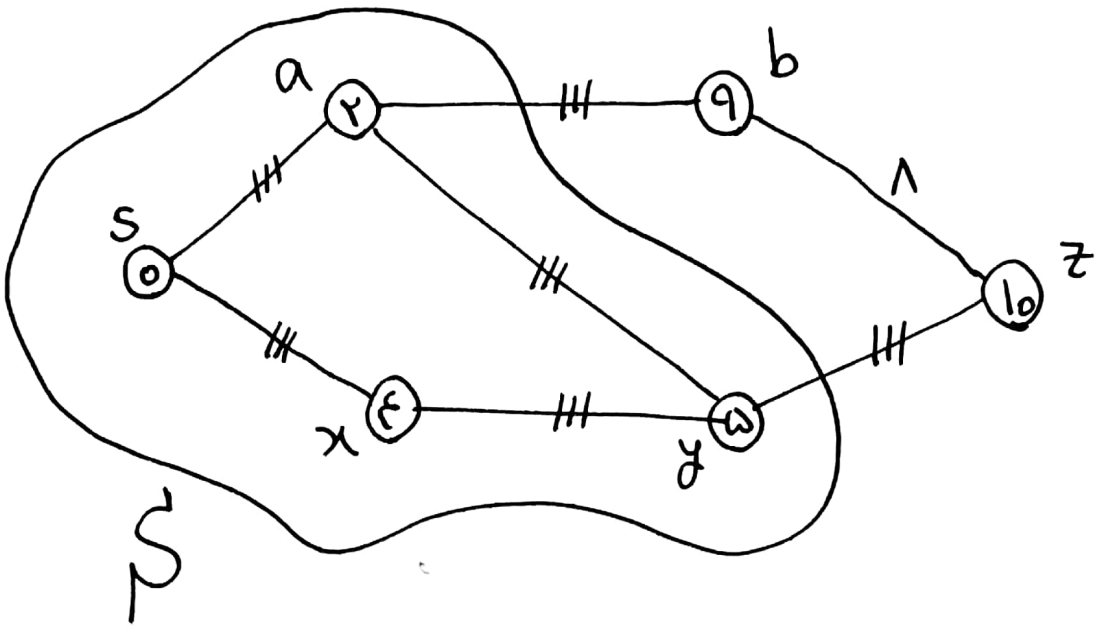


4) در بین عناصر خارج از S کمترین مقدار d مربوط به y است.

پس:

$$S = \{ s, a, x, y \}$$

یال‌هایی که از S به خارج S می‌روند را relax کنیم. پس یال (y, z) ، relax می‌شود مقدار $d(z)$ از $5+5$ بزرگتر است پس به‌روزرسانی می‌شود: $d(z) = 10$

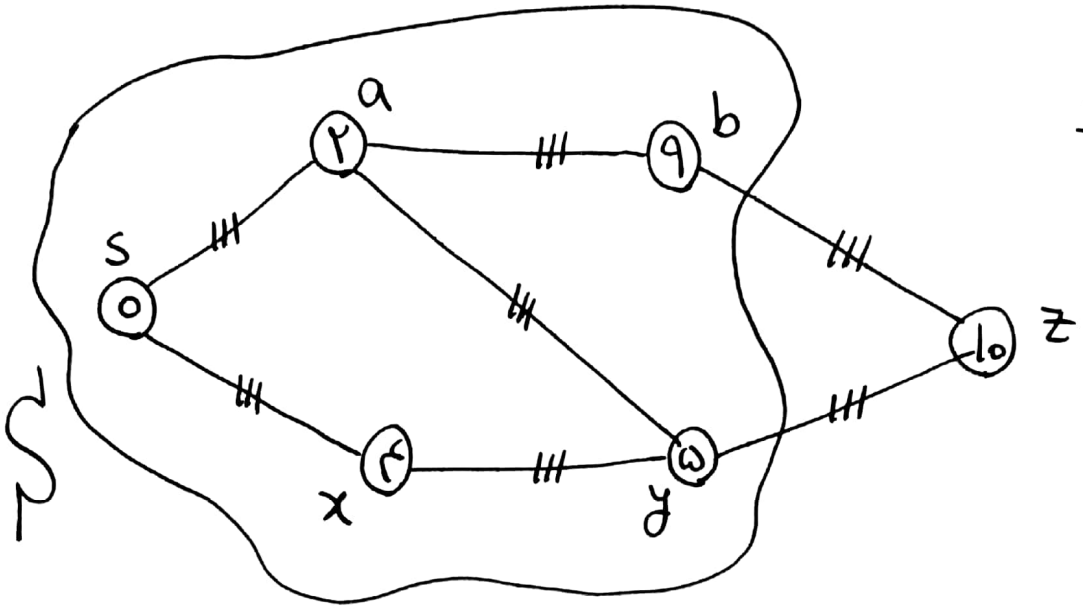


5) در خارج از S، کمترین مقدار d مربوط به گره b است. پس به S افزوده می‌شود:

$$S = \{ s, a, x, y, b \}$$

حالا یال (b,z) که از b به خارج از S می‌رود را relax می‌کنیم. مقدار فعلی $d(z)$ از مقدار $d(b) + w(b,z)$ کمتر است: $۱۰ > ۹ + ۸$ پس نیازی به به‌روزرسانی $d(z)$ نداریم.

۴۴/۷



6) فقط z در خارج از S قرار دارد. آن را به S اضافه می‌کنیم.

$$S = \{ s, a, x, y, b, z \}$$

هیچ یالی از z به خارج از S نداریم. کار تمام شده است.

نتایج مهم به دست آمده در مورد دکسترا با توجه به مثالی که به عنوان الگو مطرح کردیم:

نتایج:

۱) هر یال دقیقاً یک بار relax شد.

۲) هر گره، حداکثر به اندازه‌ی درجه‌ی ورودی اش، update می‌شود. در گراف غیر جهت دار، منظور از درجه‌ی ورودی، همان درجه است.

