بسم الله الرحمن الرحیم

پاسخ سوالات

ساختمان داده و طراحی الگوریتم

کارشناسی ارشد ۹۹

مهندسی فناوری اطلاعات

ابوالفضل گیلک

گروه بابان

@abolfazlgilak

«اگر دانشگاه اصلاح شود مملکت اصلاح می‌شود.»

امام خمینی (ره)

عصر پنج‌شنبه

۱۳۹۹/۵/۲

جمهوری اسلامی ایران

وزارت علوم، تحقیقات و فنّاوری

سازمان سنجش آموزش کشور

آزمون ورودی دوره‌های کارشناسی‌ارشد ناپیوستهٔ داخل ـ سال ۱۳۹۹

مجموعه مهندسی فناوری اطلاعات (IT) ـ کد (۱۲۷۶)

۳۷- کدام گزینه درست است؟ (دقت کنید که در زیر از حرف O کوچک استفاده شده است.)

۱) $n! = o(n^n)\ ,\ \log n! = \theta(\log n^n)$

۲) $n! = o(n^n)\ ,\ \log n! = o(\log n^n)$

۳) $n! = \theta(n^n)\ ,\ \log n! = \theta(\log n^n)$

۴) $n! = o(n^n)\ ,\ \log n! = \omega(\log n^n)$

پاسخ: گزینه (۱)

طبق قانون استرلینگ برای n های بزرگ ، سرعت رشد $n!$ اکیداً از n^n کمتر است.

اما لگاریتم هردوی آنها از مرتبه‌ی برابری است. به اثبات این مطالب توجه کنید:

فرمول استرلینگ (هم ارزی استرلینگ):

$$n! \simeq \left(\frac{n}{e}\right)^n \sqrt{n}\,\sqrt{2\pi} \qquad n \to \infty$$

$$\Rightarrow \quad n! \simeq n^n \frac{\sqrt{n}}{e^n} \sqrt{2\pi} < n^n$$

$$\Rightarrow \quad n! = o(n^n)$$

با استفاده از همین هم ارزی و با توجه به قانون ضرب داریم:

$$\log(n!) \simeq n\log n - n\log e + \frac{1}{2}\log n + \frac{1}{2}\log 2\pi$$

$$= \theta(n\log n) = \Theta(\log n^n)$$

۳۸- وضعیت جدول درهم‌سازی $H[0..9]$ بعد از درج هفت عدد $S_1, ..., S_7$ به‌صورت زیر است:

$$H[0..9] = [S_7 , S_1 , \square , S_4 , S_2 , S_5 , \square , S_6 , S_3]$$

که $\square$ نشان‌دهندهٔ خانهٔ خالی است. برای درهم‌سازی از روش درهم‌سازی باز با وارسی خطی استفاده شده است. برای جستجوی عنصری که در جدول نیست حداکثر چند مقایسه باید انجام شود؟ (دقت کنید چک کردن آن‌که یک خانه خالی است خود به یک مقایسه نیاز دارد.)

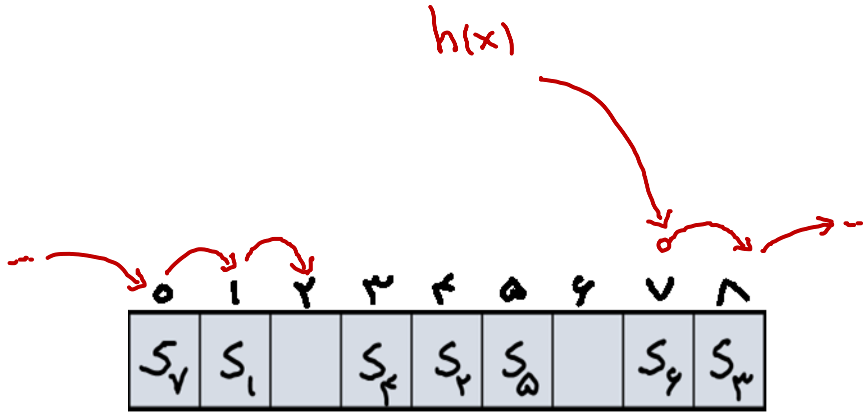
۱) ۲
۲) ۳
۳) ۴
۴) ۵

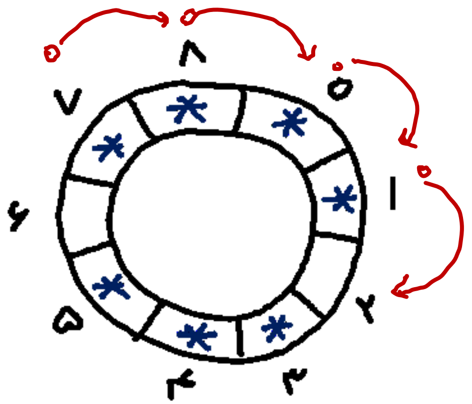
پاسخ: گزینه (۴)

فرض کنیم x عضوی باشد که در جدول درج نشده است.

