به نام او

ساختمان داده‌ها و الگوریتم‌ها

نیم‌سال دوم ۹۸-۹۷

وقت امتحان: ۳ ساعت

دانشکده‌ی مهندسی کامپیوتر

پاسخنامه آزمون میان‌ترم دوم

سوالات کوتاه‌پاسخ (۳۰ نمره)

الف) نادرست. زیرا درخت تصمیم آن دارای ۱۲۰ = !۵ برگ دارد و $log۱۲۰ > ۶$ می‌باشد. پس به حداکثر ۷ مقایسه در بدترین حالت نیاز داریم.

ب) یکی از دو جایگشت $n, n-۱, ..., ۱$ یا $۱, ۲, ...n$ می‌تواند باشد.

ج) ابتدا تعداد خروجی‌های ممکن الگوریتم را به‌دست می‌آوریم. هر کدام از خروجی‌های این الگوریتم یک جایگشت nتایی است که هر کدام از $\frac{n}{k}$ لیست داده شده در آن مرتب باشند پس تعداد خروجی‌های ممکن الگوریتم که همان تعداد برگ‌هایی است که باید توسط درخت تصمیم تولید شود $\frac{n!}{(\frac{n}{k})!^k}$ است. پس با استدلالی مشابه الگوریتم‌های مرتب‌سازی ارتفاع درخت تصمیم از $\Omega(\lg \frac{n!}{(\frac{n}{k})!^k})$ است. از طرفی داریم

$$\lg \frac{n!}{(\frac{n}{k})!^k} = \lg n! - k \lg \frac{n!}{k!} \geqslant \lg n! - k\frac{n}{k} \lg \frac{n}{k} = \lg \frac{n!}{n^n} + n \lg k$$

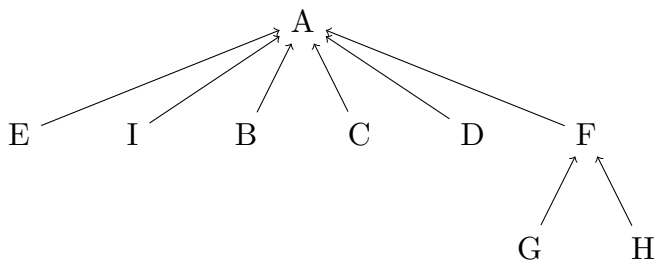
$$\lg \frac{n!}{(\frac{n}{k})!^k} \geqslant \lg \frac{n!}{n^n} + n \lg k \geqslant \lg e^{-n} + n \lg k = -n + n \lg k \in \Omega(n \lg k)$$

پس ارتفاع درخت تصمیم از $\Omega(n \lg k)$ است.

د)
- A مقدار ۱۰ و B مقادیر $۳۰ < B < ۴۱$ را می‌تواند بگیرد.
- ۴
- ۴
- ۱۹

ه)
- یک راه حل:

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



و)

- تعداد برخورد: ۲۰

	12	11	17	22	14	25	15		
0	1	2	3	4	5	6	7	8	9

- تعداد برخورد: ۱۳

14	12	11	22		17	25	15		
0	1	2	3	4	5	6	7	8	9

- تعداد برخورد: ۶

	12	11		22	14	15	25	17	
0	1	2	3	4	5	6	7	8	9

مسئله‌ی ۱. تو هیچ‌گاه تنها گام برنخواهی داشت (۱۵ نمره)

برای پیاده‌سازی جواب نیاز به یک حلقه داریم. در این حلقه از دو متغیر low و $high$ برای مشخص کردن ابتدا و انتهای بازه جست‌وجو استفاده می‌کنیم. در هر بار اجرای حلقه یک محور تصادفی در بازه $[low, high]$ (روی اندیس‌ها) انتخاب شده و آرایه حول آن محور به صورت درجا بخش‌بندی می‌شود. دقت کنید که خود عملیات بخش‌بندی نیاز به $O(1)$ حافظه اضافی دارد و در زمان خطی انجام می‌شود.
پس از پایان بخش‌بندی اگر مقدار محور را p و اندیس محور را i در نظر بگیریم دو بازه $[low, i-1]$ و $[i+1, high]$ روی اندیس‌ها به دست می‌اید. در صورتی که $i - low < p - A[low]$ باشد حتما در بازه $[low, i-1]$ و در غیر این صورت حتما در بازه $[i+1, high]$ جواب وجود دارد پس کافیست low و $high$ جدید را با توجه به بازه جدید جواب‌دار تنظیم کنیم.
حلقه باید در یکی از این سه حالت متوقف شود: $low = high = 0$ که در این صورت x برابر است با $A[low] - 1$، $low = high = n - 1$ یا $high = low + 1$ که در این صورت x برابر است با $A[low] + 1$.
با توجه به اینکه الگوریتم کلا از سه متغیر کمکی p و low و $high$ استفاده کرده و بخش‌بندی نیز با حافظه اضافی $O(1)$ انجام می‌شود حافظه اضافی مورد نیاز الگوریتم $O(1)$ است. برای تحلیل زمانی اگر متوسط زمان اجرای الگوریتم روی آرایه‌ای با اندازه n را با $\bar{T}(n)$ نشان دهیم مشخص است که:

$$\bar{T}(n) = \sum_{i=0}^{n-1} \frac{\bar{T}(i)}{n}$$

اگر فرض استقرا این باشد که $\forall 1 < i < n : \ T(i) \leqslant Ci$ و $T(0) \leqslant C$ آنگاه

$$\bar{T}(n) \leqslant \sum_{i=0}^{n-1} \frac{Ci}{n} = \frac{C}{n} + C\frac{n(n-1)}{2n} \leqslant Cn \to \bar{T}(n) = O(n)$$

مسئله‌ی ۲. دیدن سوال صددرصد دلخواه در صبح زیبای ماه رمضان (۲۰ نمره)

برای حل این مسئله از داده‌ساختار $trie$ استفاده می‌کنیم. برای هر گره یک شمارنده ذخیره می‌کنیم که نشان‌دهنده این است که این گره پیشوند چه تعداد از رشته‌های درج شده است و آن‌را بدین صورت به‌روز رسانی می‌کنیم که هنگام درج هر رشته، به شمارنده هر گره در مسیر درج یک واحد اضافه می‌کنیم.
برای محاسبه پاسخ مسئله، متغیر $answer$ را با مقدار اولیه صفر تعریف می‌کنیم. حال هنگام درج کردن یک رشته ، به هر گره‌ای که رسیدیم مقدار شمارنده آنرا به $answer$ اضافه می‌کنیم. دقت شود که این کار را می‌بایست قبل از آنکه به شمارنده آن گره یک واحد اضافه می‌کنیم انجام دهیم.
پیچیدگی زمانی این الگوریتم به اندازه تعداد درج‌ها در $trie$ است که هر عملیات درج خود به اندازه طول رشته زمان می‌برد. با توجه به آنکه طول رشته‌ها محدود است، پس در مجموع پیچیدگی زمانی الگوریتم از مرتبه $O(n)$ باقی می‌ماند.

مسئله‌ی ۳. وصیت‌نامه‌ی گالوا (۱۵ نمره)

کلید حل این سوال در روشی است که درخت را پیمایش می کنیم. اگر از پیمایش پایین به بالا $(bottom - up)$ استفاده کنیم می توانیم با پاس دادن یک سری اطلاعات خاص از زیردرخت سمت چپ و زیردرخت سمت راست به راس پدر سوال را حل کنیم.
اطلاعاتی که در هر مرحله باید از زیر درخت به راس پدر منتقل شود:

- که آیا خود زیردرخت BST هست یا نه.
- اگر زیردرخت، زیردرخت سمت چپ راس پدر باشد، مقدار Max این زیردرخت به پدر منتقل شود.
- اگر زیردرخت، زیردرخت سمت راست راس پدر باشد، مقدار Min این زیردرخت به پدر منتقل شود.
- سایز زیردرخت اگر BST است.

با داشتن این اطلاعات چک کردن این که زیردرختی که ریشه اش راس پدر است BST است یا نه در $O(۱)$ انجام می پذیرد و در کل الگوریتم از $O(n)$ خواهد بود.
*دقت شود که پیمایش از بالا به پایین منجر به الگوریتمی از $O(n^۲)$ می شد.

مسئله‌ی ۴. مَمَدهایی که بودن (۲۰ نمره)

فرض کنید k طول رشته ای با حداقل دو بار تکرار باشد ؛در نتیجه رشته ای به طول $k - ۱$ نیز وجود دارد که حداقل دو بار تکرار شده است.
با در نظر گرفتن این موضوع برای پیدا کردن اندازه طولانی ترین زیر رشته روی k جستجوی $binary$ انجام میدهیم.
در هر مرحله از جستجوی زیررشته های به طول k را با بکارگیری روش رابین کارپ هش میکنیم. به کمک $hashset$ مناسب در $O(n)$ اعداد تکراری در فهرست هش های محاسبه شده را پیدا میکنیم.