MAC0338 - Entrega da lista 3

Exercício 1

Qual é o número esperado de comparações executadas na linha 6 do algoritmo?

Seja X uma v.a. que represente o número total de execuções da linha 6. Seja $X_i = \begin{cases} 1 \text{ se a linha 6 é executada} \\ 0 \text{ caso contrário} \end{cases}$.

Logo $X = X_1 + \cdots + X_n$.

Note que a linha 6 é executada apenas quando a condição da linha 4 (se $v[i] >$ maior) é falsa. Portanto, seja a esperança de X_i definida como

$$\begin{aligned} E[X_i] &= \text{prob. de que } A[i] \text{ NÃO seja o máximo em } A[1..i] \\ &= 1 - \text{prob. de que } A[i] \text{ seja o máximo em } A[1..i] \\ &= 1 - \frac{1}{i} \\ &= \frac{i-1}{i}. \end{aligned}$$

Logo, o número esperado de execuções da linha 6, ou seja, a esperança de X é

$$\begin{aligned} E[X] &= E[X_1 + \cdots + X_n] = E[X_1] + \cdots + E[X_n] \\ &= \frac{1-1}{1} + \cdots + \frac{n-1}{n} = 0 + \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \cdots + \frac{n-1}{n} \\ &= \sum_{k=1}^{n} \frac{k-1}{k} \end{aligned}$$

Além disso, note que

$$E[X] = \sum_{k=1}^{n} \frac{k-1}{k} < \sum_{k=1}^{n} 1 = n$$

E portanto, $E[X]$ é $\mathcal{O}(n)$.

Qual é o número esperado de atribuições efetuadas na linha 7 do algoritmo?

Seja Y uma v.a. que represente o número total de execuções da linha 7. Seja $Y_i = \begin{cases} 1 \text{ se a linha 7 é executada} \\ 0 \text{ caso contrário} \end{cases}$.

Logo $Y = Y_1 + \cdots + Y_n$.

Note que a linha 7 é executada apenas quando a condição da linha 6 (se $v[i] <$ menor) é falsa. Portanto, seja a esperança de Y_i definida como

$$E[Y_i] = \text{prob. de que } A[i] \text{ seja o mínimo em } A[1..i] = \frac{1}{i}$$

Logo, o número esperado de execuções da linha 7, ou seja, a esperança de Y é

$$E[Y] = E[Y_1 + \cdots + Y_n] = E[Y_1] + \cdots + E[Y_N] = \frac{1}{1} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{n}$$

Além disso, note que

$$E[Y] = \sum_{k=1}^{n} \frac{1}{k} < 1 + \lg n$$

E portanto, $E[Y]$ é $\mathcal{O}(\lg_n)$.

Exercício 3

Algorithm 1: PARTICIONA-MEDIANA

```
Data: A, p, r
Result: A particionado de p até r
Function PARTICIONA-MEDIANA(A, p, r):
    m ← mediana(A, p, r)                ▷ O(n)
    k ← índice de m no vetor A          ▷ O(n)
    A[k] ↔ A[r]
    return particiona(A, p, r)          ▷ O(n)
```

Algorithm 2: SELECT-MEDIANA

```
Data: A, p, r, k
Result: k-ésimo menor elemento de A
Function SELECT-MEDIANA(A, p, r, k):
    if p = r then return A[p]
    q ← particiona-mediana(A, p, r)                        ▷ 3*O(n)
    if k = q − p + 1 then return A[q]
    if k < q − p + 1 then
        return select-mediana(A, p, q − 1, k)              ▷ T(n/2)
    else
        return select-mediana(A, q + 1, r, k − (q − p + 1)) ▷ T(n/2)
```

Note que os algoritmos são semelhantes aos `PARTICIONA-ALEA` e `SELECT-ALEA`, sendo este $\mathcal{O}(n)$. Nesse algoritmo, o lugar certo do pivô **sempre** será a posição $\lfloor (p+r)/2 \rfloor$, ou seja, dividirá o vetor (ou subvetor, em uma chamada recursiva) em 2 segmentos descritos a seguir.

Como temos esses invariantes no final de cada execução de `PARTICIONA`(A, p, r)

- todos os elementos do segmento à esquerda do pivô são menores que ele
- todos os elementos do segmento à direita do pivô são maiores que ele

então se k for

- igual a $\lfloor (p+r)/2 \rfloor$, temos o que queremos
- menor que $\lfloor (p+r)/2 \rfloor$, com certeza o k-ésimo menor elemento está no segmento à esquerda do pivô
- maior que $\lfloor (p+r)/2 \rfloor$, com certeza o k-ésimo menor elemento está no segmento à direita do pivô.