برای جستجوی x در این جدول، ابتدا به آدرس $h(x)$ می‌رویم. اگر x در آن آدرس موجود نبود طبق وارسی خطی باید به آدرس بعدی برویم. این جستجو تا جایی که به یک آدرس خالی برسیم و متوجه شویم x در جدول قرار ندارد ادامه می‌یابد. بیشترین تعداد مقایسه در این جدول هنگامی رخ می‌دهد که $h(x) = 7$ باشد.

در این حالت آدرس‌های ۷، ۸، ۰، ۱ و ۲ را باید بررسی کنیم.





۳۹- چند درخت دودویی جست‌وجوی متفاوت با n گره و برچسب‌های ۱ تا n وجود دارد، به‌طوری که پیمایش پیش‌ترتیب و میان‌ترتیب آن‌ها یکسان باشد؟

۱) ۰

۲) ۱

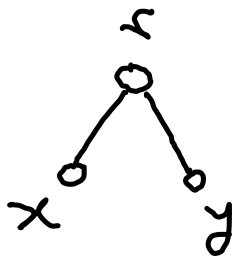
۳) $n!$

۴) عدد n ام کاتالان

پاسخ: گزینه (۲)

pre order: r x y

In order: x r y



اگر بخواهیم این پیمایش‌ها یکسان شوند، لازم است که $x=null$ باشد یعنی زیردرخت چپ تهی باشد. این موضوع در تمام گره‌های میانی تکرار شود.

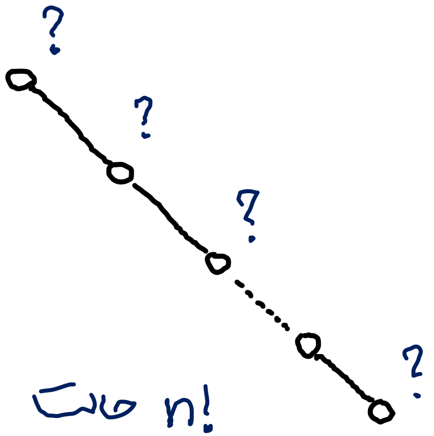
حالا اگر T یک درخت دودویی باشه، با توجه به انواع برچسب گذاری

$1, 2, \ldots, n$ تعداد $n!$ حالت خواهیم داشت.

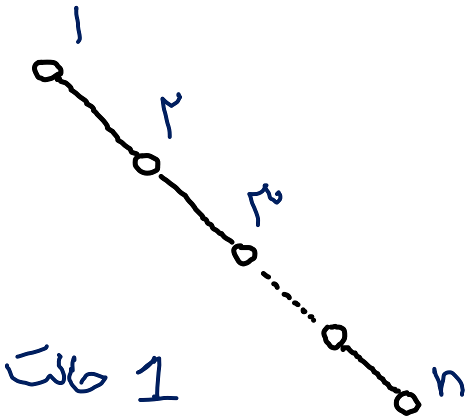
اما اگر قرار باشه T درخت جستجوی دودویی BST باشه،

ترتیب مهم است و باید فرزند راست ارزشش بزرگ تر باشه و

فقط یک حالت برای T داریم.



$\Longrightarrow$



۴۰- اعداد ۱ تا ۵۰۰ را در یک درخت دودویی جست‌وجو ذخیره کرده‌ایم. می‌خواهیم عدد ۱۹۳ را در این درخت جست‌وجو کنیم. کدام دنباله <u>نمی‌تواند</u> مسیر جست‌وجو برای عدد ۱۹۳ باشد؟