برای مثال گره y که درجه اش ۳ بود، فقط یک بار update شد آن هم از طریق یال (a,y).

البته یال (x,y) هم relax شد اما باعث تغییر مقدار d(y) نشد.

۳) گره z که درجه اش ۲ بود، دو بار می‌توانست update شود. یک بار وقتی که (y,z) را relax کردیم و باعث شد مقدار d(z)=∞ به d(z)=۵ تغییر کند و یک بار هم در زمان relax کردن یال (b,z) که البته باعث update مقدار d(z) نشد.

۴) وقتی دکمه ۱ به پایان می‌رسد، V = رنگ شده است. درحالی‌که در ابتدا ϕ = رنگ بود. در هر مرحله یک گره به رنگ اضافه می‌شود.

۵) وقتی دکمه ۱ به پایان می‌رسد، در هر گره، مقدار $d(u)$ نشان دهنده‌ی طول (هزینه) (وزن) کوتاه‌ترین مسیر از s به u است. اگر هیچ مسیری از s به u وجود نداشته باشد، مقدار $d(u)$ در طول اجرای الگوریتم هرگز تغییر نمی‌کند یعنی در پایان $d(u) = \infty$ خواهد بود.

۶) [مهم] اگر بخواهیم علاوه بر وزن سبک‌ترین (کوتاه‌ترین) مسیر، خود این مسیر را هم ذخیره کنیم، باید در ابتدای کار، برای همه‌ی گره‌های u، $\pi(u) = NIL$ قرار دهیم. یعنی در ابتدا هیچ گره‌ای دارای پدر (ماقبل) نیست. سپس، هرگاه که یال (u, v) را relax کردیم و باعث شد که

۴۶/۹

مقدار $d(u)$ تغییر کند، بلافاصله $\pi(u)=v$ را ذخیره کنیم.

یعنی به این صورت:

relax(v, u)

if $d(u) > d(v) + w(v,u)$ then:

$$\begin{cases} d(u) = d(v) + w(v,u) \\ \pi(u) = v \end{cases}$$

در گراف مثال قبل:

0) در ابتدای کار برای $\pi(a)$ ، $\pi(b)$ ، ... ، $\pi(z)$ ، $\pi(s)$ همگی مقادیر NIL را دارند.

1) در این مرحله (s,a) و (s,x) باعث شد $d(a)$ و $d(x)$ تغییر کنند پس: $\pi(a)=s$ و $\pi(x)=s$.

2) در این مرحله یال (a,b) باعث تغییر $d(b)$ شد پس $\pi(b)=a$

یال (a,y) باعث تغییر $d(y)$ شد پس $\pi(y)=a$

3) در این مرحله (x,y) را relax کردیم اما d(y) تغییری نکرد.

4) در این مرحله، (y,z) را relax کردیم و d(z) تغییر کرد پس $\pi(z) = y$ می‌شود.

5) در این مرحله (b,z) را relax کردیم اما d(z) تغییر نکرد.

6) در این مرحله چیزی تغییر نکرد.

بنابراین در پایان الگوریتم داریم:

$\pi(z) = y$

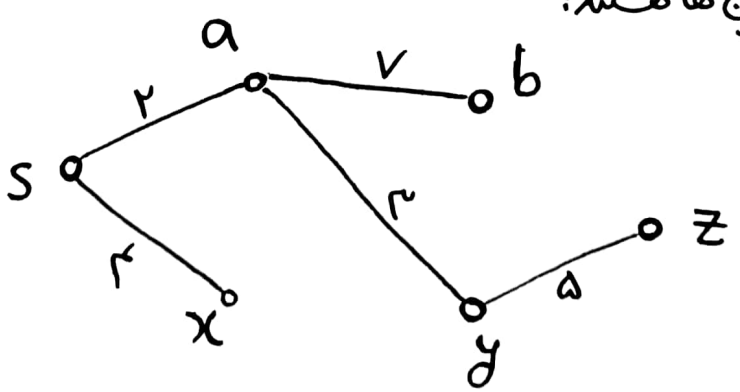
$\pi(y) = a$

$\pi(b) = a$

$\pi(a) = s$

$\pi(x) = s$

پس کوتاه‌ترین مسیرها این‌ها هستند:



۴۴ پاسخ: گزینه (۳)

ابتدا توجه کنید که آرایه‌ی مورد نظر مرتب شده است. پس عددهایی که دوبار تکرار شده‌اند حتماً در دو اندیس متوالی آمده‌اند.

ایده‌ی اصلی این است:

تا جایی که عددها به صورت جفت ظاهر شوند، همیشه $x_i = x_{i+1}$ برای فرد مشاهده می‌شود. مثلاً:

۳	۳	۶	۶	۷	۷	۱۸	۱۸	۲۳	۲۵	۲۵	۳۰	۳۰
۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳

(*)

دقت کنید تا جایی که هنوز به عدد تنهای مورد نظر نرسیده‌ایم، تساوی $x_i = x_{i+1}$ برای فرد مشاهده می‌شود.

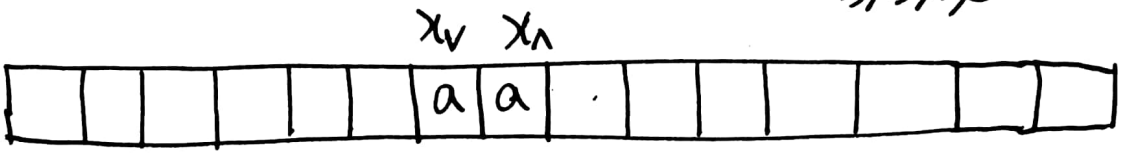
یعنی:

$$x_1 = x_2$$
$$x_3 = x_4$$
$$x_5 = x_6$$
$$x_7 = x_8$$

اما به محض آن که به عدد تنها می‌رسیم، این نظم خراب می‌شود:

$$x_9 \neq x_{10} \quad , \quad x_{11} \neq x_{12}$$

بنابراین اگر مثلاً در آرایه‌ای به طول $n=15$ به شما اطلاع بدهم که $x_7 = x_8$ است سریعاً متوجه می‌شوید که عدد تنهای مورد نظر در فاصله‌ی $A[9,10,\dots,15]$ یعنی در سمت راست قرار دارد.

