بسم الله الرحمن الرحیم

ساختمان داده و طراحی الگوریتم
دکتری نرم افزار ۱۴۰۲

ابوالفضل گیلک
گروه بابان

آزمون (نیمه‌متمرکز) ورود به دوره‌های دکتری ـ سال ۱۴۰۲

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

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

دفترچهٔ شمارهٔ (۱)

صبح پنج‌شنبه
۱۴۰۱/۱۲/۱۱

مهندسی کامپیوتر ـ نرم‌افزار و الگوریتم (کد ۲۳۵۴)

تعداد سؤال: ۴۵

زمان پاسخ‌گویی: ۱۳۵ دقیقه

عنوان مواد امتحانی، تعداد و شمارهٔ سؤالات

ردیف	مواد امتحانی	تعداد سؤال	از شمارهٔ	تا شمارهٔ
۱	مجموعه دروس تخصصی: ـ ساختمان داده‌ها و طراحی الگوریتم‌ها ـ سیستم‌های عامل پیشرفته ـ پایگاه داده‌های پیشرفته	۴۵	۱	۴۵

استفاده از ماشین‌حساب مجاز نیست.

این آزمون نمرهٔ منفی دارد.

سلام دوستان

پاسخ تشریحی و کامل سوالات ساختمان داده و طراحی الگوریتم دکتری ۱۴۰۲ تقدیم می‌شود.

توصیه‌ی من آن است که حتماً فیلم حل سوالات دکتری ۱۴۰۲ و ارشد ۱۴۰۲ را در سایت بابان مشاهده کنید.

در پاسخ هر سوال تلاش شده با یادآوری درس و سوالات مشابه سال‌های قبل و قراردادن بخشی از جزوه‌ی درسی، حواشی آن موضوع برای شما عزیزان گفته شود.

پوشش ۱۰۰ درصدی تمام سوالات آزمون دکتری و ارشد در کلاس‌های درس و تست نشانه‌ی کامل و جامع بودن این کلاس در گروه بابان است.

با امید پیروزی شما

ابوالفضل گیلک، مولف کتب گسسته، هوش مصنوعی و داده الگوریتم

انتشارات راهیان ارشد.

۱- الگوریتم فلوید ـ وارشال از یک الگوریتم .... برای حل مسئله کوتاه‌ترین مسیرهای تمام جفت رئوس در یک گراف جهت‌دار $\mathbf{G = (V , E)}$ در زمان .... استفاده می‌کند.

۱) حریصانه، $\Theta(v^3)$

۲) حریصانه، $\Theta(V^2 \log E)$

۳) برنامه‌نویسی پویا، $\Theta(v^3)$

۴) برنامه‌نویسی پویا، $\Theta(V^2 \log E)$

پاسخ: گزینه (۳)

فیلم و جزوه جلسه ۱۶ و ۱۷

خلاصه تمام الگوریتم‌های کوتاه‌ترین مسیر:

- مسائل کوتاه‌ترین مسیر
  - تک منبع از یک به سایر گره‌ها
    - گراف غیروزن‌دار — BFS(S) — $E+V$
    - وزن‌ها نامنفی — دکسترا — $E\log V$
    - وزن‌ها دلخواه — بلمن‌فورد — EV
  - بین هر جفت از گره‌ها
    - وزن‌ها نامنفی — V بار دکسترا — $VE\log V$
    - وزن‌ها دلخواه
      - V بار بلمن‌فورد — EV^2
      - ماتریسی
        - عادی — V^4
        - مجذور — $V^3\log V$
      - فلوید وارشال (پویا) — V^3
      - جانسون — $VE\log V$

۲- با فرض اینکه $P \neq NP$ باشد، کدام مورد درست است؟

۱) $NP - hard = NP$

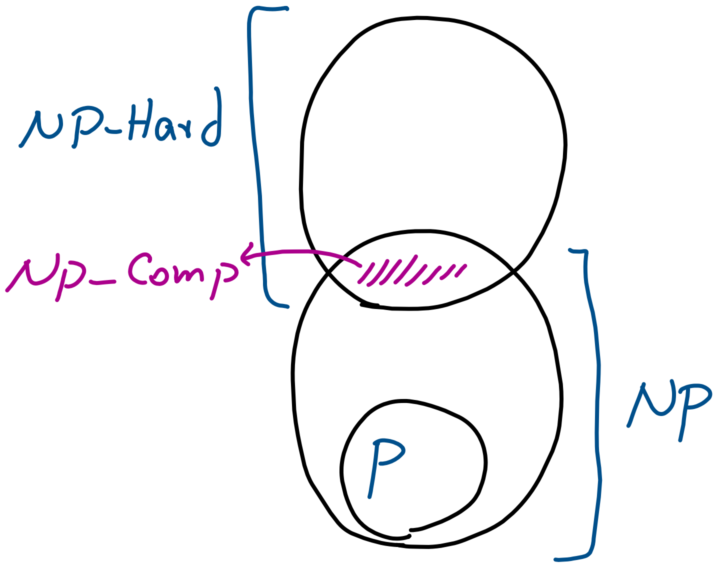
۲) $NP - complete = P$

۳) $NP - complete = NP$

۴) $NP - complete \cap P = \varnothing$

پاسخ: گزینه (۴)

اگر $P \neq NP$ داریم:

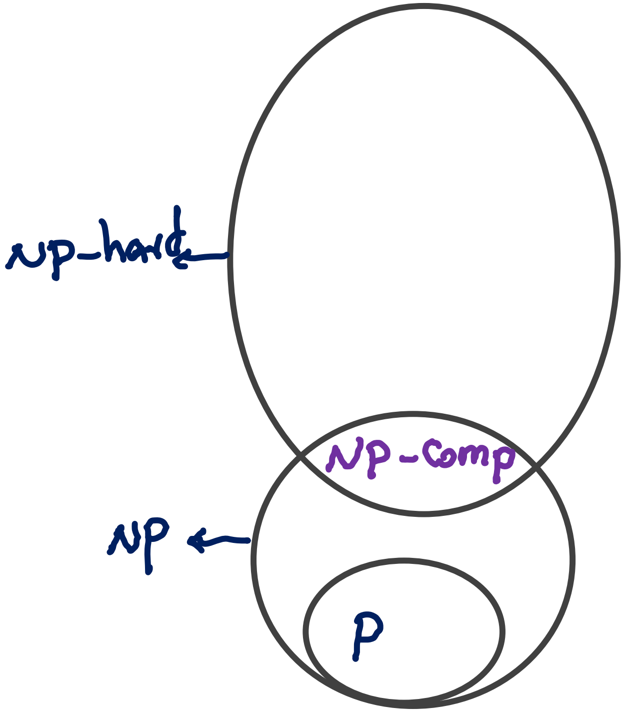


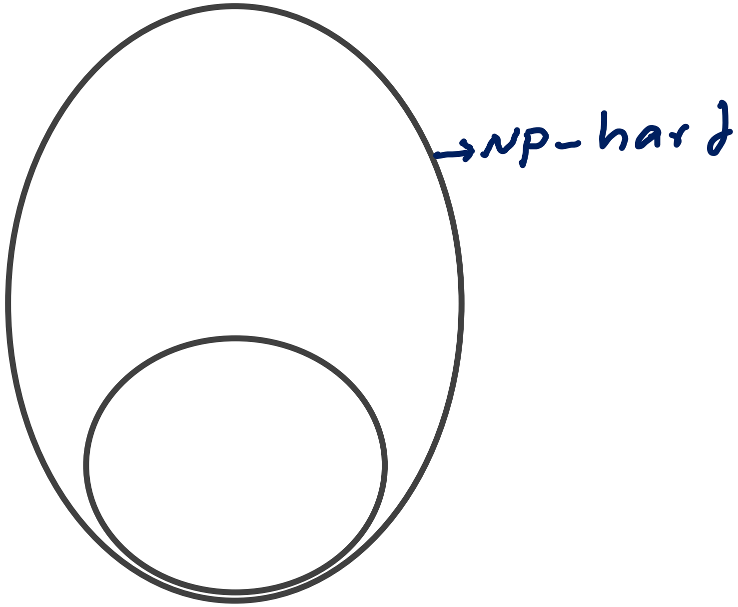
یادآوری:

جلسه ۲۱:

یکی از این دو حالت برقرار است

$P \neq NP$ $\qquad\qquad$ $P = NP$





در این حالت $P \neq NP$

و $P \cap NP\text{-}Comp = \phi$

است.

$P = NP = NP\text{-}comp \subseteq NP\text{-}hard$

در این حالت $P = NP = NP\text{-}Comp$

۳- کم‌ترین تعداد مقایسه مورد نیاز برای تعیین اینکه یک عدد صحیح بیش از $\frac{n}{2}$ مرتبه در یک آرایه مرتب از اعداد صحیح به طول **n** ظاهر می‌شود، از کدام مرتبه است؟

۱) $\Theta(1)$

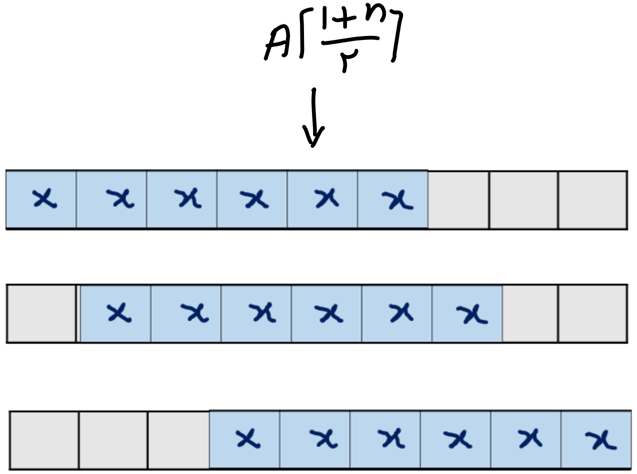
۲) $\Theta(\log n)$

۳) $\Theta(n)$

۴) $\Theta(n \log n)$

پاسخ: گزینه (۱)

در یک آرایه مرتب چنین عددی حتماً در وسط قرار می‌گیرد.



۴- n آرایه نامرتب $A_1, \dots, A_n$ را در نظر بگیرید (n عددی فرد است.) هر کدام از این آرایه‌ها دارای n عنصر متمایز است. هیچ عنصر مشترکی میان هیچ دو آرایه‌ای وجود ندارد. کمترین پیچیدگی زمانی الگوریتمی بـرای محاسـبه میانه این آرایه‌ها از چه مرتبه‌ای است؟

۱) $\Theta(n)$ ۲) $\Theta(n \log n)$

۳) $\Theta(n^2)$ ۴) $\Omega(n^2 \log n)$

پاسخ: گزینه (۳)

یافتن میانه از بین همه اعداد واقع در m لیست که روی هم رفته N عدد دارند:

اگر لیست‌ها مرتب باشند: $O(m \log N)$

اگر لیست‌ها نامرتب و دلخواه باشند: $\Theta(N)$

در این تست $N = n \times n$ است پس جواب: $\Theta(N) = \Theta(n^2)$

یادآوری

جلسه ۶ و ۷ :

یافتن میانه و همچنین یافتن i امین کوچکترین

در یک لیست به طول n

روش ناکارآمد (انتخاب تصادفی)

$\Theta(n^2)$ بدترین

$\Theta(n)$ متوسط

$\Theta(n)$ بهترین

روش کارآمد (غیر تصادفی)

$\Theta(n)$ بدترین

$\Theta(n)$ متوسط

$\Theta(n)$ بهترین

یادآوری

جلسه ۵ نکته تست

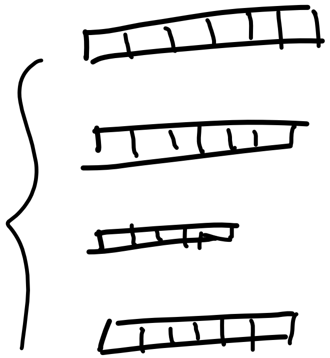
یافتن میانه در اعداد موجود در m لیست:

الف) اگر m لیست مرتب باشند:

$O(m\log(N))$

m: تعداد لیست های مرتب

N: تعداد کل عناصر



m لیست. در کل دارای N عدد.

ب) اگر لیست ها نامرتب باشند،

$O(N)$

N: تعداد کل عناصر

۵- فرض کنید $W(n)$ و $A(n)$، به‌ترتیب، نشان‌دهنده بدترین حالت و میانگین زمان اجرای الگوریتم اجراشـــده بــر روی ورودی با اندازه n باشند. کدام مورد همواره درست است؟

۱) $A(n) = O(W(n))$

۲) $A(n) = \Theta(W(n))$

۳) $A(n) = \Omega(W(n))$

۴) $A(n) = o(W(n))$

پاسخ: گزینه (۱)

نمادهایی که از نظر نرخ رشد نوشته شده اند:

بهترین — متوسط — بدترین

$$B(n) \leq A(n) \leq W(n)$$

$$B(n) = O(A(n)) \qquad A(n) = O(W(n))$$

یادآوری و توضیح تکمیلی

دقت کنید $A(n)$ ممکن است با بدترین یا بهترین برابر شود. مثال ها:

	بدترین $W(n)$	متوسط $A(n)$	بهترین $B(n)$
merge sort	$n\log n$	$n\log n$	$n\log n$
insertion sort	n^2	n^2	n
یافتن میانه با انتخاب تصادفی (روش ناکارآمد)	n^2	n	n

۶- **یک آرایه مرتب‌شده از اعداد داریم. می‌خواهیم دو عدد در این آرایه پیدا کنیم که جمع آن دو عدد مساوی یک عدد داده‌شده x باشد. کمترین پیچیدگی زمانی حل این مسئله کدام است؟**

۱) $\Theta(n)$ ۲) $\Theta(n^2)$

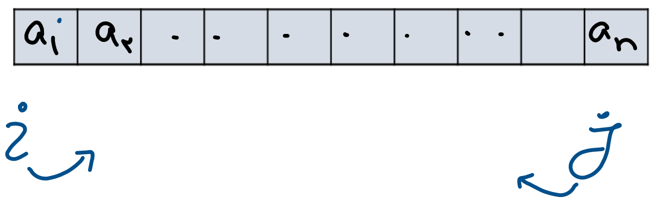
۳) $\Theta(\log n)$ ۴) $\Theta(n \log n)$

پاسخ: گزینه (۱)

برای لیست مرتب می‌توان از جستجوی خطی استفاده کرد.