۱) ۴ , ۲۰ , ۳۰ , ۵۵ , ۱۰۱ , ۱۰۲ , ۱۰۵ , ۱۷۷ , ۱۹۳

۲) ۵ , ۴۵۴ , ۳۰۰ , ۱۰۰ , ۲۵۰ , ۱۵۰ , ۲۰۰ , ۱۹۳

۳) ۴۳۷ , ۱۵۷ , ۲۳۷ , ۲۲۱ , ۲۰۱ , ۱۴۳ , ۱۹۰ , ۱۹۳

۴) ۵۰۰ , ۴۰۰ , ۳۰۰ , ۲۰۰ , ۵۰ , ۱۰۰ , ۱۵۰ , ۱۹۳

پاسخ : گزینه (۳)

کلید مورد جستجوی ما $x = ۱۹۳$ است. وقتی x از $a_۲ = ۱۵۷$ بزرگتر است، باید در ادامه، اعدادی که ملاقات می‌کنیم همگی در سمت راست $a_۲$ باشند یعنی از $a_۲$ بزرگتر باشند اما ۱۴۳ اینطور نیست.

توضیحات تکمیلی :

فرض کنید دنباله اعداد ملاقات شده در طی جستجوی کلید x به این صورت باشد :

$$a_1 ، a_2 ، \ldots ، a_n$$

با توجه به ویژگی جستجوی دودویی باید برای هر a_i این شرط برقرار باشد :

$$\text{if} \quad a_i > x \quad \Longrightarrow \quad \forall j > i \quad a_i \geq a_j$$

$$\text{if} \quad a_i < x \quad \Longrightarrow \quad \forall j > i \quad a_i \leq a_j$$

راه دوم: (الگوریتم تشخیص دنباله‌هایی که می‌توانند دنباله‌ی ملاقات‌ها در جستجوی دودویی باشند).

از انتها یعنی a_n به ابتدا حرکت می‌کنیم. رکوردهای max و min را ثبت می‌کنیم.

در هر مرحله:

$$M = \max(a_{i+1}, a_{i+2}, \dots, a_n)$$
$$m = \min(a_{i+1}, a_{i+2}, \dots, a_n)$$

نباید a_i در بازه‌ی (m, M) قرار داشته باشد.

زمان اجرای این الگوریتم $O(n)$ است.

در گزینه (۳) هنگامی که از انتها به ابتدا حرکت می‌کنیم :

۴۳۷ ۱۵۷ ۲۳۷ ۲۳۱ ۲۰۱ ۱۴۳ ۱۹۰ ۱۹۳

$M = max = ۲۳۷$

$m = min = ۱۴۳$

?

عدد ۱۵۷ بین m و M قرار دارد

$۱۴۳ < ۱۵۷ < ۲۳۷$

پس این دنباله نمی‌تواند جواب باشد.

۴۱- آرایه $A[1..13]$ شامل ۱۳ عدد صحیح را در نظر بگیرید. می‌توانیم هر بار دو خانهٔ دلخواه از این آرایه را با هم جابه‌جا کنیم. کدام گزینه را <u>نمی‌توان</u> با حداکثر یک بار جابه‌جایی به هرم بیشینه تبدیل کرد؟

۱) $A[1..13] =$ ۸۹ , ۱۹ , ۷۰ , ۱۷ , ۱۲ , ۴۰ , ۲ , ۵ , ۷ , ۱۱ , ۶ , ۹ , ۱۰

۲) $A[1..13] =$ ۸۹ , ۱۹ , ۴۰ , ۱۷ , ۱۲ , ۷۰ , ۲ , ۵ , ۷ , ۱۱ , ۶ , ۹ , ۱۰

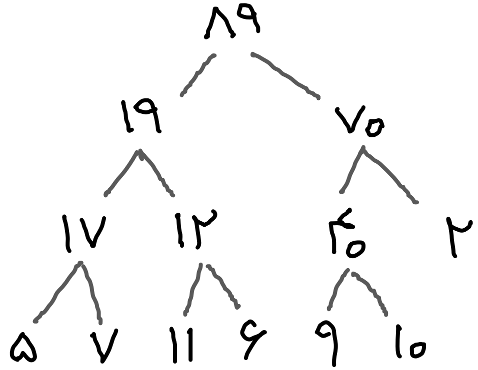
۳) $A[1..13] =$ ۸۹ , ۱۹ , ۷۰ , ۱۷ , ۲ , ۴۰ , ۱۲ , ۵ , ۷ , ۱۱ , ۶ , ۹ , ۱۰

۴) $A[1..13] =$ ۸۹ , ۱۹ , ۴۰ , ۱۷ , ۱۲ , ۱۰ , ۲ , ۵ , ۷ , ۱۱ , ۶ , ۹ , ۷۰

پاسخ : گزینه (۴)