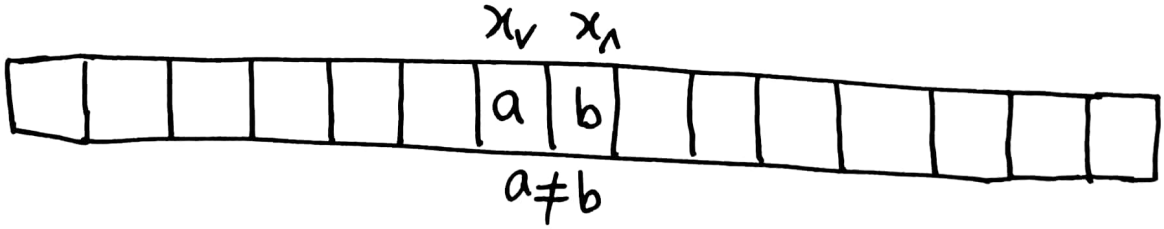


زیرا اگر آن عدد تنها در سمت چپ قرار داشت، این نظم را خراب کرده بود.

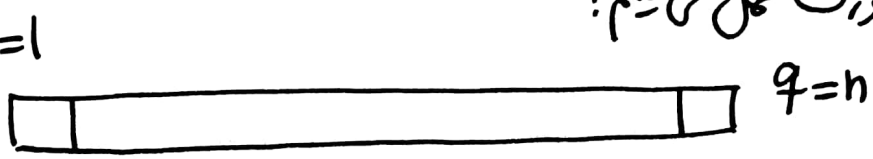
اگر هم به شما اطلاع بدهم که $x_7 \neq x_8$ است، سریعاً متوجه می‌شوید که عدد تنهای مورد نظر، در قطعه‌ی

$A[1,2,3,\dots,7,8]$

قرار دارد زیرا نظم جملات متوالی را به هم زده است.



بنابراین به این صورت عمل می‌کنیم:



0) در ابتدای کار: $p=1$ و $q=n$ است. یعنی روی $A[1,2,\dots,n]$ کار می‌کنیم.

1) قرار می‌دهیم: $i=\lfloor \frac{p+q}{2} \rfloor$

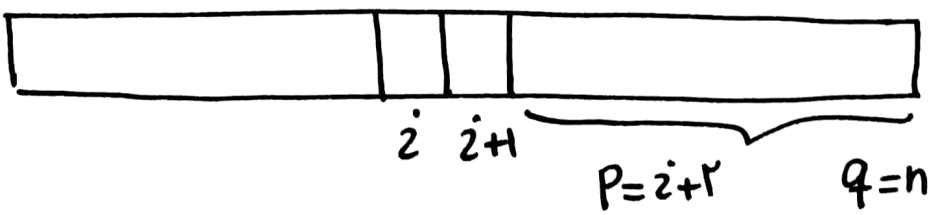
[دوست داریم i فرد باشد. پس اگر i زوج بود، فرض می‌کنیم $i=i-1$.]

2) حالا مقادیر $A[i]$ و $A[i+1]$ را مقایسه می‌کنیم.

اگر $A[i]=A[i+1]$ بود، معلوم می‌شود که عدد مورد نظر در نیمه‌ی سمت راست قرار دارد. پس برای مرحله‌ی بعد داریم:

$$p=i+2$$

و q تغییری نمی‌کند.



اما اگر $A[i] \neq A[i+1]$ بود معلوم می‌شود که عدد مورد نظر در نیمه‌ی سمت چپ قرار دارد. منظورمان نیمه‌ی $A[1,2,\ldots,i]$ است.

پس قرار می‌دهیم: $q=i$ و p تغییر نمی‌کند.

حالا به (۱) بازمی‌گردیم و این کار را آنقدر ادامه می‌دهیم که به یک خانه‌ی مختص به فرد برسیم. یعنی $p=q$ شود.

از آنجاکه در هر مرحله، اندازه‌ی مسئله، نصف می‌شود بنابراین مرتبه‌ی زمانی این الگوریتم $\Theta(\log n)$ است.

حالا با دو مثال هم، این روش را توضیح می‌دهیم:

مثال شماره‌ی (۱)

۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
۱	۱	۲	۲	۴	۴	۵	۵	۶	۶	۹	۱۰	۱۰	۱۷	۱۷

یک آرایه‌ی مرتب به طول $n=15$ داریم. قبل از اجرای الگوریتم توجه کنید که جواب نهایی در اندیس ۱۱ قرار دارد. در ابتدا $p=1$ و $q=15$ است.

مرحله اول: $i = \lfloor \frac{1+15}{2} \rfloor = 8$

چون $i=8$ زوج است یک واحد از آن کم می‌کنیم تا فرد شود.

پس $i=7$

حالا $A[7]$ و $A[8]$ را مقایسه کنیم. با هم برابرند معلوم می‌شود که هدف ما در نیمه‌ی سمت راست قرار دارد پس:

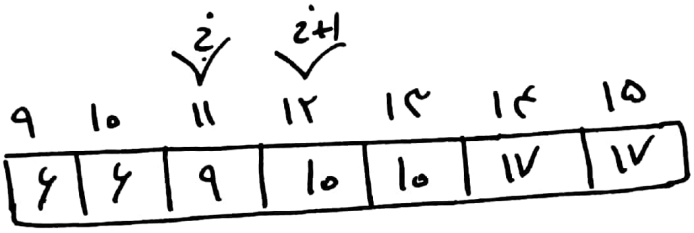
$p = i+2 = 9$

مقدار q تغییری نمی‌کند. می‌خواهیم روی نیمه‌ی $A[p, \dots, q] = A[9, 10, \dots, 15]$ کار کنیم.

