تمرین شماره ۱

دانشکده مهندسی برق و کامپیوتر

ساختمان داده – بهار ۱۳۹۹

استاد : فتحیه فقیه

مهلت تحویل : ۱۳۹۸/۱۲/۱۷
ساعت ۸ صبح

مسئول تمرین : شراره نوروزی

به نکات زیر توجه کنید

- در تمامی پرسش‌ها اگر پاسخ شما طبق پیچیدگی مدنظر نباشد حداکثر ۶۰٪ نمره‌ی آن را دریافت خواهید کرد.
- پاسخ‌های خود را در سایت درس آپلود کنید. سعی کنید پی‌دی‌اف پاسخ‌ها خوانا باشند، در غیر این‌صورت برگه‌ی شما تصحیح نمی‌شود.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

۱. پیچیدگی زمانی الگوریتم‌های زیر را به دست آورید.

a)
```
for (int i = 0; i < n; i+=2)
    for (int j = 0; j < i; j+=2)
        print(“*”);   //O(1)
```

b)
```
int i = 1;
while (i < n){
    i++;
    int j = 1;
    while (j < n){
        j *= 2;
        int k = 1;
        while (k < n){
            k *= 5;
        }
    }
}
```

c)
```
for (int i = 0; i < n; i++)
    for (int j = n; j > 1; j--)
        for (int k = 0; k < j; k+= j)
            print("*") // O(1)
```

d)
```
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j+=i)
        print("*"); //O(1)
```

۲. فرض کنید $f(n)$ و $g(n)$ توابع مجانبی (نامنفی) هستند. ثابت کنید:

a) $\max(f(n), g(n)) = O(f(n) + g(n))$

b) $f(n) + g(n) = \Omega(\min(f(n), g(n))$

c) $\max(f(n), g(n)) = \Omega(f(n) + g(n))$

۳. پیچیدگی روابط بازگشتی زیر را از روش ممکن (جای گذاری، درخت بازگشت، تغییر متغیر، قضیه‌ی اصلی) به دست آورید.

a) $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + O(1)$

b) $T(n) = 9T\left(\frac{n}{3}\right) + n$

c) $T(n) = 2T(n-1) + O(1)$

d) $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$

۴. رابطه‌ی بازگشتی قطعه کدهای زیر را به دست آوردید و پیچیدگی زمانی آن را محاسبه کنید.

a)
```
rec_fib(n){
    if (n < 2)
        return 1;
    return rec_fib(n - 1) + rec_fib(n - 2)
}
```

b)
```
DFS(Tree T, int n){  // n is height of tree, and b is branching factor
    if (n < 1)
        return;

    print(T.name)
    if (n == 1)
        return;

    for (int i = 0; i < b && i < T.children.size(); i++)
        DFS(T.children[i], n - 1);

}
```

ب) برای تابع زیر

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + 1 & if\ n\ is\ even \\ 2T\left(\frac{n-1}{2}\right) & if\ n\ is\ odd \end{cases}$$

- نشان دهید نامتناهی n وجود دارد به طوری که $T(n) = O(\log n)$
- نشان دهید نامتناهی n وجود دارد به طوری که $T(n) = \Omega(n)$

۵. الف) جست‌وجوی دودویی[1] روشی برای جست وجو در آرایه‌های مرتب شده است. که در آن هربار عدد مربوطه را با عنصر میانی آرایه مقایسه می‌کنیم و اگر برابر بود جواب را برمی‌گردانیم، در غیر این‌صورت، اگر عدد بزرگ‌تر از عنصر میانی بود در نیمه‌ی سمت راست آرایه و در غیر این‌صورت در نیمه‌ی سمت چپ آرایه به دنبال آن می‌گردیم. شبه‌کدی بنویسید که محل مناسب برای درج یک عنصر را جست و جو و ایندکس آن‌را برگرداند.(با روش جست و جوی دودویی) و پیچیدگی را محاسبه کنید.

ب) در مرتب‌سازی درجی، در هرمرحله عنصر A[i] در محل مناسب خود در آرایه‌ی مرتب شده‌ی A[1...i-1] قرار می‌گیرد. الگوریتم را به‌شیوه‌ای تغییر دهید که پیچیدگی زمانی آن در بدترین حالت کاهش پیدا کند.

۶. الگوریتمی بهینه طراحی کنید که تعداد نابه‌جایی های موجود در یک آرایه را به‌دست آورد. شبه‌کد[3] آن را بنویسید. نابه‌جایی برای عنصر iام در آرایه A تعداد عناصری است با شمارنده jکه : $i < j.\ A[i] > A[j]$

۷. الگوریتمی طراحی کنید که در دو آرایه‌ی از پیش مرتب‌شده به طول n , m عنصر kام در آرایه‌ی مرتب حاصل از ادغام این دو آرایه را با پیچیدگی‌های زیر پیدا کند.

الف) $O(n + m)$

ب) $O(n \log m + m \log n)$

ج) $O(\log n + \log m)$ *امتیازی*

[1] Binary Search

[2] Inversion

[3] Pseudo Code