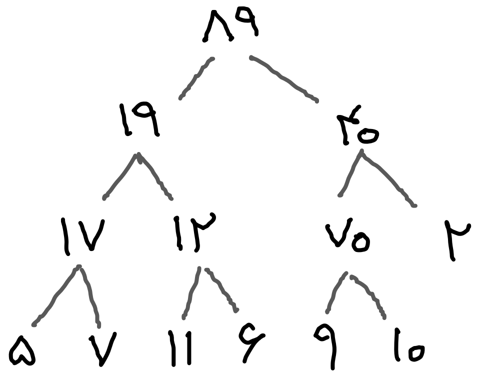
اعداد موجود در آرایه در یک هرم (با پر کردن سطوح از چپ به راست) نوشته شوند:

گزینه یک:



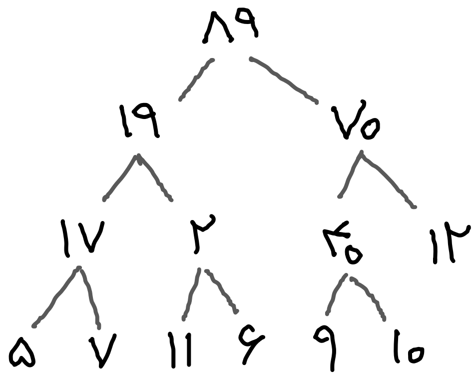
یک هرم بیشینه است.

گزینه دو:



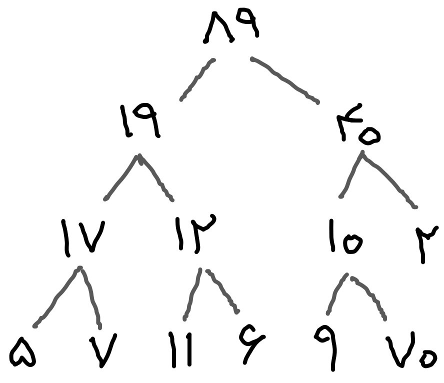
فقط به جابه‌جایی ۷۵ با ۴۵ نیاز دارد.

گزینه ۳:



فقط به جابه‌جایی ۱۱ و ۲ نیاز دارد.

گزینه (۴):



به دو جابه‌جایی نیاز داریم: ۷۵ و ۱۵ جابه‌جا شوند
سپس ۷۵ و ۴۵ جابه‌جا شوند.

درواقع ۷۵ برای رسیدن به محل مناسب خود به حداقل
۲ جابه‌جایی نیاز دارد.

۴۲- به‌ازای چند تا از الگوریتم‌های مرتب‌سازی زیر، پیچیدگی زمانی حالت متوسط، بـدترین حالـت و بهتـرین حالـت یکسان است؟

- مرتب‌سازی سریع
- مرتب‌سازی ادغامی
- مرتب‌سازی شمارشی
- مرتب‌سازی درجی

۱) ۰

۲) ۱

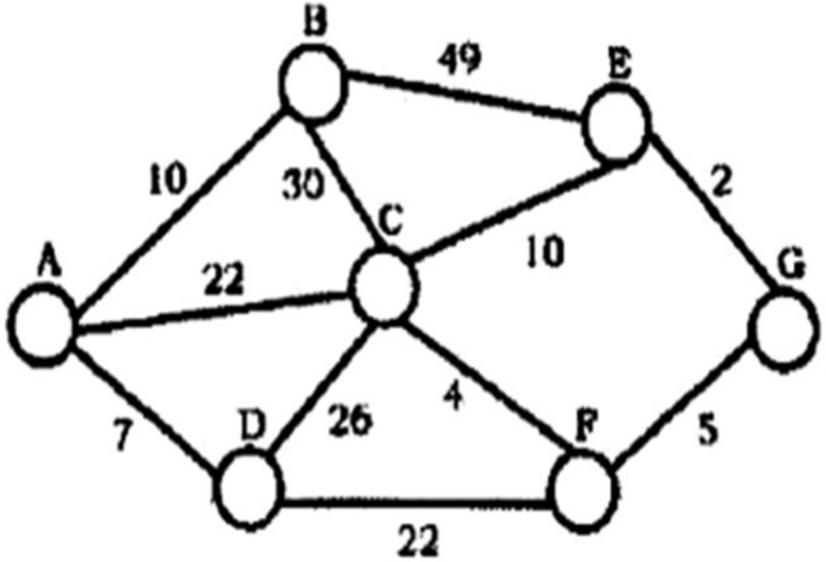
۳) ۲

۴) ۳

پاسخ: گزینه (۳)