مرحله دوم: $q = \lfloor \frac{9+15}{2} \rfloor = 12$

q زوج شد پس $i = q-1 = 11$.

حال $A[11]$ و $A[12]$ را مقایسه کنیم.

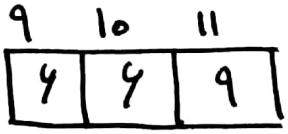


باهم برابر نیستند پس معلوم می‌شود که هدف ما در نیمه‌ی سمت چپ قرار دارد منظورمان $A[p, \ldots, i+1]$ است پس:

$q = i = 11$

و p تغییری نمی‌کند. ($p=9$)

مرحله‌ی سوم:



$q = \lfloor \frac{9+11}{2} \rfloor = 10$

q زوج شد پس $i = q-1 = 9$ را در نظر بگیریم.

$A[9]$ و $A[10]$ را مقایسه کنیم. باهم برابرند پس هدف در سمت راست قرار دارد یعنی هدف $A[11]$ است.

توجه داشته باشید که طبق الگوریتم، چون $A[i]$ با $A[i+1]$ برابر شد پس q تغییری نمی‌کند و $p=i+2$ می‌شود یعنی: $p=9+2=11$.

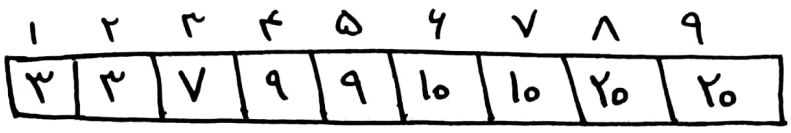
در اینجا متوجه می‌شویم که $p=q=11$ شده است در نتیجه به جواب رسیده‌ایم. (تک عضوی به دست آمد).

توجه کنید که برای $n=15$ فقط در ۳ مرحله به جواب رسیدیم زیرا

$$\lfloor \log 15 \rfloor = 3$$

است.

مثال شماره‌ی (۲)



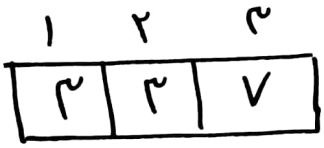
ابتدا: $p=1,\quad q=9$

مرحله اول: $i=\frac{1+9}{2}=5$ فرد است.

$A[5]$ و $A[4]$ باهم برابر نیستند پس هدف در سمت چپ است. p تغییر نمی‌کند اما $q=i=5$

مرحله دوم: $i=\frac{1+5}{2}=3$ فرد است.

$A[3]$ و $A[4]$ مقایسه شوند. باهم برابر نیستند پس هدف در سمت چپ است. p تغییر نمی‌کند. اما $q=i=3$



مرحله سوم: $i=\frac{1+3}{2}=2$ زوج است. پس $i=i-1=1$

حالا $A[1]$ و $A[2]$ مقایسه شوند. باهم برابرند پس هدف در سمت راست است. p تغییر می‌کند: $p=i+2=3$

در این جا $p=q=3$ شد پس کار تمام است.

۴۵/۱

۴۵ پاسخ: گزینه (۱)

متاسفانه یک مسأله‌ی کاملاً متکی بر حفظیات داریم زیرا تحلیل زمانی همه‌ی گزینه‌ها وقت‌گیر است.

در اینجا تک‌تک موارد در مورد مسأله‌ی کوتاه‌ترین مسیرها را مرور می‌کنیم:

الف) الگوریتم‌های بلمن فورد و دکسترا کوتاه‌ترین مسیر از گره‌ی مشخص S (منبع) به سایر گره‌ها را مشخص می‌کنند.

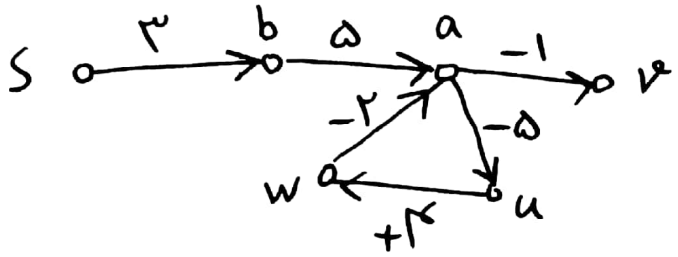
پس روش‌های تک منبع هستند.

الگوریتم دکسترا فقط برای گراف‌های با وزن مثبت به کار می‌رود اما بلمن فورد برای وزن‌های حقیقی (مثبت یا منفی) استفاده می‌شود.

در مورد گراف‌های با وزن حقیقی (مثبت و منفی) به شرطی کوتاه‌ترین مسیر از S به گره v وجود دارد که هیچ دور با وزن منفی (دوری با مجموع وزن‌های منفی) نباشد که از S قابل دسترسی باشد.

۴۵/۲

اگر یک دور با مجموع وزن منفی داشته باشیم و این دور از s قابل دسترسی باشد، الگوریتم بلمن فورد با خروجی (FALSE) اعلام می‌کند که کوتاه‌ترین مسیرها با منبع s اصلاً وجود ندارد.



مثلاً در گراف بالا کوتاه‌ترین مسیر از s به v (یعنی کم وزن‌ترین مسیر) اصلاً وجود ندارد زیرا با تکرار روی این دور می‌توان وزن مسیر را مدام کاهش داد.

