18

Analysis and Design of Algorithms

CS2102
Analysis of Algorithms
2020-II

Submission deadline: December 10, 20:10

- Write your answers (images) inside the *answers* folder in order to generate a single PDF file. Replace the image files that are already included in the project. Do not change the file name.
- Read the questions carefully and write your answers clearly. Answers that are not legible and that doesn't follow the format will not have any score.

Outcomes:

a. Apply appropriate mathematical and related knowledge to computer science.

b. Analyze problems and identify the appropriate computational requirements for its solution.

Problem 1 (Outcomes a, b) - 4 points

Given the following expression:

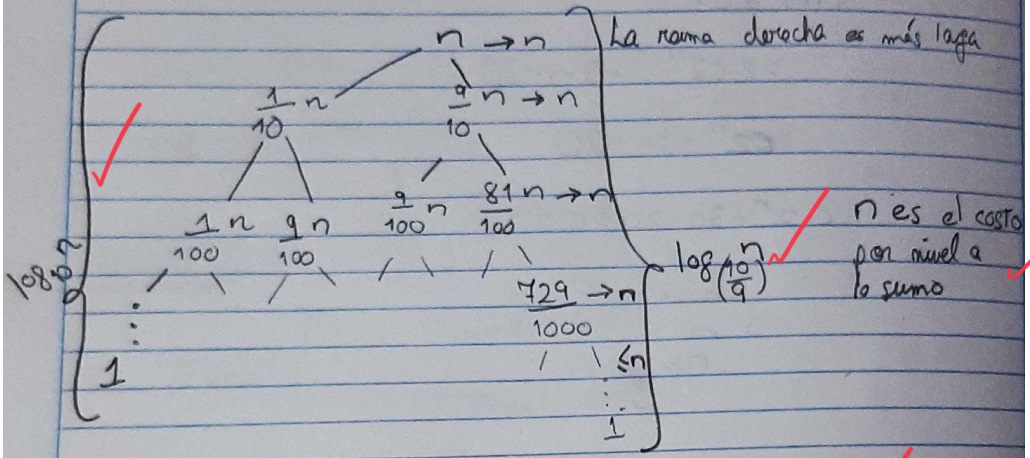
$$T(n) = T(n/10) + T(9n/10) + \Theta(n)$$

- Sketch the recursion tree of $T(n)$ and indicate the exact height, the cost per level and the total cost of the tree.
- Based on the recursion tree above estimate a good upper and lower bound of $T(n)$. If no bounds can be found or are the same, express that clearly and explain the reason.

4

1.

$T(n) = T(n/10) + T(\frac{9}{10}n) + \Theta(n)$



$$\sum_{i=1}^{\lg n} n = \boxed{n \lg n}$$

Costo total

La rama corta funciona como lower bound y la rama larga como upper bound

$T(n) = O(n \log_{\frac{10}{9}} n)$

$T(n) = \Omega(n \log_{10} n)$

Problem 2 (Outcome b) - 2 points

Consider that $T(n) = 0.5n^2 + 3n$ and for each of the items below indicate if the statement is true or false and justify the reason.

1. $T(n) = O(n)$
2. $T(n) = \Omega(n)$
3. $2^{n+1} \leq O(2^n)$
4. $T(n) = \Theta(n^2)$

2. $T(n) = 0.5n^2 + 3n$

a. $T(n) = O(n)$

$\lim_{n \to \infty} \frac{0.5n^2 + 3n}{n} = \infty$ inf

$T(n) = \omega(n)$ es decir es estrictamente menor por b que no puede ser mayor o igual $(O(n))$, es decir queda descartado que sea $T(n) = O(n)$

Falso

b. $T(n) = \Omega(n)$

$\lim_{n \to \infty} \frac{0.5n^2 + 3n}{n} = \infty$ inf

$T(n) = \omega(n)$ lo cual implica

$T(n) = \Omega(n)$ (Verdadero)

2

c. $2^{n+1} \leq O(2^n)$

$2^{n+1} = O(2^n)$

$\lim_{n \to \infty} \frac{2^n}{2^{n+1}} = \frac{1}{2}$ una constante

$2^{n+1} = \Theta(2^n)$

lo cual implica

$2^{n+1} = O(2^n)$

Verdadero

d. $T(n) = \Theta(n^2)$

$\lim_{n \to \infty} \frac{0.5n^2 + 3n}{n^2}$

$= 0.5 + 0$

$= 0.5$ una constante

por lo tanto

$T(n) = \Theta(n^2)$

Verdadero

Problem 3 (Outcome b) - 5 points

4

Consider a finite set of N points in the plane and:

- (3) Write a pseudocode of an $O(nlogn)$ divide-and-conquer algorithm to obtain the vertices of the smallest convex polygon containing all the given points.
- (1) Highlight the divide, conquer and combine steps on the proposed algorithm.
- (1) Explain the worst and best case of the algorithm and write its corresponding execution time.

(3) Convex hull (S, n)   n → cantidad mínima de puntos

```
C-h (S, n):
    if S.size ≤ n
        return S

    h_low = S[ :mid]      //divide
    h-high = S[mid:]      //divide

    h_low = C-h(h-low, n)   //conquer
    h-high = C-h(h_high, n)  //conquer
    return Merge-partial-hulls (h-low, h_high) //combine
```

This is only DC algorithm.

more details about the logic.