	بهترین حالت	حالت متوسط	بدترین حالت
مرتب سازی سریع	$n\log n$	$n\log n$	n^2
مرتب سازی ادغامی	$n\log n$	$n\log n$	$n\log n$
مرتب سازی شمارشی	$n+m$	$n+m$	$n+m$
مرتب سازی درجی	n	n^2	n^2

۴۳- در گراف زیر، الگوریتم پریم را با شروع از رأس A اجرا کرده‌ایم. کدام ترتیب زیر (از چپ به راست) می‌تواند ترتیب اضافه شدن یال‌ها به درخت پوشای کمینه باشد؟

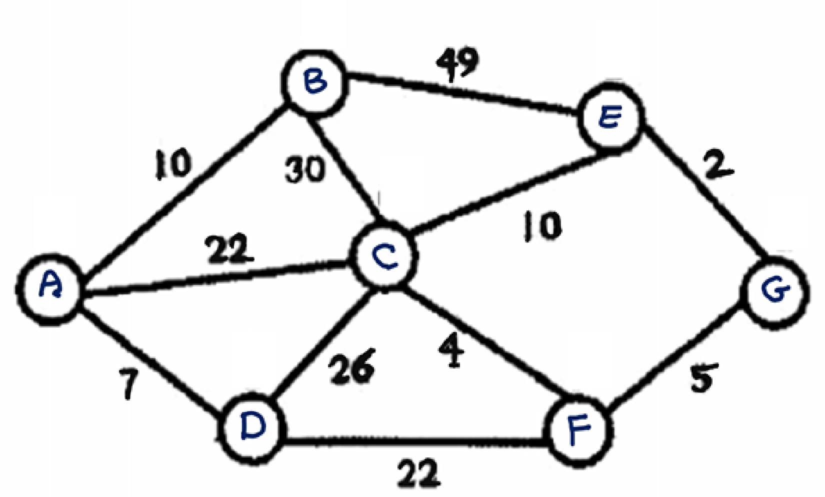


۱) (A, D),(A, B),(D, F),(F, C),(F, G),(G, E)

۲) (A, B),(A, D),(D, F),(F, G),(G, E),(F, C)

۳) (A, D),(A, B),(A, C),(C, F),(G, E),(F, G)

۴) (E, G),(C, F),(F, G),(A, D),(A, B),(A, C)



پاسخ: گزینه (۱)

دو ترتیب مختلف برای انتخاب یال‌ها در پریم با شروع از A وجود دارد:

حالت اول: از چپ به راست:

(A,D) ، (A,B) ، (A,C) ، (C,F) ، (F,G) ، (G,E)

حالت دوم: از چپ به راست:

(A,D) ، (A,B) ، (D,F) ، (F,C) ، (F,G) ، (G,E)

۴۴- در گراف همبند، بدون جهت و بدون وزن G، الگوریتم دایکسترا را با شروع از رأس S اجرا می‌کنیم. در هر گام از الگوریتم دایکسترا یک رأس مختومه می‌شود. به این معنی که طول کوتاهترین مسیر به آن رأس محاسبه می‌شود. حال ترتیبی که رئوس در الگوریتم دایکسترا مختومه شده‌اند را در نظر بگیرید. درخصوص گزاره‌های زیر کدام گزینه صحیح است؟

(الف) همیشه یک ترتیب BFS از رئوس وجود دارد که با ترتیب مختومه شدن رئوس در الگوریتم دایکسترا یکسان است.

(ب) همیشه یک ترتیب DFS از رئوس وجود دارد که با ترتیب مختومه شدن رئوس در الگوریتم دایکسترا یکسان است.

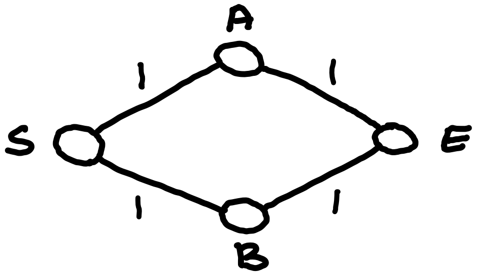
۱) (الف) درست، (ب) درست

۲) (الف) درست، (ب) نادرست

۳) (الف) نادرست، (ب) درست

۴) (الف) نادرست، (ب) نادرست

پاسخ: گزینه (۲)



اصولاً نحوه‌ی مشخص شدن گره‌ها در دکسترا در یک گراف غیروزن دار
مانند ملاقات گره‌ها در BFS است.