ب) در هر صورت، اگر ایراد فوق وجود نداشته باشد، بلمن فورد در مدت زمان $O(VE)$ می‌تواند کوتاه‌ترین مسیرها از s به سایر گره‌ها را تعیین کند.
حالا اگر بخواهیم این الگوریتم را برای تمام گره‌ها تکرار کنیم یعنی یک بار از منبع $S=v_1$ ، بار دیگر از منبع $S=v_2$ ، ..... و در نهایت از منبع $S=v_n$ آنگاه به این نتیجه می‌رسیم:

تکرار یکبن فورد برای تمام گره‌ها در مدت زمان $O(V \times VE)$

می‌توانند کوتاه‌ترین مسیر از هر گره به هر گره را مشخص کند.

پس: گزینه (۴) $= O(V^2 E)$

البته طبق فرض، گراف مورد نظر تنک است و تعداد یال‌هایش

اندک هستند در واقع فرض شده است که تعداد یال‌ها از مرتبه

تعداد گره‌ها باشد: $E = \Theta(V)$ پس در این حالت داریم:

$$\text{گزینه (۴)} = O(V^2 \times V) = O(V^3)$$

ج: در مورد دکسترا، گفتیم که این الگوریتم فقط برای وزن‌های

مثبت به کار می‌رود بنابراین برای این سؤال مناسب

نیست زیرا در صورت سؤال گفته شده که وزن یال‌ها

می‌توانند منفی باشند. پس گزینه (۳) نادرست است.

با این حال برای آن که اطلاعات شما کامل شود،

این موارد را یادآوری می‌کنیم:

*۱) مرتبه زمانی اجرای دکسترا $O(V^2+E)$ است.

از آنجاکه همیشه $E \leq V^2$ است می‌توان گفت مرتبه‌ی دکسترا در حالت معمول آن $O(V^2)$ است.

*۲) اگر گراف G تُنُک باشه [تعریف دقیق گراف sparse آن است که تعداد یال‌ها $E = o(\frac{V^2}{\log V})$ باشه.

البته در تست فوق، طراح شرط کرده که منظورش از تُنُک، $E = \Theta(V)$ است اما در حالت کلی در مراجع رسمی، اگر $E = o(\frac{V^2}{\log V})$ باشه، کافیست که بگوئیم G، تُنُک است.]

به هرحال اگر $E = o(\frac{V^2}{\log V})$ باشه می‌توانیم یک پیاده‌سازی از دکسترا توسط binary-min-heap اجرا کنیم که از مرتبه زمانی $O((V+E)\log V)$ باشه.

که با جایگذاری $E = o(\frac{V^2}{\log V})$ می‌بینیم که در نهایت به مرتبه‌ی $o(V^2)$ می‌رسیم که نسبت به $O(V^2)$ بهتر است.

۴*) با استفاده از ساختار هیپ فیبوناتچی می‌توان مرتبه‌ی زمانی $O(V \log V + E)$ را هم برای دکستا به دست آورد.

این، بهترین مرتبه زمانی برای دکستا است.

۵: فلوید وارشال و جانسون:

الگوریتم هایی که برای گراف های با وزن حقیقی، به شکل تخصصی برای مسئله‌ی کوتاهترین مسیرها از هر گره به هر گره داریم، عبارتند از فلوید وارشال و جانسون.

مرتبه زمانی فلوید وارشال $O(V^3)$ است.

مرتبه زمانی جانسون $O(V^2 \log V + VE)$ است.

اگر طبق صورت سئوال، $E = \Theta(V)$ باشد، مرتبه‌ی زمانی جانسون برای گراف تنک به این صورت به دست می‌آید:

$$O(V^2 \log V + VV) = O(V^2 \log V)$$

جمع بندی و مقایسه‌ی گزینه‌ها

با توجه به فرض های مسأله داریم:

گزینه (۱): $O(V^2 \log V)$

گزینه (۲): $O(V^3)$

گزینه (۳): غیر قابل استفاده

گزینه (۴): $O(V^3)$

بنابراین گزینه (۱) بهترین پیشنهاد است.

ابتدا به چند مقدمه نیاز داریم.

مقدمه‌ی اول:

رابطه‌ی بازگشتی $F(n,k)=F(n-1,k-1)+F(n-1,k)$

یک رابطه بازگشتی معروف است و پاسخ آن عبارت است از: $F(n,k)=c.\binom{n}{k}$ مقدار ثابت c بستگی به مقادیر اولیه دارد. برای مثال اگر بخواهیم $F(n,0)=1$ باشد آنگاه $c=1$ است.

مقدمه‌ی دوم:

اگر به یک رابطه‌ی بازگشتی همگن، یک جمله‌ی ناهمگن ثابت هم اضافه کنیم، به جواب عمومی آن هم یک عدد ثابت مانند A افزوده می‌شود. (مگر آن که ریشه‌ی تکراری رخ دهد و A را در n ضرب کنیم.)

برای مثال، می دانیم که پاسخ رابطه‌ی بازگشتی همگن:

$$H(n) = 3\,H(n-1)$$

به صورت $c \times 3^n$ است. حالا اگر بخواهیم رابطه‌ی بازگشتی ناهمگن:

$$H(n) = 3\,H(n-1) + 4 \quad (*)$$

را حل کنیم، از آنجا که جمله‌ی ناهمگن، یک عدد ثابت است می گوییم جواب معادله، از دو قسمتی تشکیل می شود:

$$H(n) = c \times 3^n + A$$

برای یافتن A باید حالت $c=0$ را در نظر بگیریم یعنی (موقتاً) فرض کنیم $H(n)=A$ حالا این جواب را در معادله‌ی $(*)$ قرار دهید:

$$A = 3A + 4 \quad \Rightarrow \quad A = -2$$

تا اینجا می دانیم که:

$$H(n) = c \times 3^n - 2$$

برای یافتن مقدار ثابت c نیاز به مقادیر اولیه مانند $H(1)$ یا $H(0)$ داریم.

