UFPE/CIn – ENGENHARIA DA COMPUTAÇÃO
IF672 – AED 2018.2 – EXERCÍCIO ESCOLAR 1
PROFESSOR: GUSTAVO CARVALHO

NOME: _____

1. {2,0 pt.} Considerando o código de ordenação abaixo (`selectionSort`), a comparação $A[j] < A[min]$ como operação básica, e $C(n)$ a quantidade de operações básicas realizadas em função do tamanho n do array A, responda:

   (a) {0,5 pt.} Por que $C_{worst}(n) = C_{average}(n) = C_{best}(n) = C(n)$?
   Resposta: É preciso calcular o pior caso, o caso médio e o melhor caso quando $C(n)$ depende não somente do tamanho n, mas também de alguma outra propriedade da entrada; por exemplo, se esta já está ordenada. No entanto, no caso do *selection sort*, a eficiência temporal deste algoritmo de ordenação depende unicamente de n.

   (b) {1,5 pt.} Defina matematicamente $C(n)$ e prove que $C(n) \in \Theta(n^2)$.
   Resposta:

$$\begin{aligned}
C(n) &= \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 \\
&= \sum_{i=0}^{n-2} ((n-1) - (i+1) + 1) \\
&= \sum_{i=0}^{n-2} (n-1-i) \\
&= \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} i \\
&= (n-1) \sum_{i=0}^{n-2} 1 - \tfrac{(0+(n-2))((n-2)-0+1)}{2} \\
&= (n-1)((n-2)-0+1) - \tfrac{(n-2)(n-1)}{2} \\
&= (n-1)^2 - \tfrac{(n-2)(n-1)}{2} \\
&= \tfrac{2(n^2-2n+1)-(n^2-n-2n+2)}{2} \\
&= \tfrac{2n^2-4n+2-n^2+3n-2}{2} \\
&= \tfrac{n^2-n}{2}
\end{aligned}$$

   Para provar $C(n) \in \Theta(n^2)$, é necessário encontrar c_1 e c_2 positivos e $n_0 \geq 0$ tal que $c_2 n^2 \leq C(n) \leq c_1 n^2$ para todo $n \geq n_0$. Como $\frac{n^2-n}{2} \leq \frac{1}{2}n^2$ para todo $n \geq 0$, e $\frac{1}{4}n^2 \leq \frac{n^2-n}{2}$ para todo $n \geq 2$, temos que $c_1 = \frac{1}{2}, c_2 = \frac{1}{4}$ e $n_0 = 2$.

Algoritmo: `void selectionSort(int A[], int n)`

```
1  for i ← 0 to n − 2 do
2      min ← i;
3      for j ← i + 1 to n − 1 do
4          if A[j] < A[min] then min ← j;
5      swap A[i] and A[min];
```

2. {1,0 pt.} Escreva um código para o algoritmo de busca em uma árvore binária de busca (BST): `bool find(BSTNode rt, int k)`. Se k estiver na BST, a função deve retornar *true*; caso contrário, o valor *false* deve ser retornado. *BSTNode* é uma estrutura composta por um inteiro (*key*) e referências para a sub-árvore esquerda e direita (*left* e *right*, respectivamente).
Resposta:

Algoritmo: `bool find(BSTNode rt, int k)`

```
1  if rt = NULL then return false ;
2  if k < rt.key then return find(rt.left, k) ;
3  else if rt = k then return true ;
4  else return find(rt.right, k) ;
```

3. {2,0 pt.} Escreva um código para o algoritmo de ordenação *quicksort*: `void quicksort(int A[], int l, int r)`. Considere o algoritmo de particionamento proposto por Hoare, assim como a escolha do elemento mais à esquerda ($A[l]$) como o pivô do particionamento.
Resposta:

Algoritmo: `void quicksort(int A[], int l, int r)`

```
1  if l < r then
2      s ← partition(A, l, r);
3      quicksort(A, l, s − 1);
4      quicksort(A, s + 1, r)
```

Algoritmo: `int partition(int A[], int l, int r)`

```
1  p ← A[l];
2  i ← l;
3  j ← r + 1;
4  repeat
5  |   repeat
6  |   |   i ← i + 1;
7  |   until A[i] ≥ p ∨ i ≥ r;
8  |   repeat
9  |   |   j ← j − 1;
10 |   until A[j] ≤ p;
11 |   swap A[i] and A[j];
12 until i ≥ j;
13 swap A[i] and A[j] swap A[l] and A[j];
14 return j;
```

4. {2,0 pt.} Seja uma tabela hash com 8 posições, $h(k) = k - (8 * \lfloor k/8 \rfloor)$ a função hash (o símbolo / denota a divisão entre números reais), uma política de resolução de colisões baseada em *quadratic probing* conforme $p(k,i) = \frac{i^2+i}{2}$, mostre o passo-a-passo da inserção dos valores (nesta ordem): 2, 4, 8, 16, 32, e -12. Desenhe uma nova tabela após a inserção de cada valor. Exiba seus cálculos de $h(k)$ e $p(k,i)$.

