طراحی الگوریتم ها – تمرین سری اول

موعد تحویل تمرین : ۱۶ اسفند ۱۴۰۲

پیش از حل سؤالات به موارد زیر دقت کنید:

- پاسخ خود را به صورت یک فایل PDF آماده کنید و با نام HW1_NAME_STDNUM.pdf در سامانه آپلود کنید. (به جای NAME ، فقط نام خانوادگی و به جای STDNUM ، شماره دانشجویی قرار بگیرد و حتماً رعایت شود.)
- در تحویل تکالیف به زمان مجاز تعیین شده دقت نمایید. ارسال های با تاخیر مورد بررسی قرار نمی گیرند.
- پاسخ تکالیف را حتماً در سامانه آپلود کنید و از ارسال تکالیف به ایمیل یا تلگرام اکیداً خودداری نمایید.
- در صورت وجود شباهت واضح، نمره ای به پاسخ تعلق نمی گیرد.
- در صورت وجود هرگونه ابهام می توانید از طریق ایمیل سؤالات خود را با TA مطرح کنید.
- از طریق ایمیل زیر می توانید با TA مربوط به این تکلیف در ارتباط باشید.
  - taheri.a@ec.iut.ac.ir

سوال ۱: توابع زیر را براساس پیچیدگی زمانی مرتب نمایید.

$$n4^n\ ,n!\,2^n\ ,\binom{100}{n}\ ,logn^{logn}\ ,\ log^{logn}n\ ,logn!\ ,2^{\frac{n}{2}}\ ,(\frac{3}{2})^n$$

$$n^3(\frac{5}{4})^n\ \ ,\sqrt{2}^{logn}\ ,n^3\ ,\sqrt{2}^{n^3}\ ,n^n\ ,n^2logn\ ,loglogn\ ,n!\ ,2^{(2e+10)^{10e}}$$

$$4^{logn}\ ,n^2 2^n\ ,e^n$$

پاسخ: $\binom{100}{n} < 2^{(2e+10)^{10e}} < loglogn < logn^{logn} < \sqrt{2}^{logn} <$
$logn! < 4^{logn} < n^2logn < n^3 < log^{logn}n < n^3(\frac{5}{4})^n < 2^{\frac{n}{2}} < (\frac{3}{2})^n <$
$n^2 2^n < e^n < n4^n < n! < n!\,2^n < n^n < \sqrt{2}^{n^3}$

سوال ۲: به ازای هر زوج تابع f(x) و g(x) مشخص کنید که تابع f(x) از O ، o، ω، Ω و Θ تابع g(x) هست یا خیر(c و k اعدادی ثابت و بزرگ تر از ۱ هستند.)

پاسخ:

f(x)	g(x)	o	O	ω	Ω	Θ
$\log n$	$log^3 n$	√	√	✗	✗	✗
2^n	$2^{n/2}$	✗	✗	√	√	✗
$log^{logn}n$	n^3	✗	✗	√	√	✗

n^n	$n!$	✗	✗	√	√	✗
$\mathrm{nlog}^3\, n$	$n^2 loglogn$	√	√	✗	✗	✗
$log\, n^2$	$logn$	✗	√	✗	√	√
$n2^n$	e^n	√	√	✗	✗	✗
n^k	c^n	√	√	✗	✗	✗

سوال ۳: گزاره های زیر را اثبات یا رد کنید(برای گزاره غلط تنها مثال نقض کافی است، و برای عبارت درست باید آنرا اثبات کنید.)

$$f(n) \in O(g(n)) \Rightarrow 2^{f(n)} \in O(2^{g(n)})$$

$$\log n \in O(\sqrt[3]{n})$$

$$f(n) \in o((f(n)^2))$$

$$f(n) + o(f(n)) \in \theta(f(n))$$

$$f(n) \in O(s(n)), g(n) \in O(r(n)) \implies \frac{f(n)}{g(n)} \in O\left(\frac{s(n)}{r(n)}\right)$$

پاسخ:

الف: نادرست است برای مثال اگر g(n) برابر $n/2$ و f(n) برابر n باشد.

ب: صحیح است، برای مثال میتوان با استفاده از هوپیتال(حتما باید به بی نهایت میل کند) درستی عبارت را نشان داد.

ج: نادرست است برای مثال اگر f(n) تابعی نزولی مانند n^{-1} باشد.

د: درست است، برای اثبات میتوان از هوپیتال یا یافتن ضرایب برای هر دو طرف معادله استفاده نمود.

ه: غلط است، برای مثال اگر s(n) و r(n) برابر n^4 باشد، و f(n) برابر n^3 باشد و g(n) برابر n^2 باشد.

سوال ۴: برنامه های زیر را از لحاظ پیچیدگی زمانی برسی کنید(با دلیل و توضیحات کامل)

پاسخ: $O(a^4)$

```
#include <bits/stdc++.h>
using namespace std;
void f(int n, int m)
{
    long long sum = 0;
    for (int i = 2; i < n; i *= 3)
    {
        for (int j = 0; j < m; j += 2)
        {
            for (int k = 0; k < j; k++)
            {
                sum += 1;
            }
        }
    }
    printf ("%d\n", sum);
}
int main ()
{
    int a;
    scanf ("%d", &a);
    for (int i = 0; i < a; i++)
    {
        f(1 << i, i);
    }
}
```

اوردر حلقه اول لوگاریتم n در مبنای ۳ هست(مبنا برای ما اهمیتی ندارد) و دو حلقه درونی از اردر m^2 است، حلقه درون تابع main هم هر بار در حال شیفت به چپ(ضرب در ۲) هست و اوردر آن برابر 2^i هست، با جایگذاری 2^i بجای n و i به جای m، به یک عبارت از اوردر i^3 میرسیم و چون این کار در یک حلقه انجام میشود که تا a جلو میرود (در اصل عبارت i^3 زیر یک سامیشن هست) از اوردر a^4 میشود.

```
#include <bits/stdc++.h>
using namespace std;
int f(int num)
{
    if (num <= 1) return num;
    return f(num - 2) + f(num - 1);
}
int main ()
{
    int num; cin>>num;
    cout << f(num);
}
```

پاسخ: با حل عبارت T(n-1) + T(n-2) + 1 به اوردر 2^n میرسیم

```
#include <bits/stdc++.h>
using namespace std;

int main()
{

    int n; cin >> n;
    for (int i = 2; i <= n; i=pow(i,2)) {
        cout << 1 <<"\t";
    }
    return 0;
}
```

پاسخ: در هر بار اجرا i به توان ۲ میرسد برای مثال میتوان نوشت:

$$(((2^2)^2)^2 \dots)^2$$

اگر فرض کنیم حلقه بالا k بار اجرا شده است(k عددی است که دنبال آن می گردیم)، با دو بار لوگاریتم گرفتن از طرفین داریم:

$$((2)^2)^k = n \quad => \quad 2^k = logn \quad => k = loglogn$$

```cpp
#include <bits/stdc++.h>
using namespace std;

void f(int n){
    int r = 0;
    for(int i = 1 ; i < n ; i++)
        for(int j = 1 ; j < i ; j++)
            for(int k = j ; k < i + j ; k++)
                r++;
    cout<<r;
}
int main()
{
    int n ; cin>>n;
    f(n);
}
```

پاسخ: از اوردر $O(n^3)$ است.