۴۶/۴

اکنون به پاسخ تست ۴۶ می پردازیم:

فرض کنید $T(n,K)$ نشان دهنده‌ی تعداد دفعاتی باشد که تابع Comb خودش را صدا می زند.

در حالت هایی که $K=0$ یا $n=K$ باشد می دانیم که فقط یک بار خودش را صدا می زند در واقع فقط با یک حرکت، محاسبه تمام می شود:

$$T(n,0)=1$$

اما در حالت کلی (برای سایر حالات) وقتی این الگوریتم را برای (n,K) اجرا می کنیم، اولاً در همین ابتدا می خواهیم $Comb(n,k)$ را صدا بزنیم در ضمن طبق رابطه‌ی بازگشتی به کار رفته، تعداد دفعاتی که همین تابع را برای $(n-1,k-1)$ صدا می زنیم، به علاوه‌ی تعداد دفعاتی که همین تابع را برای $(n-1,K)$ صدا می زنیم باید محاسبه شوند. در نتیجه داریم:

$$(*)\quad T(n,K) = T(n-1,K-1) + T(n-1,K) + 1$$

حالا این رابطه‌ی بازگشتی را حل می‌کنیم.

با سخ معادله‌ی همگن $T(n,K) = T(n-1,K-1) + T(n-1,K)$ را می‌دانیم. (به عنوان یک نمونه‌ی معروف، این رابطه را می‌شناسیم.)

جواب همگن: $T(n,K) = c \cdot \binom{n}{K}$

در ضمن جمله‌ی ثابت 1 در (*) باعث می‌شود که عدد ثابت A را هم به جواب همگن اضافه کنیم پس:

جواب عمومی رابطه‌ی (*):

$$T(n,K) = c \cdot \binom{n}{K} + A$$

برای یافتن A موقتاً $c=0$ را در نظر می‌گیریم و $T(n,K)=A$ را در (*) قرار می‌دهیم:

$$A = A + A + 1 \implies A = -1$$

۴۶.۵

حالا می دانیم که:

$$T(n,k) = c\binom{n}{k} - 1$$

در ضمن گفتیم که برای حالت $k=0$ همچنین حالت $n=k$ پیچیدگی محاسباتی $T(n,k)=1$ است. پس:

$$T(n,0) = c\binom{n}{0} - 1 = 1$$

$$\Rightarrow c-1=1 \quad \Rightarrow \quad c=2$$

بنابراین:

برای محاسبه‌ی $Com(n,k)$، پیچیدگی محاسباتی این الگوریتم (تعداد صدا زدن‌های خودش) برابر است با:

$$T(n,k) = 2\binom{n}{k} - 1$$

به ازای $n=20$, $k=3$ داریم:

$$2\binom{20}{3} - 1 = 2279$$

۴۷ پاسخ: گزینه (۳)

با توجه به اجزایی که در صورت سئوال وجود دارد، ابتدا همه‌ی مرتبه‌های زمانی موجود برای حل مساله‌ی شار بیشینه را مرور کنیم.

۱) در روش Ford-Fulkerson ، زمان اجرای الگوریتم به این بستگی دارد که پس از تشکیل گراف G_f و شبکه‌ی پیمانه‌ها، مسیر تکمیلی p را که از s به t (منبع به چاه) می‌رسد چگونه پیدا و انتخاب می‌کنیم.

اگر انتخاب این مسیر، هر بار به صورت دلخواه انجام شود، زمان اجرای الگوریتم $O(mf^*)$ خواهد بود. f^* در اینجا برابر است با ماکزیمم جریان پیدا شده توسط الگوریتم.

علت آن است که با هر بار تکرار یافتن مسیر تکمیلی، حداقل یک واحد به جریان به دست آمده تا آن مرحله، اضافه می‌کنیم. پس مثلاً اگر $f^*=100$ باشد، حداکثر بعد از ۱۰۰ بار افزایش دادن جریان به ماکزیمم آن خواهیم رسید.

البته در هر تکرار ناچاریم روی یال‌های مسیر p تغییراتی را اعمال کرده و سپس برای همه یال‌ها، شبکه‌ی پیمانه را حساب کنیم.

۴۷٫۲

به همین دلیل f^* مرتبه و در هر مرتبه E محاسبه داریم:

$$O(E f^*) = O(m f^*)$$

حالا اگر تصور کنید که در گرافی با n گره و m یال، ظرفیت یال‌ها آنقدر زیاد باشد که ماکزیمم جریان بتواند به حد $f^* = 2^m$ برسد، واضح است که مرتبه زمانی این روش، چنین جمله‌ای نخواهد بود.

۲) برای بهبود کران بالای مرتبه زمانی در روش Ford-Fulkerson پیشنهاد می‌شود که هر بار برای یافتن مسیر تکمیلی p در گراف G_f، به جای انتخاب دلخواه یک مسیر، از جستجوی BFS استفاده کنیم. در این جستجو، کوتاهترین مسیر بر حسب تعداد یال‌ها را از s به t می‌توانیم در مرتبه‌ی زمانی $O(E' + V)$ به دست آوریم. البته در شبکه‌ی پیمانه، تعداد یال‌ها، حداکثر به اندازه‌ی ۲ برابر یال‌های گراف G است $E' \leq 2E$ و در ضمن گراف مورد نظر ما در مسأله‌ی حداکثر شار، گرافی همبند ضعیف است پس $E \geq V-1$ است. در نتیجه پیمایش BFS از مرتبه‌ی

$$O(E' + V) = O(E)$$

انجام می‌شود. پس با استفاده از BFS برای یافتن مسیر P، هر بار به اندازه‌ی $O(E)$ زمان صرف یافتن مسیر P می‌شود.

