نام و نام خانوادگی: مراد نیولی به نام خدا کد دانشجویی: ۸۱۰۶۰۱۰۵۱

Subject: تمرین سوم - ساختمان داده Year. Month. Date.

جواب سوال ۱:

الف)

ابتدا از هر دو رأس شروع می‌کنیم و همینطور به پدرهای آن‌ها می‌رویم تا برای هر دو رأس u و v عمق آن‌ها را درون متغیری ذخیره کنیم. سپس از رأسی که عمق بیشتری دارد (ارتفاع کمتری) شروع می‌کنیم و به اندازه آن می‌رویم، در مرحله بعد دوباره رأسی که ارتفاع بیشتر دارد را انتخاب می‌کنیم و به پدرش می‌رویم، آن قدر این کار را انجام می‌دهیم تا دو رأس به هم برسند.

می دانیم در درخت بین هر دو رأس یک مسیر وجود دارد پس این کوتاه ترین مسیر است. (هر رأس در این مسیر حداکثر یکبار دیده شده است.)

ب)

در بدترین حالت هر دو رأس u و v دو برگ با عمق $\log^n$ و در دو طرف چپ و راست ریشه هستند در این حالت برای هر رأس یکبار تا ریشه رفتیم $(\log^n)$ و یکبار برای رسیدن به رأس مشترک تا ریشه رفتیم $(\log^n)$ پس:

$$T(n) = O\left(2(\log^n + \log^n)\right) = O(4\log^n) = O(\log^n)$$

```
void findway(u, v, tree){
    Node* tempu = u , tempV = V;
    int deepU = 0 , deepV = 0;
    while (tempu != tree.root) { tempu = tempu.p; deepU++;}
    while (tempV != tree.root) { tempV = tempV.p; deepV++;}
    while (u != V){ if (deepU > deepV){ deepU--;
    u = u.p;} else{ deepV--; V = V.p;} }
}
```

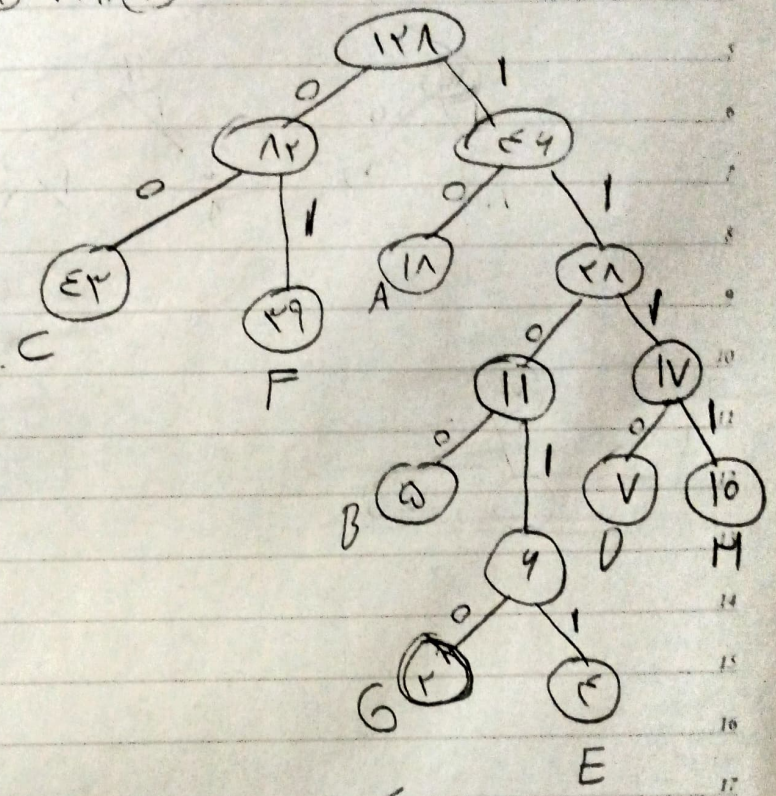
omid 20:10

نام و نام خانوادگی: امیرالمان کد دانشجویی: ۸۱۰۱۰۱۰۱

Subject: Year. Month. Date.