Resposta:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
		2					
		2		4			
8		2		4			
8	16	2		4			
8	16	2	32	4			
8	16	2	32	4	-12		

Cálculos:

- $h(2) = 2 - (8 * \lfloor 2/8 \rfloor) = 2$
- $h(4) = 4 - (8 * \lfloor 4/8 \rfloor) = 4$
- $h(8) = 8 - (8 * \lfloor 8/8 \rfloor) = 0$

- $h(16) = 16 - (8 * \lfloor 16/8 \rfloor) = 0$
  - Resolvendo colisão por *quadratic probing*:
    tentativa 1: $p(k, 1) = \frac{1^2+1}{2} = 1$
    nova posição: $(h(16) + p(k, 1))$ mod $8 = 1$
- $h(32) = 32 - (8 * \lfloor 32/8 \rfloor) = 0$
  - Resolvendo colisão por *quadratic probing*:
    tentativa 1: $p(k, 1) = \frac{1^2+1}{2} = 1$
    nova posição: $(h(32) + p(k, 1))$ mod $8 = 1$, colisão;
    tentativa 2: $p(k, 2) = \frac{2^2+2}{2} = 3$
    nova posição: $(h(32) + p(k, 2))$ mod $8 = 3$
- $h(-12) = -12 - (8 * \lfloor -12/8 \rfloor) = 4$
  - Resolvendo colisão por *quadratic probing*:
    tentativa 1: $p(k, 1) = \frac{1^2+1}{2} = 1$
    nova posição: $(h(-12) + p(k, 1))$ mod $8 = 5$

Comentários:

- A primeira linha da tabela representa os índices do array;
- A segunda linha da tabela representa a tabela sem nenhum valor;
- Uma nova linha é criada para cada inserção.

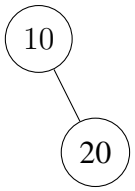
5. {2,0 pt.} Considerando uma árvore AVL inicialmente vazia, responda:

   (a) {1,5 pt.} Mostre o passo-a-passo da inserção dos valores (nesta ordem): 10, 20, 30, 5, 0 e 8. Desenhe uma nova árvore após cada inserção. Escreva *rotação X em Y*, onde *Y* representa a raiz da sub-árvore rotacionada e $X \in \{L, R, LR, RL\}$, caso uma rotação tenha ocorrido durante a inserção.
   Resposta:

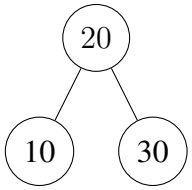
   Inserindo 10:

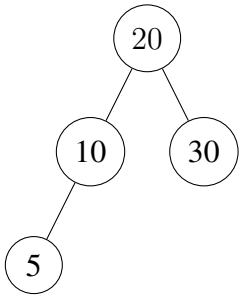
Inserindo 20:



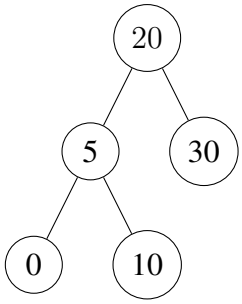
Inserindo 30 (rotação L em 10):



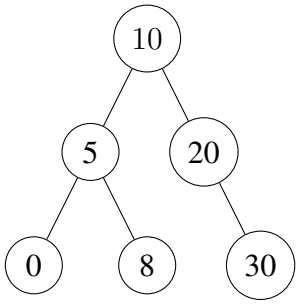
Inserindo 5:



Inserindo 0 (rotação R em 10):



Inserindo 8 (rotação LR em 20):



(b) {0,5 pt.} A sequência de valores obtida em uma travessia em pós-ordem.
Resposta: 0, 8, 5, 30, 20, 10

6. {1,0 pt.} Considerando uma inserção *bottom-up*, mostre o passo-a-passo da construção da heap máxima formada pelos números (nesta ordem): 10, 23, 8, 20, 15 e 18. Desenhe uma nova heap (como um array) após o processo de *heapify* de cada nó interno.
Resposta:

1	**2**	**3**	**4**	**5**	**6**
10	23	8	20	15	18
10	23	18	20	15	**8**
10	**23**	18	20	15	8
23	20	18	**10**	15	8

Comentários:

- A primeira linha da tabela representa os índices do array;
- A segunda linha da tabela representa a heap no estado inicial;
- Uma nova linha é criada após o *heapify* de cada nó interno (em negrito).