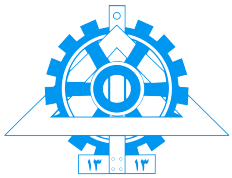

باسمه تعالی

دانشگاه تهران - دانشکده مهندسی برق و کامپیوتر

ساختمان‌داده‌ها و الگوریتم‌ها

تمرین اول - پیچیدگی و الگوریتم های بازگشتی

کوروش سجادی

تاریخ تحویل: ۱۴۰۲/۱۲/۲۷

۱. ۲۰ نمره

پیچیدگی زمانی هر یک از قطعه کدهای زیر را محاسبه کنید.

الف)

```
int i, j, k = 0;
for(i = n/2; i <= n; i++) {
    for(j = 2; j <= n; j = j * 2) {
        k = k + n/2;
    }
}
```

ب)

```
int i = n;
while(i > 1) {
    int j = 1;
    while(j < n) {
        j = j*5;
        i = i/3;
        cout << "*";
    }
}
```

ج)

```
void function(int n) {
    int count = 0;
    for(int i = 0; i < n; i++)
        for(int j = i; j < i*i; j++)
            if(j%i == 0) {
                for(int k = 0; k < j; k++)
                    print("*");
            }
}
```

۲. ۱۰ نمره

روابط زیر را رد یا اثبات کنید برای مورد آخر گزاره را بررسی کنید.

الف) $log_۲ f(n) \in \theta(log_۲ g(n)) \Rightarrow f(n) \in \theta(g(n))$

ب) $f(n) \in \Omega(g(n)),\ g(n) \in \Omega(h(n)) \Rightarrow f(n) \in \Omega(h(n))$

ج) $f(n) \in O(g(n))$ or $f(n) \in \Omega(g(n))$ or $f(n) \in \theta(g(n))\, ExactlyOneOfTheseRelationsOccur$

۳. ۲۰ نمره

پیچیدگی روابط بازگشتی زیر را با استفاده از روش‌های گفته شده به دست آورید.

الف) $T(n) = T(\sqrt{n}) + O(log(log(n)))$

ب) $T(n) = ۲۵T(n/۵) + n^۲$

ج) $T(n) = T(۳n/۴) + T(n/۴) + ۱/۲n^۲(۱ - sin(n))$

۴. ۱۰ نمره

توابع زیر را براساس پیچیدگی زمانی آنها مرتب کنید.

الف) $log(n)!, log(log * (n)), log * (log(n)), n^۹, ۵^n, n۲^n, ۱۰۰n^۷$

ب) $n^{n(log(log(n)))}, ۱۰^{۱۰۰}, \sum_{i=۰}^{n} \frac{n^i}{i!}, n^{۱۰۰}, ۱۰^۹ n, log(n), log * (n)$

ج) $\sum_{j=۱}^{n} \sum_{i=۱}^{j} i, n^{\frac{۱}{log(n)}}, \frac{۱}{۱۰۰} logn, ۱۰n, n^۴, nlog(n), n^n$

۵. ۲۰ نمره

پیچیدگی زمانی قطعه کد های زیر را با نوشتن رابطه بازگشتی آن محاسبه کنید.

الف)

```
int Sum(int n) {
    if(n == 1)
        return 1;
    return n + Sum(n-1);
}
```

ب)

```
int Find(int a[], int x) {
    switch (len(a)) {
        case 0:
            return 0;
        case 1:
            if(x <= a[0])
                return 0;
            return 1;
    }
    mid = 1 + (len(a) - 1) / 2;
    if(x <= a[mid -  1])
        return Find(a[:mid], x);
    return mid + Find(a[mid:], x);
}
```

ج)

```
void f(int A[]) {
    n = len(A);
    sq = sqrt(n) // square root
    for(i = 1; i < n; i++)
        cout << "*";
    if(n == 1)
        return;
    itr = 0;
    while(itr < n) {
        f(A[i:i+s])
        i += s;
    }
}
```

۶. ۱۰ نمره

توضیح دهید که چگونه میتوان پیچیدگی زمانی الگوریتمی که هم جزء iterative و هم جزء recursive دارد را محاسبه نمود سپس راه بیان شده خود را بر روی قطعه کد زیر اجرا نمایید .

```
int modifiedBSearch(arr, target):
    if (len(arr) == 0)
        return -1;

    mid = len(arr) / 2;
    if(arr[mid] == target)
        return mid;

    for(i = mid + 1; i < len(arr); i++)
        if(arr[i] == target)
            return i;

    return modifiedBSearch(arr[:mid], target);
```

۷. **۱۰ نمره**

تصور کنید سه پایه و تعدادی دیسک با اندازه‌های مختلف داریم که بر اساس اندازه روی یکی از این پایه‌ها قرار گرفته‌اند، به طوری که هیچ دیسک بزرگ‌تری بر روی دیسک کوچک‌تری قرار نگیرد. میخواهیم تمام دیسک‌ها را به پایه دیگر با استفاده از پایه واسطه ببریم با این شرط که فقط مجاز به جابجایی دیسک‌ها بین پایه‌های مجاور هستیم و نمی‌توانیم دیسک‌ها را مستقیماً از پایه اول به پایه سوم منتقل کنیم. همچنین، در هر حرکت تنها می‌توان یک دیسک جابجا کرد و همیشه باید قاعده دیسک کوچکتر روی دیسک بزرگتر را رعایت کرد. یک تابع بازگشتی برای این مسئله نوشته و پیچیدگی زمانی تابع خود را تحلیل کنید. (با نوشتن شبه کد نیز میتوانید آن را تحلیل کنید)