طراحی الگوریتم - تکلیف اول

موعد تحویل ۲۹ اسفند ۱۳۹۹

پیش از حل سوالات به موارد زیر دقت کنید:

- پاسخ تکلیف را به صورت یک فایل PDF آماده کنید و با نام {Student Number}_HW1 در سامانه آپلود کنید.
- در تحویل تکالیف به زمان مجاز تعیین شده دقت نمایید. موعد تکالیف قابل تمدید نمی‌باشند.
- در صورتی که مجموع تاخیر کل تکالیف شما کمتر از ۲۴ ساعت باشد نمره‌ای از شما کسر نمی‌گردد. در غیر این صورت به ازای هر روز تاخیر درصدی از نمره تکلیف شما کسر می‌گردد.
- پاسخ تکالیف را حتما در سامانه آپلود کنید و از ارسال تکالیف به ایمیل یا تلگرام اکیدا خودداری نمایید.
- در صورت وجود شباهت غیر قابل اغماض نمره‌ای به سوال تعلق نمی‌گیرد.
- در صورت وجود هرگونه ابهام می‌توانید در گروه تلگرام یا گروه اسکایپ سوالات خود را مطرح کنید.
- از طریق ایمیل زیر می‌توانید با ta مربوط به این تکلیف در ارتباط باشید.

– mroghani+algo@ec.iut.ac.ir

سوال ۱. (۲۰ نمره)

آ) به طور خلاصه تابع زیر را از نظر زمانی تحلیل کنید:

```
۱  void f( int n, int m ) {
۲      long long sum = 0;
۳      for (int i = 2; i < n; i *= 3) {
۴          for (int j = 0; j < m; j += 2) {
۵              for (int z = 0; z < j; z++) {
۶                  sum += 1;
۷              }
۸          }
۹      }
۱۰     cout << sum;
۱۱ }
```

ب) پیچیدگی زمانی برنامه زیر را تحلیل کنید (دقت کنید که تابع f تابع قسمت الف است):

```
۱  int main() {
۲      int a;
۳      cin >> a;
۴      for (int i = 0; i < a; i++) {
۵          f(1 << i, i); \\ "<<" is shift operation, therefore "1 << i" is
               the ith power of two.
۶      }
۷      return 0;
۸  }
```

سوال ۲. (۲۵ نمره) به ازای هر زوج از توابع $f(x)$ و $g(x)$ با توضیح مختصری نشان دهید که تابع $f(x)$ از $\mathcal{O},\ o,\ \omega,\ \Omega,\ \Theta$ تابع $g(x)$ هست یا خیر. (فرض کنید که c یک عدد ثابت بزرگتر از ۱ است.)

$f(x)$	$g(x)$	$\mathcal{O}$	o	ω	Ω	Θ
n^k	c^n					
2^n	$2^{n/2}$					
$\log n!$	$\log n^n$					
2^n	2^{n-2}					
$n2^n$	3^n					

سوال ۳. (۲۰ نمره) درستی یا نادرستی گزاره‌های زیر را مشخص کنید: (ادعای خود را اثبات کنید.)

آ) $n = \mathcal{O}(n \log n)$

ب) برای تمام $f(n), g(n) \geq 0$ ، اگر $f(n) = \mathcal{O}(g(n))$ آنگاه $2^{f(n)} = \mathcal{O}\left(2^{g(n)}\right)$

ج) $f(n) + g(n) = \Theta\left(max\{f(n), g(n)\}\right)$

د) $1 + c + c^2 + \cdots + c^n = \Theta\left(c^n\right),\ c > 1$

سوال ۴. (۱۰ نمره) روابط بازگشتی زیر را حل کنید. به عنوان مثال هر کدام را به صورت $T(n) = \mathcal{O}(f(n))$ برای کوچک‌ترین کلاس $f(n)$ بیان کنید و ادعای خود را به طور خلاصه توضیح دهید.

آ) $T(n) = 7T(n/2) + \Theta(\sqrt{n})$

ب) $T(n) = 4T(n/2) + n^2 \log n$

سوال ۵. (۲۵ نمره) می‌خواهیم nامین عدد فیبوناچی را با یک الگوریتم سریع‌تر از $\mathcal{O}(n)$ به دست آوریم. در این الگوریتم از ماتریس‌ها استفاده می‌کنیم.

ابتدا از نوشتن معادله‌های $F_1 = F_1$ و $F_2 = F_1 + f_0$ در فرم ماتریسی شروع می‌کنیم:

$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

به طور مشابه:

$$\begin{pmatrix} F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^2 \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

و به طور کلی داریم:

$$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

ماتریس $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ را X می‌نامیم.

آ) نشان دهید که برای محاسبه X^n به $\mathcal{O}(log\ n)$ ضرب ماتریسی نیاز است. (راهنمایی: سعی کنید X^8 را با کمترین تعداد ضرب ماتریسی محاسبه کنید.)

ب) اگر ضرب دو عدد nـبیتی $M(n)$ عملیات نیاز داشته باشد(مثلا $\mathcal{O}(n^2)$)، نشان دهید که زمان اجرای الگوریتم معرفی شده $\mathcal{O}\left(M(n)log\ n\right)$ است.

ج) نشان دهید که زمان اجرای الگوریتم معرفی شده $\mathcal{O}\left(M(n)\right)$ است. فرض کنید $M(n) \geq 2M(\frac{n}{2})$ (راهنمایی: طول اعداد در هر مربع شدن دو برابر می‌شود)

(برای علاقه‌مندان) در نهایت اشاره به این نکته که یک فرمول برای برای محاسبه nامین عدد فیبوناچی وجود دارد خالی از لطف نیست:

$$F_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n$$

پس به نظر می‌رسد برای محاسبه nامین عدد فیبوناچی فقط باید چند عدد را به توان n برسانیم. اما مشکل اینجاست که این اعداد گنگ هستند و به توان رساندن آن‌ها با دقت مناسب بدیهی نیست. در واقع راه ماتریسی معرفی شده می‌تواند یک راه میانبر برای محاسبه توان nام این اعداد گنگ باشد. (مقادیر ویژه ماتریس X چقدر است؟)