(اسلاید ۱۱۵۲)

$$a_i + a_j \stackrel{?}{=} x$$



اگر $A[i]$ مرتب باشه: جستجوی خطی قابل انجام است:

$i=1 \qquad j=n$

تا وقتی که $i \leq j$ است:

$a_i + a_j = x \longrightarrow$ END

$a_i + a_j < x \longrightarrow i = i+1$

$a_i + a_j > x \longrightarrow j = j-1$

حداکثر در n مرحله جستجو به پایان می‌رسه

اشاره به تست مهندسی کامپیوتر ۱۴۰۲ (ارشد)

۶۵- فرض کنید $A=[a_1, a_2, \ldots, a_n]$ آرایه‌ای از اعداد صحیح متمایز باشد و k یک عدد صحیح داده شده باشد. هدف این است که دو عدد متمایز از A را پیدا کنید که مجموع آنها دقیقاً k باشد، یا گزارش دهید که چنین عناصری وجود ندارد. یک الگوریتم کارا برای حل این مسئله چه زمانی خواهد داشت؟

۱) $O(\log n)$

۲) $O(n)$

۳) $O(n\log n)$ ✓

۴) $O(n^2)$

← ابتدا لیست باید مرتب شود. سپس جستجوی دودویی یا جستجوی خطی انجام شود.

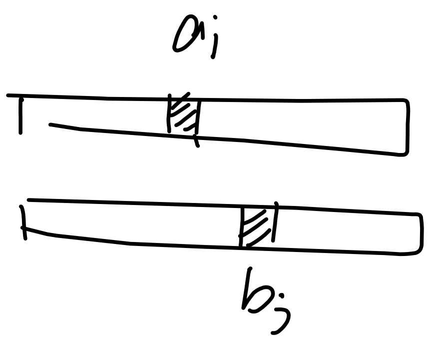
$O(n\log n)$ $O(n\log n)$ $O(n)$

پس در مجموع در هر صورت $O(n\log n)$ به دست می‌آید.

توجه مهم: اگر اعداد صحیح با کران حداکثر $O(n)$ می‌بودند می‌توانستیم از مرتب سازی شمارشی استفاده کنیم:

$$\underbrace{O(n+n)}_{\text{مرتب سازی شمارشی}}+\underbrace{O(n)}_{\text{جستجوی خطی}}=O(n)$$

یادآوری یک مسئله مشابه از کلاس تست:



$a_i + b_j = M$

$\Rightarrow a_i = M - b_j$

۶۱- دو آرایه n تایی A و B حاوی اعداد حقیقی و یک عدد M داده شده‌اند. می‌خواهیم در صورت وجود یک i و یک j پیدا کنیم به طوری که $A[i]+B[j]=M$. بهترین الگوریتم برای حل این مسئله از چه مرتبه‌ای است؟ (علوم کامپیوتر – ۸۱)

۱) $O(n\log n)$ ✓ ۲) $O(n)$ ۳) $O(n^2)$ ۴) $\Omega(n^2)$

مرتب کردن A و سپس جستجوی دودویی اعداد $M-B[j]$ در لیست A.

$$O(\underbrace{n\log n}_{\text{مرتب سازی } A} + \underbrace{n}_{n \text{ بار}}\ \underbrace{\log n}_{\text{جستجوی دودویی}})$$

۷- فرض کنید آرایه‌ای از اعداد صحیح $A=[a_1; a_2; \dots a_n]$ داده شود. فرض کنید یک اندیس (ناشناخته) k وجود دارد به‌طوری که زیرآرایه $A=[a_1; a_2; \dots a_k]$ به‌ترتیب اکیداً افزایشی مرتب شده است و زیرآرایه $A=[a_k; a_{k+1}; \dots a_n]$ به‌ترتیب اکیداً نزولی مرتب شده است (یعنی اگر $1 \le i < j \le k$، آنگاه $a_i < a_j$، و اگر $k \le i < j \le n$، آنگاه $a_i > a_j$ هدف شما تعیین k است. یک الگوریتم بهینه برای حل این مسئله چه زمان اجرایی دارد؟

۱) $\Theta(n^2 \log n)$

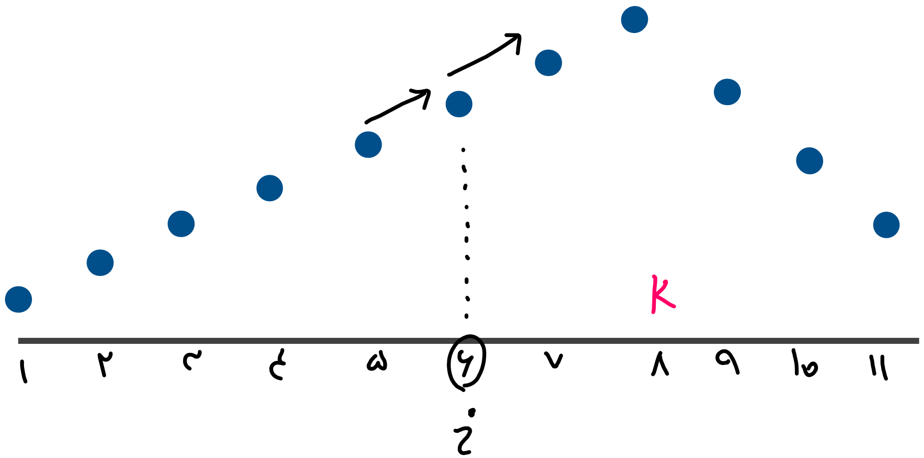
۲) $\Theta(n \log n)$

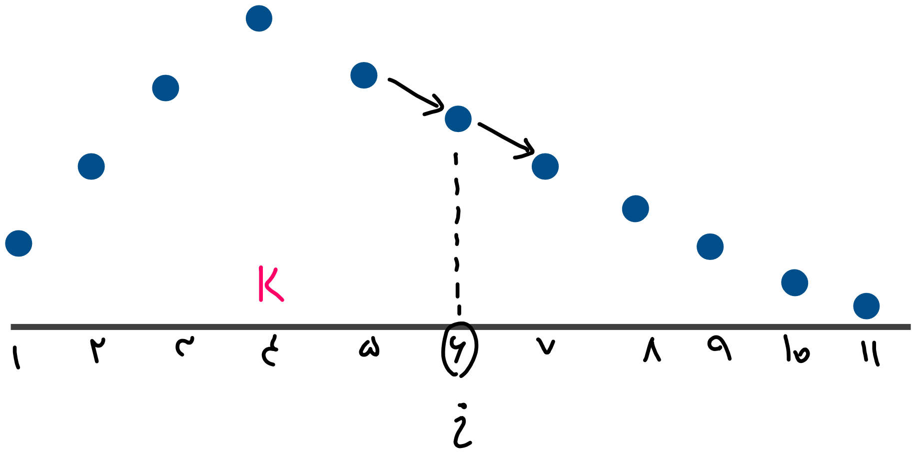
۳) $\Theta(n)$

۴) $\Theta(\log n)$

پاسخ: گزینه (۴) هربار برای اندیس میانی $i=\frac{l+u}{2}$ کافیست رابطه‌ی a_{i+1} و a_i و a_{i-1} بررسی شود تا متوجه شویم k در سمت چپ است یا در سمت راست.