* Lógica análoga a Merge-sort
* Si se asume caso promedio $O(n)$ para Merge, no considera el caso todos contra todos, quedaría

$$T(n) = 2T(n/2) + O(n)$$

$$T(n) = \Omega(n \lg n)$$

* Cuando es todos contra todos los puntos significa que todos los puntos conforman el convex hull

$$n \times (n-1) \rightarrow O(n^2)$$

Problem 4 (Outcomes a, b) - 3 points

Solve the following recurrences using Master Method:

- $T(n) = 4T(n/2) + n^2\sqrt{n}$

- $T(n) = 3T(n/2) + n$
- $T(n) = T(\sqrt{n}) + log(n)$

Hint - You can transform last expression using change of variables: $n = 2^m$

3

4.

(a) $T(n) = 4T(\frac{n}{2}) + n^2\sqrt{n}$ $\quad \cdot\ n^{\log_2 4} = n^2$ $\quad n^{\frac{5}{2}} = \Omega(n^2)$

$= 4T(\frac{n}{2}) + n^{5/2}$ $\quad \cdot\ n^{5/2}$

Aplica M.M

$T(n) = \Theta(n^{5/2})$ $\quad d > \log_b a$

$T(n) = \Theta(n^2\sqrt{n})$ ✓

A

(b) $T(n) = 3T(\frac{n}{2}) + n$ $\quad \cdot\ n^{\log_2 3}$ $\quad n = O(n^{\log_2 3 - \varepsilon})$

$\cdot\ n$ $\quad \varepsilon \geq 0$

$\log_b a > d$

$T(n) = \Theta(n^{\log_2 3})$ ✓

A

(c) $T(n) = T(\sqrt{n}) + lg(n)$ $\quad$ Hint $n = 2^m$

$n = 2^m \rightarrow m = lg(n)$

$T(2^m) = T(2^{m/2}) + m$ $\quad T(m) = T(2^m)$

$T(m) = T(m/2) + m$ $\quad \cdot\ m^{\log_2 1} = 1$ $\quad m \Rightarrow \Omega(m^{0+\varepsilon})$

$\cdot\ m$ $\quad$ polinomicamente mayor $F(m)$

$T(m) = \Theta(m)$

volviendo a la convención inicial

$T(n) = \Theta(lg(n))$ ✓

A

alph

Problem 5 (Outcome a) - 3 points

Prove that $T(n) = 2T(\lfloor n/2 \rfloor) + n$ is $\Theta(nlogn)$

2

5. $T(n)=2T(\lfloor n/2 \rfloor)+n$ es $\Theta(n\lg n)$

- Probar que es $O(n\lg n)$

↳ $T(k) \le c\,k\lg k$

$T(n) \le 2c(\lfloor \frac{n}{2} \rfloor \lg(\lfloor n/2 \rfloor))+n$

$\le \cancel{2}c(n/\cancel{2}\, \lg(n/2)) + n$

$\le cn\lg(\frac{n}{2}) + n$

sale

$\le cn\lg n - cn + n$

$-cn+n \le 0$

$cn \ge n$

$c \ge 1$ cierra la inducción

$T(n)=O(n\lg n)$

- Probar que es $\Omega(n\lg n)$

$T(k) \ge ck\lg k$

$T(n) \ge 2c(\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor))+n$

$\ge \cancel{2}c(\frac{n}{\cancel{2}}\lg(n/2))+n$

sale

$\ge cn\lg n - cn + n$

$-cn+n \ge 0$

$n \ge cn$

$1 \ge c$

$T(n)=\Omega(n\lg n)$

$1 \ge c$ cierra la inducción

$T(n)=O(n\lg n) \wedge T(n)=\Omega(n\lg n) \to \boxed{T(n)=\Theta(n\lg n)}$

Problem 6 (Outcome b) - 3 points

Given an array of N points in the Euclidean plane, write a pseudocode of a divide-and-conquer algorithm that returns the pair of points with the smallest distance between them and write the execution time of your algorithm.

#Se asume que S está ordenado por x, S2 por y.

6.

```
Closest-points (S, low, high, S2):
    if(low = high) return INT_MAX

    else if(high-low = 1) return dist(S, low, high) //Euclidian distance

    mid = ⌊(low+high)/2⌋

    S_low = Closest-points(S, low, mid)

    S_high = closest_point(S, mid+1, high, S2)

    S_across = across(S_low, S_high, S, low, high, S2)

    return min(S_low, S_high, S_across)
```

3

```
Across (slow, s_high, S, S2, low, high):
    min_value = min(S_low, S_high) , values = [ ]
    for i = low to high:
        if(|S2[i].x - S[mid].x|) < min_value:
            values.push (S2[i])

    copy = min_value
    for i to values.size:
        temp = i + 1
        while temp < values.size and (values[temp].y - values[i].y)
                                         < min_value:

            min_value = dist(values[i], values[temp])
            temp += 1

    return min(copy, min_value)
```