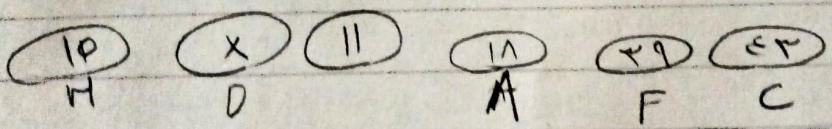
جواب سوال ۳:

A: (۱۸) B: (۵) E: (۴) G: (۲)
C: (۴۳) D: (۷) F: (۳۹) H: (۱۰)



C: ۰۰
F: ۰۱
A: ۱۰
B: ۱۱۰۰
D: ۱۱۱۰
H: ۱۱۱۱
G: ۱۱۰۱۰
E: ۱۱۰۱۱

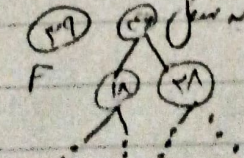
ب) برای آن که کد D تغییر کند باید در زمان انتخاب D هنوز هم حرفی از دو حرف کم تعداد باشد، در زمان انتخاب D داریم:



$۶ \le X < ۱۱$ ⟶ $۶ \le D < ۱۱$

در مرحله قبل انتخاب نشود

به خونه بعدی نرود

ج) قبل انتخاب C درخت به شکل ⟶ 

پس اگر بیشتر از ۴۶ شود ۳۹ با ۴۶ ترکیب می‌شود و ارتفاع درخت بیش‌تری می‌شود پس حداقل ۴ واحد باید اضافه کرد.

(حداقل ۴ واحد)

نام و نام خانوادگی: مهراد لویان

کد دانشجویی: ۸۱۰۱۰۱۵۰۱

Subject: Year. Month. Date.

جواب سوال ۲:

الف)

از یک درخت max-heap و یک درخت min-heap استفاده می کنیم. برای اضافه کردن عنصر جدید، اگر عنصر جدید از ریشه min-heap بزرگتر باشد آن را به min-heap اضافه می کنیم و در غیر آن صورت آن را به max-heap اضافه می کنیم.

اگر بعد از اضافه کردن یک عنصر به یکی از درخت ها، تعداد عناصر آن دو تا بیش تر از درخت دیگری شد، ریشه درخت با اعضا بیش تر را حذف کرده و به آن درخت دیگر اضافه می کنیم.

بدین صورت همیشه حداکثر اختلاف دو درخت یک عنصر هست پس $\frac{n}{2}$ اعضا در هر کدام از آن ها قرار دارد.

اگر n فرد بود میانه ریشه درخت با اعضا بیشتر می شود.

اگر n زوج بود، میانه برابر میانگین ریشه های دو درخت می شود.

max-heap	min-heap
$\left[\frac{n}{2}\right]$	$\left[\frac{n}{2}\right]$

چون ریشه max-heap از تمام اعضا خود بزرگتر است و از همه اعضا minheap کوچکتر هست پس دقیقا عنصر میانه هست. همین طور ریشه درخت min-heap از همه درخت خودش کوچکتر و از تمام اعضا درخت عناصر max-heap بزرگتر است پس آن هم میانه است.

ب)

insert و delete از مرتبه $O(\log n)$ هست و دسترسی به ریشه ها از مرتبه $O(1)$ هست پس به طور کلی پیچیدگی ما از مرتبه $O(\log n)$ هست.

$$T(n) = O(\log n)$$

نام و نام خانوادگی: مراد کیوان

کد دانشجویی: ۸۱۰۱۰۱۵۰۱

Subject: Year. Month. Date.

ادامه جواب سوال ۴:

روشی دیگر برای پیدا کردن میانه آن است تا یک درخت داشته باشیم و عناصر جدید را در آن اضافه کنیم و در هر مرحله که میانه را بخواهیم، می توانیم آن درخت را به صورت inorder در یک آرایه ذخیره کنیم و برای بدست آوردن میانه اگر n فرد باشد عنصر $\left[\frac{n}{2}\right]+1$ ام میانه می شود و اگر زوج شود میانگین عضو $\left[\frac{n}{2}\right]$ ام و $\left[\frac{n}{2}\right]+1$ ام می شود.

پیچیدگی insert از مرتبه $O(\log n)$ و پیمایش inorder از مرتبه $O(n)$ هست پس پیچیدگی این الگوریتم از مرتبه $\Theta(n)$ هست.

$$T(n) = O(n)$$