از طرفی ثابت می‌شود (نه به سادگی) که اگر مسیرها را با BFS انتخاب کنیم، تعداد کل مراحل اجرای الگوریتم حداکثر VE خواهد بود. به عبارتی در هر بار اجرا، آنقدر مقدار جریان بهبود پیدا می‌کند که پس از VE مرتبه، با اطمینان به f^* برسیم. این روش را که همان روش Ford-Fulkerson با استفاده از BFS است، به نام الگوریتم Edmonds-Karp می‌شناسیم و مرتبه‌ی زمانی آن چنین به‌دست می‌آید:

$$O(VE \times E) = O(VE^2) = O(nm^2)$$

۳) یک نوع پیاده‌سازی دیگر برای یافتن جریان حداکثری هنگامی که ظرفیت‌ها عدد صحیح مثبت باشند وجود دارد که در آن از برچسب‌دهی (برای افزایش ارتفاع گره‌ها) استفاده می‌کنیم و مرتبه زمانی آن $O(V^2E)$ است یعنی $O(n^2m)$ که نسبت به nm^2 بهتر است:

الگوریتم push-relabel (راندن و برچسب‌دهی):

$O(n^2m)$

۴) اکنون به روش اصلی Ford-Fulkerson بازگردیم. گفتیم که در این روش، انتخاب دلخواه مسیر تکمیلی p، باعث می‌شود مرتبه‌ی زمانی حداکثر $O(mf^*)$ به دست آید و اگر f^* نهایی باشد، این مرتبه، چند جمله‌ای برحسب n, m نیست.

علت ایجاد f^* آن بود که وقتی مسیر p به شکل دلخواه انتخاب می‌شود، ممکن است با هر بار اجرای این عمل، فقط یک واحد به جریان کل، اضافه شود تا پس برای رسیدن به f^*

ممکن است ناچار شویم f^* بار این کار را تکرار کنیم.

برای رفع این مشکل، راهکاری وجود دارد. آن هم این است که مسیری را پیدا کنیم که ظرفیت خالی آن حداقل K باشد. به این صورت اطمینان داریم که با به‌روزرسانی جریان‌ها روی این مسیر، حداقل K واحد به جریان افزوده می‌شود.

در ابتدا $k = C$ یعنی ماکزیمم ظرفیت یال‌هاست.
و پس اگر مسیری با این ظرفیت پیدا نکردیم آن را نصف می‌کنیم. برای مثال از $K=8$ آغاز می‌کنیم و اگر مسیری وجود نداشت که بتواند ۸ ظرفیت جدید ایجاد کند، به دنبال مسیری می‌گردیم که حداقل ظرفیت ۴ را ایجاد کند. اگر آن را نیافتیم، به نصف کردن K ادامه می‌دهیم.

حالا در مورد این روش دو مطلب را باید در کنار هم قرار دهیم:
(این روش را روش جریان‌دهی با مقیاس‌دهی (scaling) می‌نامند.)

درمورد این روش:

الف) برای یافتن مسیر P از BFS کمک می‌گیریم پس مرتبه‌ی زمانی آن حداکثر $O(VE^2)$ است.

ب) با بدرسی این الگوریتم و به این علت که هر بار $K=C$ را لغف می‌کنیم، یک مرتبه‌ی زمانی دیگر نیز به صورت $O(E^2 \log C)$ به دست می‌آید.

به عبارتی مرتبه زمانی این روش هم از $O(VE^2)$ است و هم از $O(E^2 \log C)$ است. پس هرکدام که کمتر باشد درواقع بهترین کران برای مرتبه‌ی زمانی خواهد بود.

با این توضیحات متوجه می‌شویم که منظور طراح سؤال از دادن مرتبه‌ی زمانی $O(m^2 \log C)$ اثر، به روش مقیاس دهی بوده است که علاوه بر این مرتبه، یک سقف دیگر هم به صورت $nm^2 = VE^2$ دارد. پس از مرتبه‌ی حیله‌ای است.

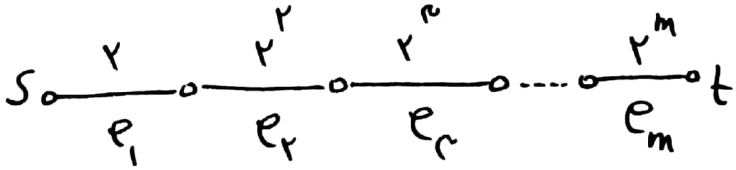
۴۷/۷

توجه کنید:

۱) مسئله، استاندارد علمی ندارد زیرا ممکن است الگوریتم‌های دیگری هم باشند که مرتبه زمانی آنها $O(m^2 \log c)$ باشد. بهتر بود طراح سئوال دقیقاً نام الگوریتم‌ها را ذکر می‌کرد.

۲) شرط صحیح بودن ظرفیت یال‌ها بسیار مهم است. اگر ظرفیت یال‌ها اعداد گنگ باشند، ممکن است، روش فورد، هرگز به مقدار حداکثر جریان نرسد.

۳) در مورد $O(mnc)$ توجه کنید که c یعنی حداکثر ظرفیت یال‌ها ممکن است $c=2^m$ باشد. مثلاً فرض کنید ظرفیت یال e_k را به صورت 2^k تعریف کرده باشیم.



در این مرتبه، به وضوح ممکن است نمایی باشد.

در مورد $O(m^2 \log c)$ هم اگر نمی‌دانستیم که یک سقف دیگر مانند $O(nm^2)$ دارد می‌توانستیم مثال نقض بزنیم.

۴۸ پاسخ: گزینه (۳)

روش اول: طبق تعریف CLRS از جستجوی DFS، پس از آن که این جستجو در یکی از مولفه‌های همبند ضعیف گراف G به پایان رسید، در صورتی که هنوز گره‌هایی باشند که ملاقات نشده‌اند، از یکی از آن‌ها به دلخواه آغاز می‌کند و DFS را در مولفه‌ی دیگر ادامه می‌دهد. به این ترتیب واضح است که گره‌ی که بعد از همه‌ی گره‌ها از پشته خارج می‌شود گره‌ی بوده است که ورودی نداشته اما احتمالاً خروجی داشته است.