(اسلاید بعدی)





جستجوی دودویی امکان‌پذیر است.

$l=1 \qquad u=n$

تا وقتی $l \leq u$ ادامه دهیم:

$i=\frac{l+u}{2}$

$a_{i-1}<a_i<a_{i+1} \longrightarrow l=i$ (جستجو در نیمه‌ی راست)

$a_{i-1}>a_i>a_{i+1} \longrightarrow u=i$ (جستجو در نیمه‌ی چپ)

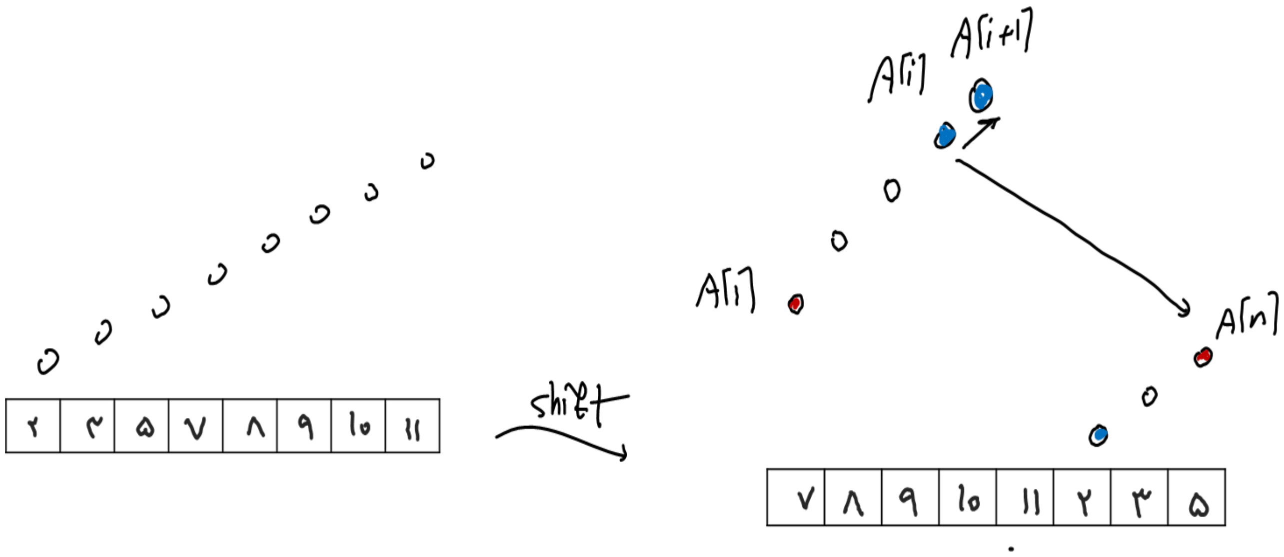
else $\longrightarrow k=i$, END

یادآوری:

مشابه تست حل شده در جلسه ۵ نکته تست:

۶۶- آرایهٔ مرتب A شامل n عدد به صورت اکیداً صعودی داده شده است. یک نفر این آرایه را به اندازه k واحد شیفت دوری داده و نتیجه را به صورت یک آرایهٔ B به ما داده است. هدف پیدا کردن مقدار k است. در چه زمانی می‌توان مقدار k را با داشتن آرایهٔ B محاسبه کرد؟ (بهترین گزینه را انتخاب کنید.) (دکتری نرم‌افزار - ۱۴۰۱)

۱) $O(n)$ ۲) $O(\sqrt{n})$ ۳) $O(\log n)$ ✓ ۴) $O(\log^2 n)$



باید محل min پیدا شود. اگر خانه‌ی وسط آرایه کافیست مقادیر

$A[i]$ و $A[i+1]$ و $A[n]$

همچنین $A[i]$ و $A[i-1]$ و $A[1]$ مقایسه شود.

۸- **کدام‌یک از موارد زیر درست است؟**

۱) آرایه $A = [10; 3; 5; 1; 4; 2]$ یک max heap است.

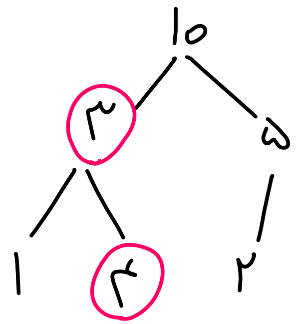
۲) هر مسئله محاسباتی با اندازه ورودی n را می‌توان با یک الگوریتمی با زمان چندجمله‌ای برحسب n حل کرد.

۳) برای تمام توابع مثبت $f(n)$، $g(n)$ و $h(n)$، اگر $f(n) = O(g(n))$ و $f(n) = \Omega(h(n))$ باشد، آنگاه $g(n) + h(n) = \Omega(f(n))$ است.

۴) اگر هر رقم جداگانه در RADIX SORT را با استفاده از INSERTION SORT به‌جای COUNTING SORT مرتب کنیم، آنگاه RADIX SORT به‌درستی کار نمی‌کند (یعنی خروجی صحیح را تولید نمی‌کند).

پاسخ: گزینه (۳)

گزینه (۱) نادرست:



max heap نیست

گزینه ۲ نادرست مثلاً مسأله‌ی n وزیر در زمان چندجمله‌ای حل نمی‌شود.

گزینه ۳ درست: از نظر سرعت رشد داریم: $f=O(g)$ (مانند $f\leq g$) و $f=\Omega(h)$ (مانند $f\geq h$).

$$\begin{matrix} f\leq g \\ f\geq h \end{matrix} \Rightarrow h\leq f\leq g \Rightarrow h+g \overset{\text{جذب}}{\underset{\Theta}{=}} g \Rightarrow h+g \overset{\Omega}{\geq} f$$

گزینه ۴: نادرست؛ مرتب‌سازی درجی، stable است پس مشکلی رخ نمی‌دهد فقط زمان بیشتر می‌شود.

یادآوری ها

برای راحتی بیشتر از $\leqslant$ استفاده می کنیم.

$$\begin{array}{ccccc} \Theta & O & o & \Omega & \omega \\ = & \leqslant & < & \geqslant & > \end{array}$$

البته رابطه ی ترتیب O و Ω و .. ترتیب کامل نیست. مثلاً برخی از توابع هیچ رابطه ای باهم ندارند. مانند

$$f(n) = \begin{cases} n^2 & n \text{ زوج} \\ n & n \text{ فرد} \end{cases} \quad \text{و} \quad g(n) = \begin{cases} n & n \text{ زوج} \\ n^2 & n \text{ فرد.} \end{cases}$$

هیچکدام:

$$f \neq \Omega(g) \qquad f \neq o(g) \qquad f \neq \Theta(g)$$

قانون جذب:

$$\text{if} \quad h = o(g) \Rightarrow h + g = \Theta(g)$$

یادآوری (جزوه جلسه‌ی هفتم) (درباره‌ی گذشته)

* اهمیت stable بودن:

در Radix sort اگر از روشی stable استفاده کنیم، مرتب‌سازی نادرست انجام می‌شود:

یکان	دهگان	
۷۵	۷۲	۷۵
۸۳	۸۳	۷۲
۷۲	۷۵	۸۳

۷۵ ، ۸۳ ، ۷۲ ⇒ ۷۲ ، ۸۳ ، ۷۵ ⇒ ۷۵ ، ۷۲ ، ۸۳ (اشتباه مرتب شده‌اند)

روش‌های شمارشی و درجی، stable هستند.

