Trabalho de Análise de Algoritmos

Pedro Correa - 11718563

1. Mostre que a solução $T(n) = T(\frac{n}{2}) + 1$ é $O(\log(n))$:

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

$$O(\log x)$$

$$T(n) \leq c * \log(n)$$
$$T(n) \leq T(\frac{n}{2}) + 1$$
$$T(n) \leq c * \log(\frac{n}{2}) + 1$$
$$T(n) \leq c * \log(n) - c * \log(2) + 1$$
$$T(n) \leq c * \log(n) - c + 1$$

Base: com $n = 4$

$$T(n) \leq c * \log(n) - c + 1$$
$$T(\frac{4}{2}) + 1 \leq c * \log(4) - c + 1$$
$$2 + 1 - 1 \leq 2 * c - c$$
$$2 \leq c$$

---

2. Mostre que a recorrência $T(n) = 8T(\frac{n}{2}) + n^2$ é tal que $T(n) = \Omega(n^3)$. Adote $T(1) = 1$

$$T(n) = 8 * T(\frac{n}{2}) + n^2$$
$$T(n) = \Omega(n^3)$$
$$T(1) = 1$$

$$T(n) \geq 8 * T(\frac{n}{2}) + n^2$$
$$T(n) \geq 8 * c * (\frac{n}{2})^3 + n^2$$
$$T(n) \geq \frac{8 * c * n^3}{8} + n^2$$
$$T(n) \geq c * n^3 + n^2$$

Hipotese: $T(n) \geq c * n^3 + n^2$

$n = 2$

$$T(2) \geq 8 * T(\frac{2}{2}) + 2^2$$
$$T(2) \geq 8 * T(1) + 4$$
$$T(2) \geq 8 + 4$$
$$T(2) \geq 12$$

$$T(2) \geq c * 2^3 + 2^2$$
$$12 \geq 8 * c + 4$$
$$12 - 4 \geq 8 * c$$
$$8 \geq 8 * c$$
$$1 \geq c$$

$c = 1 \mid n = 2$

---

3. Mostre que a recorrência $T(n) = 8T(\frac{n}{2}) + n^2$ é tal que $T(n) = O(n^3)$:

$$T(n) = 8 * T(\frac{n}{2}) + n^2$$
$$O(n^3)$$

Hipotese: $T(n) = c * n^3 - b * n^2$

$$T(n) \leq 8 * T(\frac{n}{2}) + n^2$$
$$T(n) \leq 8 * c * (\frac{n}{2})^3 - b * (\frac{n}{2})^2 + n^2$$
$$T(n) \leq cn^3 - \frac{b * n^2}{4} + n^2$$

Base: $T(n) \leq cn^3 - \frac{b*n^2}{4} + n^2$

$b = 4$

$$T(n) \leq c * n^3 - \frac{4 * n^2}{4} + n^2$$
$$T(n) \leq c * n^3 - n^2 + n^2$$
$$T(n) \leq c * n^3$$

$n = 2$

$$T(2) = 8 * T(\frac{2}{2}) + 2^2$$
$$T(2) = 8 * T(1) + 4$$

$$T(2) = 8 * 1 + 4$$
$$T(2) = 12$$

$$T(2) \leq c * n^3$$
$$12 \leq c * 2^3$$
$$12 \leq 8 * c$$
$$\frac{12}{8} \leq c$$
$$\frac{3}{2} \leq c$$

$b = 4 \mid n = 2 \mid c \geq \frac{3}{2}$

---

4. Considere o seguinte algoritmo recursivo, onde n é um inteiro positivo. Quantos * serão impressos em uma chamada ASTERISCO(n)? Encontre a recorrência em função de n:

a)

$$T(1) = 1$$
$$T(n) = 2 * T(n-1) + n$$

b)

$$T(1) = 1$$
$$T(n) = \frac{n + T(n-2)}{2}$$

---

5. Força Bruta

a) Codigo criado utilizando Rust

```
use std::f64;

pub fn find_closer(points: Vec<(i32, i32)>) -> f64 {
    let mut min_distance: f64 = f64::MAX;

    for (i, aux) in points.iter().enumerate() {
        let (x1, y1) = aux;
            let (x2, y2) = points[n];

            let x2_pow = i32::pow(x2 - x1, 2) as f64;
            let y2_pow = i32::pow(y2 - y1, 2) as f64;

            let aux2 = (x2_pow + y2_pow).sqrt();

            if aux2 < min_distance {
                  min_distance = aux2;
            }
        }
    }
```

```
    min_distance
}
```

Tabela de custo:

Custo	Linha
1	`let mut min_distance: f64 = f64::MAX;`
n	`for (i, aux) in points.iter().enumerate() {`
n + 1	`let (x1, y1) = aux;`
n + (n - 1)	`for n in (i+1)..points.len() {`
n + 1	`let (x2, y2) = points[n];`
n + 1	`let x2_pow = i32::pow(x2 - x1, 2) as f64;`
n + 1	`let y2_pow = i32::pow(y2 - y1, 2) as f64;`
n + 1	`let aux2 = (x2_pow + y2_pow).sqrt();`
n + 1	`if aux2 < min_distance {`
n + 1	`min_distance = aux2;`

Total: $10n + 7$