دکسترا با مبدأ S : S , A , B , E

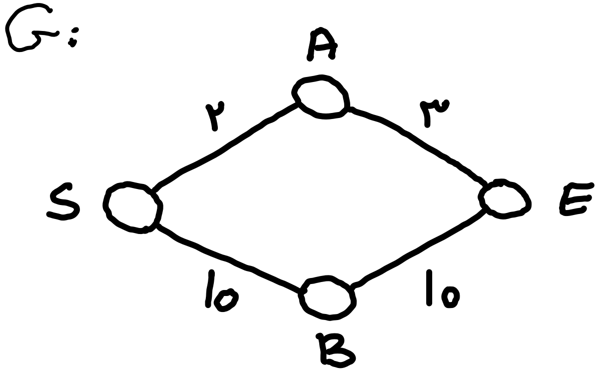
BFS(S) : S, A, B, E

DFS(S) : S, A, E, B

توجه کنید:

(*) در گراف وزن‌دار هر دو جمله‌ی الف و ب نادرست هستند:

برای هر دو ادعای بالا مثال نقض وجود دارد.



در گراف G با شروع از مبدأ S:

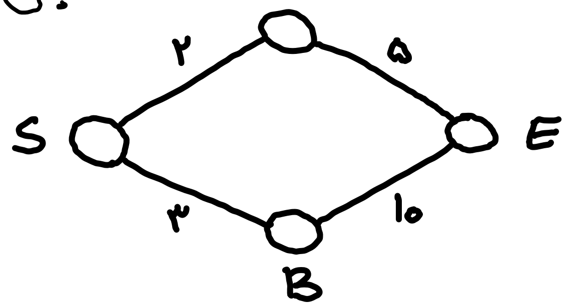
ترتیب مختومه شدن گره‌ها در دکسترا: از چپ به راست:

S, A, E, B

زیرا پس از relax کردن یال AE، مقدار d(E) از d(B) کمتر است.

اما ترتیب در BFS به این صورت است که: ابتدا S، پس A و B، و در پایان E.

G':



اکنون به گراف G' توجه کنید:

این بار با توجه به وزن یال‌ها، ترتیب مختوم‌شدن در دکسترا

از چپ به راست چنین است: S, A, B, E

در حالی که با اجرای DFS(S) به هر صورتی که اولویت‌ها را در نظر بگیریم، گره E آخرین گره

نخواهد بود. ترتیب DFS یا به صورت S, A, E, B یا به صورت S, B, E, A است.

۴۵- فرض کنید در یک گراف همبند و بدون جهت G، الگوریتم جست‌وجوی سطح اول را با شروع از رأس r اجرا کنیم. فرض کنید u و v دو رأس دلخواه و متمایز G به‌غیر از r باشند. همچنین فرض کنید $d(r,u)$ و $d(r,v)$ طول کوتاه‌ترین مسیر از r به u و v باشند. اگر u قبل از v در جست‌وجوی سطح اول ملاقات شده باشد، کدام گزینه صحیح است؟

۱) $d(r,u) \le d(r,v)$

۲) $d(r,u) < d(r,v)$

۳) $d(r,u) > d(r,v)$

۴) هیچ‌یک از موارد صحیح نیست.

توضیح: اگر فرض کنیم گراف G وزن دار (با وزن دلخواه) و منظور از $d(r,u)$ فاصله‌ی u از r در گراف G (نه در درخت BFS) است، گزینه (۴) صحیح خواهد بود. اما با فرض آن که گراف، گراف ساده باشد و $d(r,u)$ همان کوتاه‌ترین مسیر از r به u در BFS باشد مسأله را حل می‌کنیم و به پاسخ گزینه (۱) می‌رسیم. کلید نهایی گزینه (۱) بوده است.
(چون صحبتی از وزن یال‌ها نشده است، گراف غیروزن‌دار قابل فرض است.