Time:

	Time	
Radix + درجی	$O(dn^2)$	ok ← درست کار می‌کند اما زمان زیاد می‌شود.
Radix + شمارشی	$O(d(n+p-1))$	ok ← درست کار می‌کند.
Radix + ادغامی	False	(اشتباه) ← نادرست می‌شود. (ادغامی، stable نیست)

۹- کدام مورد زیر مطمئناً عبارت $f(n) = \Omega(g(n))$ را پشتیبانی می‌کند؟

۱) $f(n) \le ۴ \times g(n)$ برای تمام $n \ge ۱$

۲) $f(n) \ge ۴ \times g(n)$ برای تمام $n \ge ۱۳۶$

۳) $f(n) \le ۴ \times g(n)$ برای تمام $n \ge ۱۰۰$

۴) $\lim_{n \to \infty} \frac{f(n)}{g(n)} = ۰$

پاسخ: گزینه (۲)

برای تضمین $f = \Omega(g)$ باید داشته باشیم: (C_1 و C_2 اعداد ثابت مثبت دلخواه هستند)

$$\exists N \quad \forall n \geq N \quad c_1 f(n) \geq c_2 g(n)$$

در گزینه (۲) این را داریم.

در مورد گزینه (۴):

یادآوری جلسه اول (اگر مثبت باشند قدر مطلق لازم نیست)

$$\lim_{n\to\infty}\left|\frac{f(n)}{g(n)}\right| = \infty \longrightarrow g(n) = o(f(n)) \quad , \quad f(n) = \omega(g(n))$$

$$\lim_{n\to\infty}\left|\frac{f(n)}{g(n)}\right| = 0 \longrightarrow g(n) = \omega(f(n)) \quad , \quad f(n) = o(g(n))$$

$$\lim_{n\to\infty}\left|\frac{f(n)}{g(n)}\right| = c > 0 \longrightarrow g(n) = \Theta(f(n)) \quad , \quad f(n) = \Theta(g(n))$$

۱۰- کدام‌یک از گزاره‌های زیر درست است؟

الف- اگر مسئله P_1 بتواند به مسئله P_2 در زمان خطی کاهش (reduce) یابد، آنگاه اگر P_2 یک مسئله NP-hard باشد، می‌توان نتیجه گرفت P_1 نیز NP-hard است.

ب- یک Clique در یک گراف بدون جهت لزوماً یک vertex cover در گراف مکمل نیست.

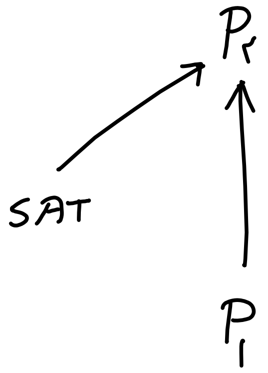
۱) فقط گزاره «الف» درست است.

۲) فقط گزاره «ب» درست است.

۳) هر دو گزاره «الف» و «ب» درست است.

۴) هر دو گزاره «الف» و «ب» نادرست است.

پاسخ: گزینه (۲)



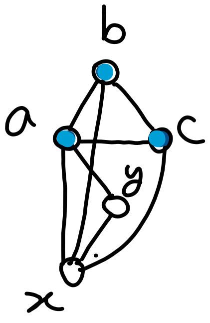
الف: نادرست است.

P_2 در زمان خطی (که چندجمله‌ای است) به P_2 کاهش یافته پس P_2 از P_1 سخت‌تر است.

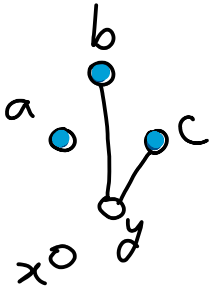
P_2 مسأله‌ی NP-Hard است پس مسأله‌ی (SAT) (مانند NP-Comp ها) به آن کاهش می‌یابد. از این‌ها نمی‌توان نتیجه گرفت که P_1 هم NP-Hard است.

ب: درست است. دقت کنید که ادعای (ب) درمورد maxclique نیست فقط درمورد clique است.

مثلاً a,b,c در G یک خوشه هستند اما همان‌ها در $\bar{G}$ یک پوشش رأسی نیستند.



G



$\bar{G}$

(یادآوری جلسه‌ی ۲۱)

مسائل np-hard :

هر مسئله‌ای که حداقل به سختی SAT باشه، مسئله‌ی np-hard است. دقت کنید که

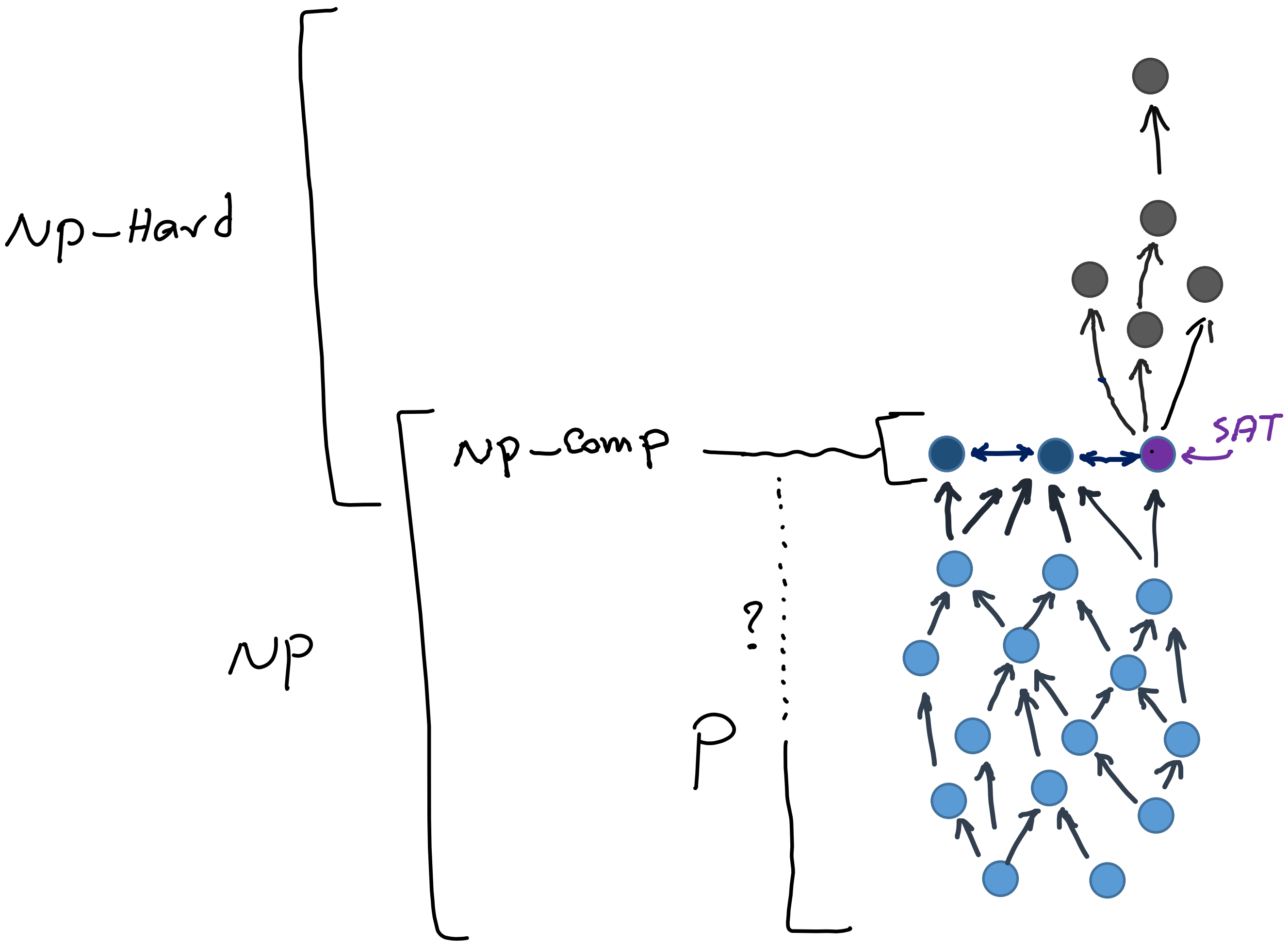
مسائل np-hard ممکن است از رده‌ی NP خارج شوند.

