Introduction to Algorithm Design HW #1

20/10/2017

Deniz Can Erdem Yılmaz
151044001

1) • $\lim_{n\to\infty} \frac{T_4(n)}{T_6(n)} = \frac{\ln^2 n}{\sqrt[3]{n}} = \frac{(\ln n)^2}{n^{\frac{1}{3}}} = \sqrt[3]{\frac{(\ln n)^6}{n}} = 0 \Rightarrow T_4 = O(T_6)$

• $\lim_{n\to\infty} \frac{T_1(n)}{T_6(n)} = \frac{3n^4 + 3n^3 + 1}{\sqrt[3]{n}} = \frac{n^4}{n^{\frac{1}{3}}} = \infty \Rightarrow T_6 = O(T_1)$

• $\lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} = \frac{3n^4 + 3n^3 + 1}{3^n} = \frac{n^4}{3^n} = 0 \Rightarrow T_1 = O(T_2)$

• $\lim_{n\to\infty} \frac{T_2(n)}{T_5(n)} = \frac{3^n}{2^{2n}} = \frac{3^n}{4^n} = 0 \Rightarrow T_2 = O(T_5)$

• $\lim_{n\to\infty} \frac{T_3(n)}{T_5(n)} = \frac{(n-2)!}{2^{2n}} = \frac{(n-2)!}{4^n} = \infty \Rightarrow T_5 = O(T_3)$

$$\underbrace{\ln^2 n}_{\text{logarithmic}} < \underbrace{\sqrt[3]{n}}_{\text{cubic root}} < \underbrace{3n^4 + 3n^3 + 1}_{\text{quartic}} < \underbrace{3^n < 2^{2n}}_{\text{exponential}} < \underbrace{(n-2)!}_{\text{factorial}}$$

2)

```
def delicious:
  plum=float("inf")
  watermelon=0
  orange=0
  orangeTime=False
  while not orangeTime:
    for fruit in fruits:
      if fruit>watermelon:
        watermelon=fruit
      if fruit<plum:
        plum=fruit
        break
    else:
      orangeTime=True
  for fruit in fruits:
    if abs(fruit-(wm+plum)//2)<abs(orange-(wm+plum)//2)
      orange=fruit
  return orange
```

a) plum: Becomes minimum value of the array
watermelon: Becomes maximum value of the array
orange: Becomes nearest element to the avg of min and max values

- First for loop finds the min and max values
- It takes shif amount of iterations to find the values
- First for loop always breaks after first iteration (inf > anything)
- While lono makes for loop to work again after min value has set
- Second for loop checks distance of each element to our target
- Sets any element closer to the average as new orange
- Finally returns orange which is aveage of min & max

b) While loop iterets 2 times. First for loop iterates (shift amount + 1) times. Since we need notation depends on input size (n), first for loop takes (n+1) time when no shift has done. Which is worst case. Second for loop takes another (n) time to traverse each element. (n) doesn't effect completition time. So;

$\underline{f(n) = \Omega(n)}$ & $\underline{f(n) = O(n)} \Rightarrow n \leq f(n) \leq n$

$\Rightarrow \underline{f(n) = \Theta(n)}$

3) a) $\sum_{i=0}^{n-1} (i^2+1)^2 = \int_0^{n-1} (i^2+1)^2 di \le f(n) \le \int_1^{n} (i^2+1)^2 di$ (non-decreasing function)

$$= \frac{i^5}{5} + \frac{2i^3}{3} + i \Big|_0^{n-1} \le f(n) \le \frac{i^5}{5} + \frac{2i^3}{3} + i \Big|_1^{n}$$

$$= \quad n^5 \le f(n) \le n^5 \quad \Rightarrow f(n) = \theta(n^5)$$

b) $\sum_{i=2}^{n-1} \log i^2 = \int_2^{n-1} \log i^2 \, di \le f(n) \le \int_3^{n} \log i^2 \, di$ (non-decreasing function)

$$= 2i(\log i - 1)\Big|_2^{n-1} \le f(n) \le 2i(\log i - 1)\Big|_3^{n}$$

$$= n.\log n \le f(n) \le n.\log n \Rightarrow f(n) = \theta(n.\log n)$$

c) $\sum_{i=1}^{n} (i+1).2^{i-1} = \int_1^{n} (i+1).2^{i-1} di \le f(n) \le \int_2^{n+1} (i+1)2^{i-1} di$ (non-decreasing fn)

