به نام خدا

تمرین‌های فصل اول — **دکتر مرجان کائدی**

دانشکده مهندسی کامپیوتر دانشگاه اصفهان — **نیمسال دوم ۱۳۹۹-۱۴۰۰**

---

۱. توابع پیچیدگی زیر را از لحاظ سرعت رشد مرتب کنید.

$2^{\log n}$, e^n, $\log n^2$, $100n^n + \pi$, e^{20}, $10n\sqrt{n}$, $n^{15} + 20n + 25\log n$,
$n! + n$, $(\log n)^2$, $12(\log n)!$, $\frac{1}{5}n^{(\frac{1}{\log n})}$, $\ln(\ln n)$, $\sqrt{\log n}$

۲. صحت روابط ۳ تا ۶ را مانند موارد نمونه‌های حل شده، اثبات کنید.

توجه: توابع پیچیدگی را صعودی در نظر بگیرید.

1) $(n + \log n)^3 \in \Theta(n^3)$ 2) $n\sqrt{n} \in O(n^n)$

1. $c_1 \times n^3 \le (n + \log n)^3 \le c_2 \times n^3 \; for \; n \ge N \xRightarrow{c_1=1,\; c_2=2,\; N=0}$

$n^3 \le n^3 + (\log n)^3 + 3n^2 \log n + 3n(\log n)^2 \le 2n^3$ *always true*

2. $n\sqrt{n} \le c \times n^n \; for \; n \ge N \xRightarrow{c=1,\; N=2} n^{1.5} \le n^n \Rightarrow 1.5 \le n$ *always true*

3) $\sum_{i=1}^{n}\sum_{j=1}^{i} j \in \Theta(n^3)$ 4) $f(n) + o(f(n)) \in \Theta(f(n))$

5) $f(n) + g(n) \in O(\max(f(n), g(n)))$ 6) $f^2(n) \in \Omega(f(n))$

۳. با فرض مثبت بودن توابع f و g، درستی یا نادرستی روابط زیر را مشخص کنید. (برای درستی بیان اثبات و برای نادرستی تنها ارائه‌ی یک مثال نقض کافی است.)

راهنمایی: برای در نظر گرفتن توابع، همه‌ی توابع اعم از صعودی و نزولی را درنظر بگیرید؛ نه صرفا توابع صعودی.

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

3) $f(n) + g(n) \in \Theta(\min(f(n), g(n)))$ 4) $f(n) \in \Theta(f(\frac{n}{2}))$

5) $f(n) \in O(g(n)) \Rightarrow \log(f(n)) \in O(\log(g(n))$
$where \; \log(g(n)) \ge 0, f(n) > 0 \; and \; f(n) \ge 1 for \; all \; sufficiently \; large \; n$

به عنوان نمونه، حل مورد اول نشان داده شده است:

1. *false*, $if \; f(n) = 2n \; and \; g(n) = n \Rightarrow O(2^{2n} = 4^n) \notin O(2^n)$

۴. الگوریتم A و B دقیقا $T_A(n) = 0.1n^2 \log n$ و $T_B(n) = 2.5n^2$ میکرو ثانیه برای مسئله‌ای با سایز n طول می‌کشند.

الف) از بین این دو الگوریتم، الگوریتمی را انتخاب کنید که از نظر "O بزرگ" بهتر باشد.

ب) کوچک‌ترین n_0 ممکن را پیدا کنید که به ازاء هر $n \geq n_0$ یک الگوریتم از نظر زمان اجرا بهتر از دیگری باشد.

ج) اگر $n \leq 2^9$ باشد، کدام الگوریتم به‌صرفه‌تر است؟

۵. با حداقل چه تعداد ضرب می‌توان X^{64} را محاسبه کرد؟ (ایده‌ی حل را به صورت کامل توضیح دهید.)

۶. پیچیدگی زمانی تکه کدهای زیر را مانند نمونه حل شده، به دست آورید.

تکه کد:

```
array = mixed([i for i in range(1, n+1)])  #unsorted array of 1 to n
cnt = 0
for i in range(n):
    for j in range(i+1, n):
        if array[j] < array[i]:
            cnt += 1
```

حل:

- اگر i=0 باشد، آنگاه عمل اصلی $n-1$ بار انجام می‌شود.
- اگر i=1 باشد، آنگاه عمل اصلی $n-2$ بار انجام می‌شود.
- اگر i=2 باشد، آنگاه عمل اصلی $n-3$ بار انجام می‌شود.
- $\vdots$
- اگر i= $n-1$ باشد، آنگاه عمل اصلی 0 بار انجام می‌شود.

در نتیجه به ازاء هر i، عمل اصلی به اندازه $n-(i+1)$ بار انجام می‌شود.

حال پیچیدگی تکه کد را محاسبه می‌کنیم:

$$T(n) = \sum_{i=0}^{n-1} n-(i+1) = (n-1)+(n-2)+\cdots+1+0 = \frac{(n-1)\times(n)}{2} \Longrightarrow n^2$$

```
for i in range(n):
    j = 0
    while j <= n:
        print(i, j)
        j += n//50
```

A

```
i = n
while i >= 1:
    j = i
    while j <= n:
        print(i)
        j *= 2
    i //= 2
```

B

```
i = 2
while i <=n:
    print("*")
    i = i * i
```

C

```
for i in range(1, n+1):
    j = 1
    while j <= i:
        print("*")
        j *= 2
```

D

توجه: علامت "//"، علامت تقسیم با خارج قسمت صحیح است.

موفق باشید.