* مسئله‌ی M عضو رده‌ی np-hard است اگر و تنها اگر مسئله‌ی SAT به آن قابل کاهش باشه.

$SAT \rightarrow M$

$M \in$ np-hard

یادآوری — جلسه ۲۱



شکل مقابل نحوه‌ی کاهش مسئله‌ها به هم.

دقت کنید ممکن است کل P و NP

درون ناحیه‌ی NP-Comp

جمع شود.

مسئله‌ها در NP-Comp به یکدیگر

قابل کاهش (در چندجمله‌ای) هستند.

SAT یکی از آنهاست.

۱۱- کدام‌یک از گزاره‌های زیر درست است؟

الف- اینکه زمان حل یک مسئله P حد پایین $\Omega(n^2)$ دارد به این معنی است که برای هر الگوریتم A که P را حل می‌کند فقط برخی از نمونه‌های P وقتی به‌عنوان ورودی به A داده شوند، باعث می‌شود A زمان $\Omega(n^2)$ صرف کند.

ب- اینکه زمان حل یک مسئله P حد پایین $\Omega(n^2)$ دارد به این معنی است که برای هر الگوریتم A که P را حل می‌کند هر نمونه از P که به‌عنوان ورودی به A داده شود، باعث می‌شود A زمان $\Omega(n^2)$ صرف کند.

۱) فقط گزاره «الف» درست است.
۲) فقط گزاره «ب» درست است.
۳) هر دو گزاره «الف» و «ب» درست هستند.
۴) هر دو گزاره «الف» و «ب» نادرست هستند.

پاسخ: گزینه ۲

وقتی می‌گوییم $Time(Problem) = \Omega(n^2)$

(به زبان ساده یعنی: $cn^2 \leq Time(Problem)$)

یعنی: هر الگوریتمی که برای حل این مسئله یافت شود، هر نمونه از آن را در زمان $\Omega(n^2)$ حل می‌کند.

۱۲- **کدام‌یک از گزاره‌های زیر درست است؟**

الف- اگر یک الگوریتم زمان چندجمله‌ای برای یک مسئله که NP-hard است ارائه شود، آنگاه می‌توان نتیجه گرفت $P = NP$ است.

ب- اگر یک مسئله NP-complete است، آنگاه می‌توان نتیجه گرفت که آن مسئله هیچ راه‌حلی ندارد.

۱) فقط گزاره «الف» درست است.
۲) فقط گزاره «ب» درست است.
۳) هر دو گزاره «الف» و «ب» درست هستند.
۴) هر دو گزاره «الف» و «ب» نادرست هستند.

پاسخ: گزینه (۱)

الف: اگر یک np-hard را در زمان چندجمله‌ای حل کنیم آنگاه مسأله‌ی SAT هم در زمان چندجمله‌ای حل می‌شود پس np-comp با P اشتراک پیدا می‌کند

پس $P = NP$

ب: مسأله‌ی np-comp راه‌حل دارند اما هنوز در زمان چندجمله‌ای ممکن است حل نشده باشند.

۱۳- کدام‌یک از گزاره‌های زیر درست است؟

الف- اگر یک مسئله که به کلاس NP تعلق دارد یک راه‌حل زمان چندجمله‌ای داشته باشد، آنگاه P = NP است.

ب- اگر کسی یک حد پایین زمان نمایی برای یک مسئله که NP-complete است بدهد، آنگاه P ≠ NP است.

۱) فقط گزاره «الف» درست است. ۲) فقط گزاره «ب» درست است.

۳) هر دو گزاره «الف» و «ب» درست هستند. ۴) هر دو گزاره «الف» و «ب» نادرست هستند.

پاسخ: گزینه (۲)

الف: نادرست است. همه مسائل عضو P عضو NP هم هستند یعنی بسیاری از اعضای NP در زمان چندجمله‌ای حل شده‌اند (مانند مسئله‌ی مرتب‌سازی) اما این به معنای تساوی P و NP نیست.

توجه: شکل درست جمله (الف) چنین است:

(اگر یک مسئله‌ی NP-comp در زمان چندجمله‌ای حل شود آنگاه P=NP است.)

ب: درست است.

اگر برای یک مسئله NP-comp ثابت شود که $Time = \Omega(2^n)$ آنگاه ثابت شده که راه‌حل در زمان حداکثر چندجمله‌ای ندارد پس عضو P نیست. در نتیجه $P \neq NP$.

۱۴- مسئله کوله‌پشتی ۱−۰ را در نظر بگیرید که n شئ با وزن صحیح داریم و گنجایش کوله‌پشتی عدد صحیح M است. این مسئله دارای یک الگوریتم مبتنی بر روش برنامه‌ریزی پویا با زمان $O(M.n)$ است. این مرتبه زمانی برحسب اندازه مسئله چگونه است؟

۱) خطی
۲) درجه دو
۳) نمایی
۴) شبه چندجمله‌ای

پاسخ: گزینه (۴)

ورودی‌ها عبارتند از: تعداد اشیاء، وزن هر کدام و گنجایش کوله.

زمان اجرا $O(nM)$.

ازآنجاکه M ممکن است وابسته به n باشد این را برحسب اندازه ورودی، شبه چندجمله‌ای گویند.