$$= \left(\frac{i^2}{2}+i\right).2^{i-1}\Big|_1^{n} \le f(n) \le \left(\frac{i^2}{2}+i\right)2^{i-1}\Big|_2^{n+1}$$

$$= n^2.2^n \le f(n) \le n^2.2^n \Rightarrow f(n) = \theta(n^2.2^n)$$

d) $\sum_{i=0}^{n-1}\sum_{j=0}^{i-1} (i+j) = \int_0^{n-1}\left(\int_0^{i-1}(i+j)dj\right)di \le f(n) \le \int_1^{n}\left(\int_0^{i-1}(i+j)dj\right)di$ (non-decreasing fn)

$$= \int_0^{n-1}\left(ji + \frac{j^2}{2}\Big|_0^{i-1}\right)di \le f(n) \le \int_1^{n}\left(ji + \frac{j^2}{2}\Big|_0^{i-1}\right)di$$

$$= \int_0^{n-1}\left(\frac{3i^2 - 4i + 1}{2}\right)di \le f(n) \le \int_1^{n}\left(\frac{3i^2 - 4i + 1}{2}\right)di$$

$$= \frac{i^3 - 2i^2 + i}{2}\Big|_0^{n-1} \le f(n) \le \frac{i^3 - 2i^2 + i}{2}\Big|_0^{n}$$

$$n^3 \le f(n) \le n^3 \Rightarrow f(n) = \theta(n^3)$$

```
def q3a(n):
    result=0
    for i in range(0,n):
        result += pow(pow(i,2)+1,2)
    return result
```

```
def q3d(n):
    result = 0
    for i in range(0, n):
        for j in range(0, i):
            result += i*j
    return result
```

4)
```
int fun(int n){
    int count = 0;
    for (int i=n; i>0; i/=2) → log2n time
        for (int j=0; j<i; j++) → i times
            count += 1;
    return count;
}
```

$$fun(n) = \sum_{i=1}^{\log_2 n} \sum_{j=0}^{2^i} 1$$

$$\int_1^{\log_2 n}\left(\int_0^{2^i} 1\,dj\right)di \le fun(n) \le \int_2^{\log_2(n)+1}\left(\int_0^{2^i} 1\,dj\right)di$$

$$\int_1^{\log_2 n}\left(j\,\Big|_0^{2^i}\right)di \le fun(n) \le \int_2^{\log_2 n+1}\left(j\,\Big|_0^{2^i}\right)di$$

$$\int_1^{\log_2 n} 2^i\,di \le fun(n) \le \int_2^{\log_2 n+1} 2^i\,di$$

$$\underbrace{\frac{2^i}{\log 2}\Big|_1^{\log_2 n}}_{\frac{2^{\log_2 n}}{\log 2} = \frac{n}{\log 2}} \le fun(n) \le \frac{2^i}{\log 2}\Big|_2^{\log_2 n+1}$$

$$n \le fun(n) \le n \Rightarrow fun = \Theta(n)$$

5) a) $\lim_{n\to\infty} \frac{3^{2n}}{n^3} = \frac{9^n}{n^3} = \infty$

(exponential > polynomial)

so $n^3 = O(3^{2n})$ is true

b) $\lim_{n\to\infty} \frac{\log(\log n)}{n} = 0$

(poly logharitmic < linear)

so $n = o(3^{2n})$ is false, it is $n = O(3^{2n})$

c) $\lim_{n\to\infty} \frac{n!}{(n\log n)^2} = \infty$

(factorial > linearithmic)

so $(n\log n)^2 = O(n!)$ is true

d) If $c_1 n \le \sqrt{10n^2+7n+3} \le c_2 n$ for $n \ge n_0$, $c_1>0$, $c_2>0$ it is true So;

- $c_1^2 n^2 \le 10n^2+7n+3 \le c_2^2 n^2$, $n^2 > 7n$, $n^2 > 3$
- $0 < 10n^2+7n+1 \le 10n^2+7n^2+n^2 = 20n^2 \cong 4n$
- for $c_1 = 1$, $c_2 = 4$ all values bigger than $n_0 = 2$ provides the equation
- $n \le \sqrt{10n^2+7n+3} \le 4n$, $n_0 = 2$ $\Big\}$ so $\sqrt{10n^2+7n+3} = \Theta(n)$ is true