DATA STRUCTURES & ALGORITHMS TA SESSION

GROWTH COMPLEXITY

RECURRENCE RELATIONS

INSERTION SORT & MERGE SORT

پرسش یکم

توابع زیر را از نظر سرعت رشد با یکدیگر مقایسه کنید.

a) $\sqrt{\log n}, \log\log n$

b) $(\log n)!, n^{\log\log n}$

c) $\log\log n, \sqrt{n}$

d) $\log n!, n^2$

e) $\log * n, \log\log n$

f) $n!, n^n$

g) $\sum_{k=0}^{\infty} \frac{n^k}{k!}, 2^n$

a) $m = \log m\ ,\ \sqrt{m} > \log m \Rightarrow \sqrt{\log n} > \log\log n$

b) $n = 2^m\ , (\log 2^m)! = m!\ ,\ (2^m)^{\log\log(2^m)} = 2^{mlogm}$

اگر از هر دو log بگیریم (که صعودی است)

$$\log m! = \log 1 + \log 2 + \ \ldots + \log n$$
$$< \log n + \log n + \ \ldots + \log n \ \text{(n times)} = n \log n$$

c) $\log\log n < \log n < \sqrt{n}$

d) $\log n! = \log 1 + \log 2 + \ \ldots + \log n < n \log n < n^2$

e) $\log * n = \begin{cases} 0 \ if\ n \le 1 \\ 1 + \log* n \ \ if\ n > 1 \end{cases}$

f) $1 * 2 * 3 * \ldots * n < n * n * \ldots * n\ (n\ times)$

g) $e^n > 2^n$.

پرسش دوم

اگر آرایه s به طول n باشد، پیچیدگی زمان اجرای کد در بدترین و بهترین حالت را بنویسید.

```
s = list                                          c۱
result = list                                     c۲
while(s is not empty)                             c۳
  element = s.get                                 c۴
  if(element > 3)                                 c۵
    i = size of result                            c۶
    while(i != 0)                                 c۷
      if(result[i-1] > element)                   c۸
        result[i] = result[i-1]                   c۹
        i = i - 1                                 c۱۰
      else                                        c۱۱
        break                                     c۱۲
    endwhile                                      c۱۳
    result[i] = element                           c۱۴
  endif                                           c۱۵
endwhile                                          c۱۶
```

در صورتی که تمام اعداد از ۳ کوچکتر باشد پیچیدگی $o(n)$ است و در بدترین حالت تمام عناصر بزرگتر از ۳ و به صورت نزولی چیده شده اند که پیچیدگی $O(n^2)$ است.

پرسش سوم

پیچیدگی توابع زیر را به دست آورید. می‌توانید از هر روشی (قضیه‌ی اصلی، جایگذاری، رسم درخت، تغییر متغیر و ...) استفاده کنید.

a) $T(n) = T(n-1) + n$

b) $T(n) = 4T(\frac{n}{2}) + n$

c) $T(n) = 2T(\frac{n}{2}) + n^2$

d) $T(n) = T(\frac{3n}{7}) + T(\frac{4n}{7}) + n$

e) $T(n) = \sqrt{n}T(\sqrt{n}) + n$

جایگذاری $T(n) = O(n)$

قضیه‌ی اصلی – درخت بازگشت $T(n) = O(n^2)$

قضیه‌ی اصلی – درخت بازگشت $T(n) = O(n)$

درخت بازگشت $T(n) = O(n \log n)$

تغییر متغیر و جایگذاری $T(n) = O(n \log \log n)$

پرسش چهارم

رابطه‌ی بازگشتی و رابطه‌ی **صریح** پیچیدگی برای قطعه کدهای روبرو را به دست آورید.

```
int maxSubArr(int arr[], int l, int h) {
  if (l == h)
    return arr[l];
  int mid = (l + h)/2;
  return max(maxSubArr(arr, l, mid),
             maxSubArr(arr, mid+1, h),
             maxC(arr, l, mid, h)); // O(n) function
}
```

$$T(n) = 2T\left(\frac{n}{2}\right) + n = O(n \log n)$$

```
int sumDigit(int number, int num_d) {
  if (number == 0)
    return 0;
  return(number%10 + sumDigit(number/10, num_d-1));
}
```

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

پرسش پنجم

برای تابع پیچیدگی زمانی $T(n)$ که به صورت زیر تعریف می شود، یک فرم بسته پیدا کنید.

$$T(n) = \left(\frac{۲}{n}\right)\left(T(۰) + \cdots + T(n-۱)\right) + c$$

$$T(۰) = ۰$$

با امتحان کردن چند عدد اول به یک الگو می‌رسیم سپس می‌توانیم با استفاده از استقرا حدس را اثبات کنیم

$$T(1) = \frac{2}{1}\left(T(0)\right) + c = c$$

$$T(2) = \frac{2}{2}\left(T(0) + T(1)\right) + c = c + c = 2c$$

$$T(3) = \frac{2}{3}\left(T(0) + T(1) + T(2)\right) + c = \frac{2}{3}\ 3c + c = 3c$$

$$T(n) = cn$$

$$T(n+1) = \frac{2}{n+1}\ c(1 + 2 + \cdots + n) + c = \frac{2c}{n+1}.\frac{n(n+1)}{2} + c = c(n+1)$$

پرسش ششم

می‌خواهیم در یک آرایه با اندازه $۱۰^۷$، ۱۰۰۰ عدد بزرگتر را پیدا کنیم. راه حل خود را ارائه دهید.
فرض کنید برای یک آرایه به اندازه ۱۰۰، ۰٫۱ میلی‌ثانیه طول می‌کشد تا عنصر مورد نظر پیدا شود و همچنین ۰٫۱ میلی‌ثانیه طول می‌کشد تا آرایه ۱۰۰ تایی مرتب شود.
زمان مورد نیاز برای راه حل خود را محاسبه کنید.

زمان عملیات برای جست‌وجو $\frac{0.1}{100} = 0.001$

زمان عملیات برای مرتب سازی $\frac{0.1}{100 \log 100} \approx 0.0005$

می‌توانیم هربار بزرگترین عنصر را پیدا کرده و هزار بار این کار را تکرار کنیم تا ۱۰۰۰ عدد بزرگتر را پیدا کنیم.

$$T_a = 0.001 \,.\, 10^7 . 1000 = 10^7 \; ms$$

می‌توانیم کل آرایه را مرتب کنیم و هزار عنصر بزرگتر را از آخر آرایه برداریم

$$T_b = 0.0005 \,.\, 10^7 . 7 = 3.5 \times 10^3 \; ms$$