بسمه تعالی

گزارش تمرین ۶ سوالات تحلیلی

ساختمان داده

محسن کربلائی امینی، ۹۸۲۴۲۱۲۸

دی ۱۴۰۲

سوال ۱:

- در صورتیکه هر گره درخت، تعداد عناصر سمت چپ خود را نگه دارد(برای مثال در lefts)، برای یافتن شماره عنصر مورد نظر، در هنگام پیمایش، هر موقع که به سمت راست یک عنصر حرکت می‌کنیم، مقدار اولیه index را با lefts آن گره جمع می‌کنیم و در index ذخیره می‌کنیم. به این ترتیب لازم هست در هنگامی که insert اتفاق می‌افتد، اگر به سمت چپ قرار هست حرکت کنیم، مقدار lefts آن گره را increment کنیم.
- با استفاده از همان روش قسمت اول، می‌توان در طول درخت پیمایش کرد و در صورت بیشتر بودن مقدار x از lefts، به راست حرکت کنیم و lefts را از x کم می‌کنیم. دوباره به همین ترتیب در درخت پیمایش می‌کنیم تا دیگر پیمایشی لازم نباشد.

سوال ۲:

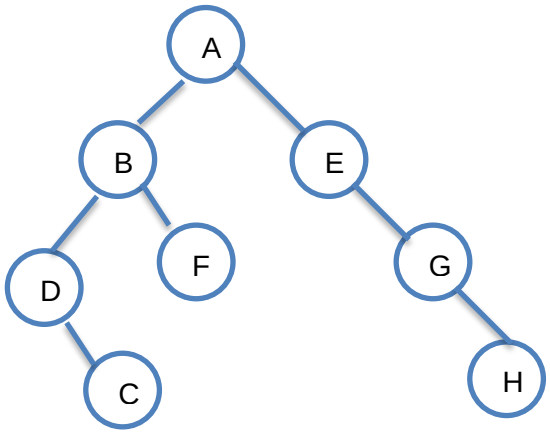
```
Node {
    int data;
    Node right;
    Node left;
    Node parent;

}
firstNext(Node root, int x) {
    if (root.data == null) {
        print("empty");
    }
    if (x == root.data)
        return root.data;

    if (x < root.data)
        if (root.left == null)
            return root.data;
        else
            return firstNext(root.left, key);
    else if (x > root.data)
        if (root.right == null)
            print ("doesn't exist");
        else
            return firstNext(root.right, key);
    return root;
}
```

پیدا کردن `firstNext(l)` و `firstNext(r)` هر کدام به اندازه $O(h)$ پیچیدگی خواهد داشت. با کمک گرفتن از تابعی مانند `findByIndex` در سوال قبل نیز میتوانیم جایگاه هر کدام از این عناصر را در درخت با پیچیدگی $O(h)$ بیابیم. بنابراین با تعداد محدودی عملیات با پیچیدگی $O(h)$ می‌توان تعداد این عناصر را به دست آورد.

سوال ۳:

با داشتن یک تابع به نام `listParents()` می‌توانیم تمام والدهای یک راس را پیدا کنیم. این تابع این کار را در بد ترین شکل یک درخت که ممکن است به شکل یک آرایه و در طول باشد، در $O(n)$ انجام می‌دهد. به این شکل که در طول حرکت درخت به سمت یک راس، از هر راسی که عبور کند، آن را در لیست خروجی قرار می‌دهد. به این ترتیب والدهای هر دوی این رئوس در $O(n)$ پیدا می‌شوند. سپس والدهای مشترک از این دو لیست در حداکثر $O(n)$ خارج شده و با توجه به داشتن فاصله این رئوس تا راس ریشه، می توان بهترین جد مشترک را بدست آورد. در نتیجه با انجام چند فرایند محدود در $O(n)$ عملیات یافتن بهترین جد مشترک در $O(n)$ انجام می‌شود.

سوال ۵:



روش بدست آوردن درخت به این صورت است: در پیمایش `preorder`، نخستین گره، گره ریشه است. بنابراین `A` ریشه این درخت می باشد. با پیداکردن گره ریشه در `inorder`، گره های دو طرف ریشه مشخص میشود. یعنی در ابتدا گره های `B, D, C, F` گره های سمت چپ و `E, G, H` در سمت راست ریشه قرار دارند.

همین کار را با حذف هر ریشه به صورت بازگشتی روی عناصر هر سمت انجام می‌دهیم. یعنی برای هر سمت یک `preorder` و یک `inorder` بدست می‌آید و در هر مرحله یک گره ریشه انتخاب و نوشته می شود.