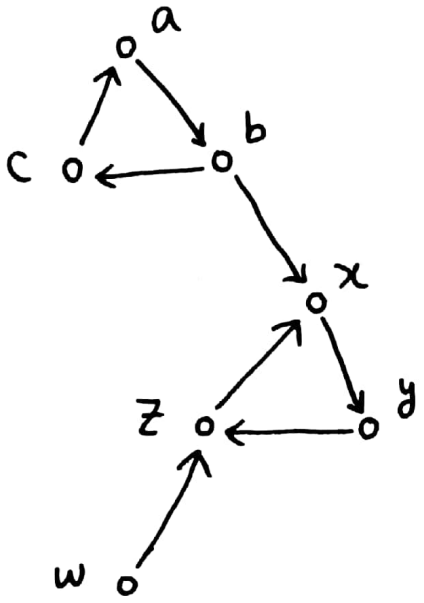
دقت کنید عکس این مطلب صحیح نیست. اگر گره u ورودی نداشته باشد، دلیلی ندارد که بیشترین مقدار f_u را داشته باشد. اما اگر بیشترین مقدار f_u را داشته باشد، حتماً گره‌ی بوده که ورودی نداشته است.

البته برای درک بهتر شما، همین مطلب را با مثال توضیح می‌دهیم.

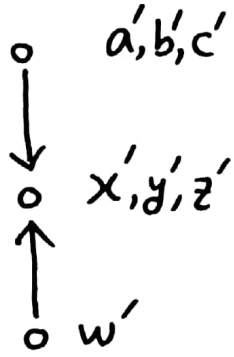
روش دوم:

بررسی و توضیح با توجه به چند مثال:

برای آن که مثال خوبی داشته باشیم ابتدا گراف را در نظر گیرید که ۴ مولفه‌ی همبندی قوی داشته باشد:



G



G'

توجه کنید که در گراف G، گره‌های a، b و c یک مولفه‌ی همبندی قوی هستند یعنی از هرکدام به هرکدام مسیر وجود دارد. این ها با هم یک گره از G' را تشکیل داده‌اند که آن گره را با a'=b'=c' نشان می‌دهیم. (طبق توضیحات صورت سئوال)

به همین ترتیب x و y و z هم یک مولفه‌ی همبندی قوی است و با هم یک گره از G' را تشکیل می‌دهند. w هم تنها است و گره w' نماینده آن است.

فرض کنید در گراف G از مولفه‌ی $\{a,b,c\}$ به مولفه‌ی $\{x,y,z\}$ مسیری یک طرفه وجود دارد، به همین دلیل در گراف G' یالی از a' به x' وجود دارد. به همین ترتیب یالی از w به x' وجود دارد.

بیا رخوب! حالا DFS را از گره a اجرا کنید.

لحظه به لحظه این اتفاق‌ها رخ می‌دهد:

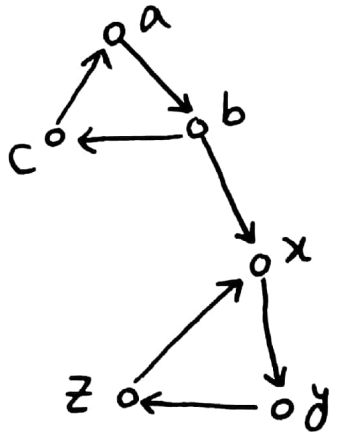
time	رخداد	
۱	ورود به a	دقت کنید که طبق تعریف CLRS اگر روال DFS از یک مولفه خارج شد و هنوز گره‌ها باقی مانده بود به دلخواه از یک گره باقی مانده آغاز می‌کند. در ضمن ما اولویت را ترتیب حروف الفبا فرض کرده‌ایم.
۲	ورود به b	
۳	ورود به c	
۴	خروج از c	
۵	ورود به x	
۶	ورود به y	
۷	ورود به z	
۸	خروج از z	
۹	خروج از y	
۱۰	خروج از x	
۱۱	خروج از b	
۱۲	خروج از a	
۱۳	ورود به w	
۱۴	خروج از w	

بنابراین در این گراف این w است که هم بیشترین مقدار S و هم بیشترین مقدار f را دارد.

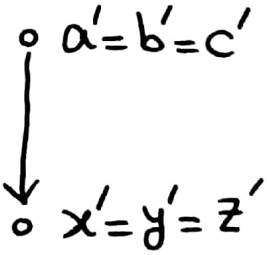
$$\text{زمان ورود } w = S_w = ۱۳$$
$$\text{زمان خروج } w = f_w = ۱۴$$

و اگر دقت کنیم می بینیم که در گراف G'، گره w' ورودی ندارد اما خروجی دارد. بنابراین گزینه های (۲) و (۴) نادرست هستند.

حالا فرض کنید w در G قرار نداشت. در این صورت G' هم w' را نداشت:



G



G'

حالا با شروع از a در DFS لحظه به لحظه این رخدادها را خواهیم داشت،

	رخداد	time
	ورود a	۱
	ورود b	۲
	ورود c	۳
	خروج c	۴
	ورود x	۵
	ورود y	۶
*	ورود z	۷
	خروج z	۸
	خروج y	۹
	خروج x	۱۰
	خروج b	۱۱
*	خروج a	۱۲

پس در این گراف، گره a بیشترین مقدار f_a را دارد.
گره z بیشترین مقدار S_a را دارد.

دقت کنید که گزینه‌های (۲) و (۴) قبلاً رد شده‌اند.

۴۸,۶

گره Z دارای ورودی است پس گزینه (۱) نادرست است.

اما گزینه (۳) بازهم صحیح است. گره A که بیشترین مقدار f را

دارد، ورودی ندارد.

پیروز و سربلند باشید،

ابوالفضل گیلک.

گروه بابان