پاسخ: گزینه (۱)

می دانیم که در گراف ساده غیروزن دار، مسیری که در درخت $BFS(r)$ از ریشه تا u به دست می آید

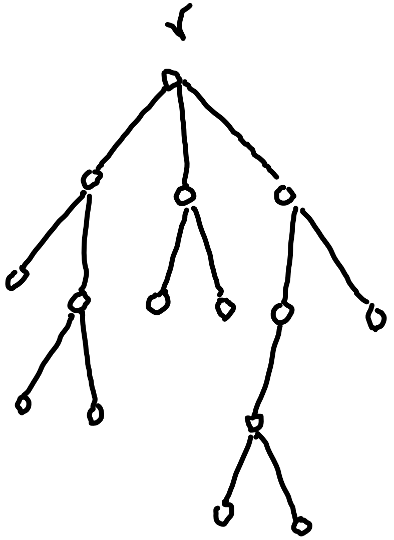
همان کوتاه ترین مسیر از r تا u است. در این حالت:

$d(r,u) =$ عمق گره u در درخت BFS

اگر u قبل از v ملاقات شده است پس

عمق u در این درخت، کمتر یا مساوی v است.

پس $d(r,u) \le d(r,v)$



۴۶- فرض کنید یک کوله‌پشتی با ظرفیت ۱۰۰ داریم. تعدادی شیٔ با حجم و ارزش صحیح داده شده است. می‌خواهیم تعدادی از این اشیاء را در کوله بگذاریم، طوری‌که اولاً در کوله جا شوند و ثانیاً مجموع ارزش‌شان بیشینه شود. الگوریتم حریصانه زیر را در نظر بگیرید.

نزولی

اشیاء را به‌صورت صعودی براساس ارزش روی حجم مرتب می‌کنیم. اشیاء را به‌ترتیب لیست فوق مورد بررسی قرار داده و اگر فضای خالی کوله‌پشتی حداقل به اندازهٔ شیٔ مورد بررسی بود آن شیٔ را درکوله قرار می‌دهیم.

به‌ازای کدام ورودی زیر الگوریتم حریصانه جواب بهینه برنمی‌گرداند؟ (در زیر حجم و ارزش اشیاء به‌ترتیب در آرایه‌های V و W آمده است. یعنی شیٔ iام دارای حجم V[i] و ارزش W[i] است.)

۱) $V[1..5]=[76,25,25,25,25]$ ، $W[1..5]=[3,1,1,1,1]$

۲) $V[1..5]=[70,30,45,25,40]$ ، $W[1..5]=[20,5,9,10,1]$

۳) $V[1..5]=[80,25,25,25,25]$ ، $W[1..5]=[80,20,22,21,20]$

۴) همهٔ موارد فوق

پاسخ: این سوال از آزمون حذف شده است.

اما اگر صورت سوال را اصلاح کنیم و به‌جای واژه‌ی صعودی از نزولی استفاده کنیم گزینه (۳) جواب است.

در گزینه (۱)، با انتخاب چهار اولویت اول

یعنی چهار شی با حجم ۲۵ کوله پشتی پر می‌شود

و بهترین حالت نیز همین است. مجموع ارزش

اشیاء در کوله پشتی به ۴ می‌رسد که بهینه است.

i	۱	۲	۳	۴	۵
w_i	۳	۱	۱	۱	۱
v_i	۷۶	۲۵	۲۵	۲۵	۲۵
اولویت	۵	۱	۲	۳	۴

i	۱	۲	۳	۴	۵
w_i	۲۰	۵	۹	۱۰	۱
v_i	۷۰	۳۰	۴۵	۲۵	۴۰
اولویت	۲	۴	۳	۱	۵

در گزینه (۲) اولویت های اول و دوم اشیاء با حجم ۲۵ و ۷۰ هستند. مجموع ارزش آنها به ۳۵ می رسد

در گزینه (۳) :

روش حریصانه با برداشتن شی به حجم ۸۰ به پایان می‌رسد.

ارزش اشیای برداشته شده با این روش همان ۸۰ است.