۱۵- برای پیدا کردن **k** امین عدد در میان **n** عدد که به‌عنوان کلید در گره‌های یک درخت جستجویی دودویی متوازن ذخیره شده‌اند، کمترین پیچیدگی زمانی ممکن کدام است؟ (هر گره درخت فقط شامل کلید و اشاره‌گر به پدر و فرزند چپ و راست است.)

۱) $\Theta(n)$
۲) $\Theta(\log n)$
۳) $\Theta(n \log n)$
۴) $\Theta(\log \log n)$

پاسخ: گزینه (۱)

یافتن K امین عنصر در درخت جستجوی دودویی BST حتی اگر متوازن باشد،
بازهم از مرتبه‌ی $\Theta(n)$ است. مگر آن که علاوه بر فیلد کلید و اشاره‌گرهای پدر، فرزند،
تعداد گره‌های هر زیردرخت در ریشه درج شده باشد (آماره‌ای) در این صورت زمان $O(h)$
است و برای درخت BST متوازن $h = \Theta(\log n)$ است.

یادآوری

مرتبه زمانی یافتن k امین کوچکترین عنصر:

۱) در یک لیست دلخواه: $\Theta(n)$ (مانند میانه)

۲) در ساختمان های داده تا حدی مرتب مانند BST و minheap، maxheap، ... $\Theta(n)$ ← مگر آن که درخت BST از نوع آماره‌ای باشد در این صورت: $O(h)$

اگر متوازن باشد $h = \log n$

۳) در یک لیست مرتب، $\Theta(1)$ (در اندیس k ام قرار دارد)

یادآوری : جعبه ۱۷ نکته تست

ارتفاع درخت BST متوازن $h=\Theta(\log n)$ است.

۶۷- فرض کنید یک درخت دودویی جستجو بر روی n عدد حقیقی متمایز با ارتفاع $O(\log n)$ در اختیار داریم. چه تعداد از پرسمان‌های زیر را بدون پیش‌پردازش و اطلاعات اضافی می‌توان در $O(\log n)$ پاسخ داد؟ (در هر گره صرفاً یک کلید و دو اشاره‌گر به فرزندان نگه‌داشته شده است.)

- محاسبه کوچکترین عدد T
- محاسبه میانه F
- تعیین آنکه آیا عدد داده شده x در درخت وجود دارد. T
- محاسبه مرتبه عدد x داده شده در بین n عدد ذخیره شده در درخت F

۱) صفر ۲) ۳ ۳) ۲ ۴) ۱

این اقدامات در درخت جستجوی دودویی آماره‌ای در زمان $O(h)$ انجام می‌شوند اما در درخت جستجوی دودویی معمولی در زمان $O(n)$.

۱۶- n عدد به‌عنوان کلید در گره‌های یک درخت جستجویی دودویی متوازن ذخیره شده‌اند. هر گره علاوه بر کلید و اشاره‌گر به پدر و فرزند چپ و راست، تعداد گره‌های زیر درخت خود را هم نگهداری می‌کند. برای پیدا کردن rank کلید یک گره (یعنی اینکه کلید گره چندمین عدد در بین n عدد است) کمترین پیچیدگی زمانی ممکن کدام است؟

۱) $\Theta(\log\log n)$
۲) $\Theta(\log n)$
۳) $\Theta(n)$
۴) $\Theta(n\log n)$

پاسخ: گزینه (۲)

یافتن k امین عنصر در درخت جستجوی دودوئی BST، همچنین یافتن rank هر کلید، در درخت BST که تعداد گره‌های هر زیردرخت در ریشه درج شده باشد (آماره‌ای) در زمان O(h) انجام می‌شود. در درخت BST آماره‌ای متوازن ارتفاع $h=O(\log n)$ است.

۱۷- در یک درخت قرمز ـ سیاه، طول طولانی‌ترین مسیر ساده از یک گره x به یک برگ در زیر درخت خودش حداکثر چند برابر طول کوتاه‌ترین مسیر از گره x به یک برگ در زیر درخت خودش است؟

۱) ۴ ۲) ۳

۳) ۲ ۴) ۱

پاسخ: گزینه (۳)

در درخت قرمز سیاه برای هر گره x ، اگر l و L طول کوتاه‌ترین و بلندترین مسیر از x تا برگ‌ها باشد داریم $L \leq 2l$.

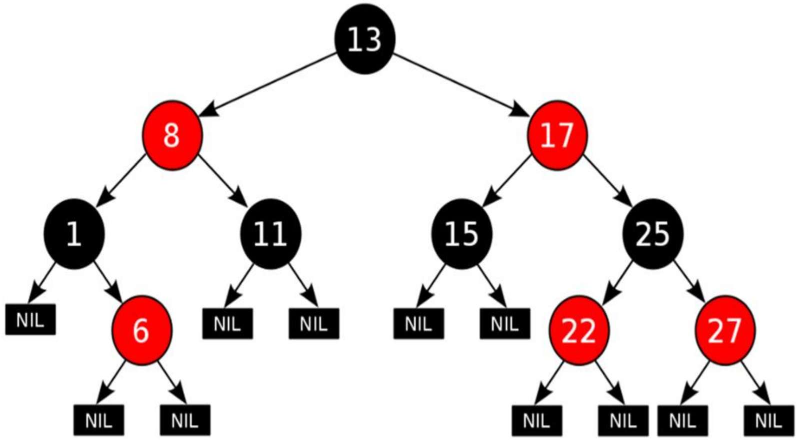
به طور کلی برای طول هر دو مسیر از x تا برگ ها داریم $\frac{1}{2} \leq \frac{l_1}{l_2} \leq 2$.

* در نتیجه‌ی همین ویژگی ارتفاع درخت قرمز سیاه $O(\log n)$ است.

یادآوری (جلسه ۱۱)

نتیجه: اگر l_1 و l_2 طول ۲ مسیر دلخواه از ریشه تا برگ باشند:

قرمز سیاه $\Longrightarrow$ $\frac{1}{2} \le \frac{l_1}{l_2} \le 2$



۱۸- **کدام‌یک از گزاره‌های زیر درست است؟**

الف- هر درخت جستجوی دودویی دلخواه با n گره می‌تواند به یک درخت جستجوی دودویی دلخواه دیگر با n گره با انجام $O(n)$ عمل rotation تبدیل شود.

ب- برای هر دو تابع $f(n)$ و $g(n)$ یکی از سه حالت (۱) $f(n) \in o(g(n))$، (۲) $f(n) \in w(g(n))$ و (۳) $f(n) \in \theta(g(n))$ برقرار است.

۱) فقط گزاره «الف» درست است.

۲) فقط گزاره «ب» درست است.

۳) هر دو گزاره «الف» و «ب» درست است.