Para demonstrar o consumo de tempo, note que `SELECT-MEDIANA` é um algoritmo recursivo. Portanto, vamos construir uma fórmula de recorrência $T(n)$ para essa função:

$$
\begin{aligned}
T(n) &= T(n/2) + 3n = \Big(T(n/4) + \frac{3}{2}n\Big) + 3n = \Big(T(n/8) + \frac{3}{4}n\Big) + \Big(3 + \frac{3}{2}\Big)n \\
&\vdots \\
&= T(n/2^q) + \sum_{i=0}^{q-1} \frac{3}{2^i} n = T(n/2^q) + 3n \sum_{i=1}^{q} \frac{1}{2^{i-1}} = T(n/2^q) + 3n \underbrace{\frac{1-(1/2)^q}{1-1/2}}_{\text{soma de PG finita}} \\
&= T(n/2^q) + 3n\Big(2 - \frac{1}{2^{q-1}}\Big) = T(n/2^q) + 3n\Big(2 - \frac{2}{2^q}\Big)
\end{aligned}
$$

Para $2^q = n$ e n potência de 2, temos a expressão

$$T(n) = T(1) + 3n\Big(2 - \frac{2}{n}\Big) = 1 + 6n - 6 = 6n - 5$$

Vamos provar por indução em q, $q = \lg n$, que $T(2^q) = 6 \cdot 2^q - 5$.

Para $q = 0$,

$$T(2^0) = 6 \cdot 2^0 - 5 \implies T(1) = 1$$

Para $q \geq 1$, suponha $T(2^{q-1}) = 6 \cdot 2^{q-1} - 5$, então

$$T(2^q) = T(2^{q-1}) + 3 \cdot 2^q = (6 \cdot 2^{q-1} - 5) + 3 \cdot 2^q = 3 \cdot 2^q - 5 + 3 \cdot 2^q = 6 \cdot 2^q - 5,$$

como queríamos provar.

Logo, para $T(n) = 6n - 5$, o consumo de tempo é proporcional a $\mathcal{O}(n)$.

Exercício 9

Algorithm 3: MEDIANA2VETORES

```
Data: X, p, q, Y, r, s
Result: mediana do vetor combinado X com Y
Function MEDIANA(X, p, q, Y, r, s):
    if p = r and r = s then
        if X[p] < Y[r] then
            return X[p]
        else
            return Y[r]
    x ← ⌊(p+q)/2⌋
    y ← ⌊(r+s)/2⌋
    if X[x] < Y[y] then
        return MEDIANA2VETORES(X, x + 1, q, Y, r, y)            ▷ T(n/2)
    else
        return MEDIANA2VETORES(X, p, x, Y, y, s)                ▷ T(n/2)
```

O algoritmo acima recebe 2 vetores X, Y, ambos de tamanho n, ordenados e **sem elementos repetidos**. Os índices p, q são referentes ao subvetor de X que será processado, enquanto r, s são referentes ao subvetor de Y. É retornado a mediana de um vetor que combinaria todos os elementos de X e Y.

O algoritmo funciona recebendo inicialmente `MEDIANA2VETORES`$(X, 1, n, Y, 1, n)$ e

1. encontra o índice da mediana de cada vetor, sendo x o índice da mediana de $X[1..n]$ e y, o de $Y[1..n]$.
2. Se $p = q$ e $r = s$, retorna o mínimo entre $X[p]$ e $Y[r]$.
3. Se $p \neq q$ e $r \neq s$, então quantifica $X[x]$ e $Y[y]$.
   (a) Se $X[x] < Y[y]$, então faz a recursão considerando $X[x+1..q]$ e $Y[r..y]$
   (b) Senão (*se* $X[x] < Y[y]$), então faz a recursão considerando $X[p..x]$ e $Y[y..s]$

A análise de consumo de tempo considera a fórmula recursiva $T(n) = T(n/2) + \mathcal{O}(1)$, sendo $T(n/2)$ a chamada recursiva e $\mathcal{O}(1) = 1$ as chamadas das outras linhas.

$$T(n) = T(n/2) + \mathcal{O}(1) = (T(n/4) + \mathcal{O}(1)) + \mathcal{O}(1) = \cdots = T(n/2^k) + k\mathcal{O}(1) = T(n/2^k) + k$$

Considerando $k = \lg n$ e n potência de 2, temos a expressão $T(n) = T(1) + \lg n = 1 + \lg n$.

Vamos provar por indução em k, que $T(2^k) = 1 + k$.

Para $k = 0$, diretamente

$$T(2^0) = 1 + 0 \implies T(1) = 1$$

.

Para $k \geq 1$ e supondo $T(2^{k-1}) = 1 + (k-1)$, temos

$$T(2^k) = T(2^{k-1}) + \mathcal{O}(1) = 1 + (k-1) + 1 \implies T(2^k) = k + 1,$$

como queríamos provar.

Logo, o consumo de tempo da função `MEDIANA2VETORES` é $1 + \lg n$, proporcional a $\mathcal{O}(\lg n)$.