درحالی که می‌توانستیم با برداشتن چهار شی دیگر، به مجموعه

ارزش ۸۳ برسیم پس روش حریصانه بهینه نیست.

i	۱	۲	۳	۴	۵
w_i	۸۰	۲۰	۲۲	۲۱	۲۰
v_i	۸۰	۲۵	۲۵	۲۵	۲۵
اولویت	۱	۴	۲	۳	۵

۴۷- یک شبکهٔ اجتماعی را در نظر بگیرید که در آن دوستی‌ها لزوماً دو طرفه نیست. بنابراین اگر شخص u، شخص v را بشناسد (دوست باشد) در گراف شبکهٔ اجتماعی یک یال جهت‌دار از u به v درج می‌شود. در این شبکهٔ اجتماعی اگر کسی از خبری مطلع شود آن را به اطلاع همهٔ دوستان خود خواهد رساند. می‌خواهیم یک خبر را به اطلاع همه در این شبکه برسانیم. حداقل چند نفر را باید از این خبر مطلع کنیم تا همه (با نشر خبر) از آن مطلع شوند؟

۱) یک نفر

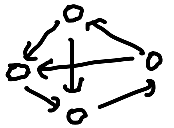
۲) به تعداد مؤلفه‌های قویاً همبند گراف شبکهٔ اجتماعی

۳) به تعداد مؤلفه‌های قویاً همبند گراف شبکهٔ اجتماعی که ورودی از هیچ مؤلفهٔ همبند قوی دیگر ندارند.

۴) هیچ‌یک از گزینه‌ها

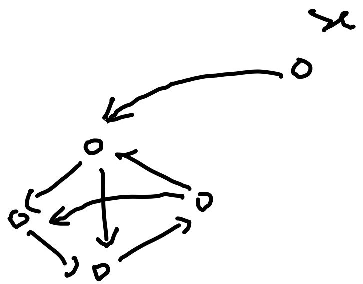
پاسخ: گزینه (۳)

در یک مولفه ی همبند قوی، از هر گره به هر گره دیگر مسیر وجود دارد. بنابراین در یک مولفه‌ی همبند قوی، کافیست خبر به یک گره برسد تا همه ی گره های دیگر آن خبر را دریافت کنند:

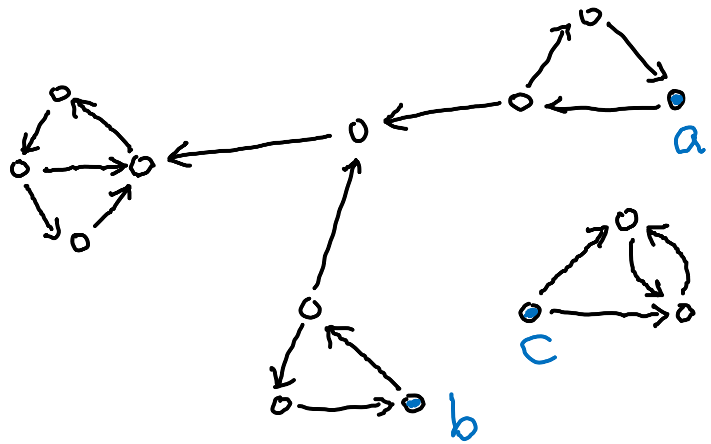


یک مولفه همبند قوی

اگر این مولفه‌ی همبند قوی دارای یک یال ورودی از گره x باشه، کافیست خبر به x برسه تا تمام گره های آن مولفه، آن را دریافت کنند.



بنابراین کافیست از هر مولفه‌ی همبند قوی که ورودی ندارد، یک گره را انتخاب کنیم.



برای مثال در گراف زیر ۶ مولفه‌ی همبند قوی
داریم که فقط سه تا از آنها یال ورودی
ندارند. کافیست خبر را به اطلاع ۳ گره
مثلاً a, b, c برسانیم تا همه افراد مطلع شوند.

۴۸- آرایه $A[0..n-1]$ از اعداد حقیقی داده شده است. می‌خواهیم از A مـاتریس $B[0..n-1, 0..n-1]$ را طـوری بسازیم، که به‌ازای هر $i \le j$ داشته باشیم $B[i,j] = \sum_{k=i}^{j} A[k]$. الگوریتم کارایی که این عملیـات را انجـام دهد از چه مرتبه‌ای است؟

۱) $O(n)$

۲) $O(n^2)$

۳) $O(n^3)$

۴) $O(n\log n)$

پاسخ: گزینه (۲).

تعداد جفت‌های (i, j) که $i \le j$ است برابر با $\binom{n}{2} + n = \Theta(n^2)$ است. محاسبه‌ی هر کدام از $B[i, j]$ ها $O(1)$ است. پس بهترین زمان $O(n^2)$ است.

برای هر $1 \le i \le n$ داریم $B[i, i] = A[i]$. سپس برای هر $1 \le i \le n$ و هر $i+1 \le j \le n$ داریم

$$B[i, j] = B[i, j-1] + A[j]$$