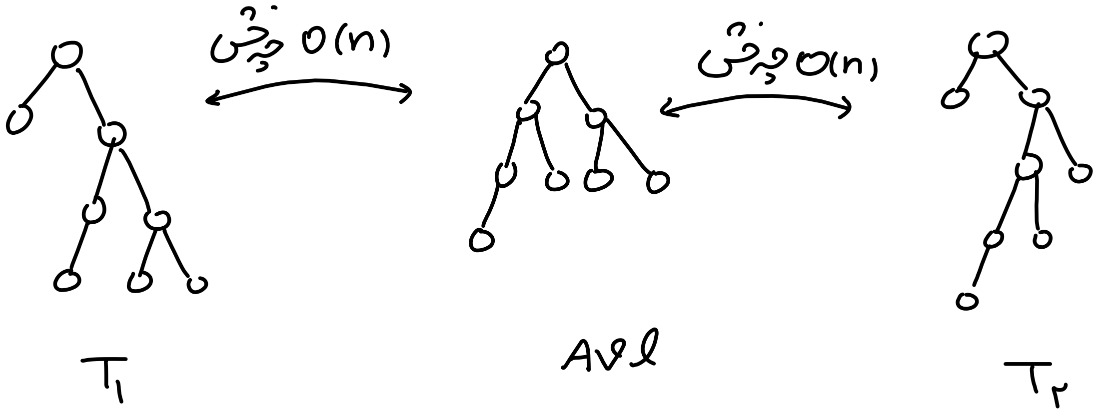
۴) هر دو گزاره «الف» و «ب» نادرست است.

پاسخ: گزینه (۱) توضیح خاصی لازم نیست به یادآوری‌های زیر از کلاس درس توجه کنید:

یادآوری

با تعداد $O(n)$ چرخش دو درخت BST به درخت BST متوازن (AVL) تبدیل می‌شود.

به همین دلیل با حداکثر $O(2n)$ چرخش از هر BST به BST دیگر می‌رسیم.



یادآوری

جنبه اول

این ادعاها غلط هستند.

غلط: $g(n) \in \omega(f(n)) \vee g(n) \in o(f(n)) \vee g(n) \in \theta(f(n))$

غلط: $g(n) \notin O(f(n)) \Rightarrow g(n) \in \omega(f(n))$

مثلاً این دو تابع هیچ رابطه‌ای از نوع θ ، O ، o ، Ω ، ω با هم ندارند:

$$f(n) = \begin{cases} n^4 & n \text{ زوج} \\ n & n \text{ فرد} \end{cases}$$

$$g(n) = \begin{cases} n^2 & n \text{ زوج} \\ n^5 & n \text{ فرد} \end{cases}$$

۱۹- **کمترین پیچیدگی زمانی ممکن برای مرتب‌سازی n عدد طبیعی که مقادیر کمتر از n^2 دارند، کدام است؟**

۱) $\Theta(n\log^2 n)$

۲) $\Theta(n^2)$

۳) $\Theta(n\log n)$

۴) $\Theta(n)$

پاسخ: گزینه (۴)

یک تمرین معروف از CLRS است که در کلاس نیز بررسی کرده‌ایم.

اگر مرتب‌سازی شمارشی استفاده کنیم: $O(n+n^2)=O(n^2)$ به دست می‌آید.

اگر از Radix استفاده کنیم $O(d(n+p-1))$ به دست می‌آید. (d تعداد ارقام و p مبنای به کار رفته است)

حالا چون همه اعداد $a_i < n^2$ هستند اگر از مبنای $p=n$ استفاده کنیم حداکثر ۲ رقمی می‌شوند: $d \le 3$

پس: $Time = O(3(n+n-1)) = O(n)$

یادآوری : جلسه ۶ (Radixsort)

8.3-4

Show how to sort n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.

$0 < a_i < n^3$ ⟶ درمبنای n حداکثر ۳ رقمی می‌شوند

مرتب‌سازی n عدد صحیح در بازه‌ی $[0, n^3-1]$ یعنی $0 \leq a_i \leq n^3-1$:

$O(n+n^3)$ ⟵ مرتب‌سازی شمارشی

$O(3n)$ ⟵ $p=n$ مرتب‌سازی Radix درمبنای

۲۰- اگر عدد ۳۶۳ را در یک درخت جستجوی دودویی، جستجو کنیم، کدام دنبالهٔ زیر نمی‌تواند دنباله‌ای از کلید گره‌هـایی باشد که بررسی می‌شوند؟ (ترتیب از راست به چپ است.)

۱) ۲، ۲۵۲، ۴۰۱، ۳۹۸، ۳۳۰، ۳۴۴، ۳۹۷، ۳۶۳

۲) ۹۲۴، ۲۲۰، ۹۱۱، ۲۴۴، ۸۹۸، ۲۵۸، ۳۶۲، ۳۶۳

۳) ۲، ۳۹۹، ۳۸۷، ۲۱۹، ۲۶۶، ۳۸۲، ۳۸۱، ۲۷۸، ۳۶۳

۴) ۹۲۵، ۲۰۲، ۹۱۱، ۲۴۰، ۹۱۲، ۲۴۵، ۳۶۳

پاسخ: گزینه (۴)

$(K = 363)$

در گزینه (۴) (راست به چپ):

$K < 925$ پس همه اعداد بعدی باید کمتر از ۹۲۵ باشند (OK)

$K > 202$ پس همه اعداد بعدی باید بیشتر از ۲۰۲ باشند (OK)

$K < 911$ پس همه اعداد بعدی باید کمتر از ۹۱۱ باشند (NOT)

توضیح و یادآوری

یک مسأله‌ی پرتکرار است که بارها در کلاس حل شده است. (در این مثال از راست به چپ)

اعداد را به ترتیب a_1 ، a_2 ، a_3 ، ... ، نامیم. کلید مورد نظر $K = ۳۶۳$ است.

قاعده:

اگر $K < a_i$ باشد باید همه اعداد بعدی نیز از a_i کمتر باشند.

اگر $K > a_i$ باشد باید همه اعداد بعدی نیز از a_i بزرگتر باشند.

(اگر $K = a_i$ متوقف می‌شویم)

یادآوری　　جعبه ۴ نکته تست

۸۳– اعداد 1 تا 500 را در یک درخت دودویی جست‌وجو ذخیره کرده‌ایم. می‌خواهیم عـدد 193 را در این درخت جست‌وجو کنیم. کدام دنباله نمی‌تواند مسیر جست‌وجو برای عدد 193 باشد؟

(مهندسی فناوری اطلاعات – ۹۹)

۱) 193، 177، 105، 102، 101، 55، 30، 20، 4

۲) 193، 200، 150، 102، 101، 55، 30، 20، 4

۳) 193، 190، 143، 201، 231، 237، 157، 437 ✓

۴) 193، 150، 100، 50، 200، 300، 400، 500

عدد مورد جستجو: $x = ۱۹۳$

در گزینه ۳ وقتی $a_i = ۱۵۷$ را می‌بینیم،

x از a_i بزرگتر است پس همه اعداد بعدی

باید از a_i بزرگتر باشند که اینطور نیست: ۱۴۳ غلط است.

(یادآوری جستجوی دودویی)

قانون جستجوی دودویی:

عدد مورد جستجو: $x = 193$

اعداد ملاقات شده: $a_1\ a_2 \ldots\ldots$

در هر مرحله:

اعداد بعدی هم از a_i بزرگتر باشند $\longrightarrow$ $\text{if} \quad a_i < x$

اعداد بعدی هم از a_i کوچکتر باشند $\longrightarrow$ $\text{if} \quad x < a_i$