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

NOME: _____

1. {1,5 pt.} Mostre o passo-a-passo da ordenação dos números (nesta ordem) $5, 16, 10, 12, 4, 17, 13, 8$, considerando o algoritmo `mergesort`. Redesenhe o array imediatamente após realizar cada operação de *merge*.
   Resposta:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Inicialmente	5	16	10	12	4	17	13	8
l = 0 e r = 1	5	16	10	12	4	17	13	8
l = 2 e r = 3	5	16	10	12	4	17	13	8
l = 0 e r = 3	5	10	12	16	4	17	13	8
l = 4 e r = 5	5	10	12	16	4	17	13	8
l = 6 e r = 7	5	10	12	16	4	17	8	13
l = 4 e r = 7	5	10	12	16	4	8	13	17
l = 0 e r = 7	4	5	8	10	12	13	16	17

2. {2,0 pt.} Mostre o passo-a-passo da inserção dos números (nesta ordem) $62, 85, 50, 58, 59, 61, 90, 87, 99$ em uma AVL inicialmente vazia. Desenhe uma nova árvore após cada inserção. Escreva *rotação X em Y*, onde *Y* é a raiz da sub-árvore rotacionada e $X \in \{L, R, LR, RL\}$, caso ocorra uma rotação na inserção.
   Resposta:

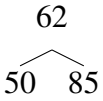
   Inserindo 62:

   62

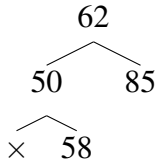
   Inserindo 85:

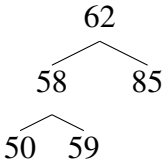
   Inserindo 50:



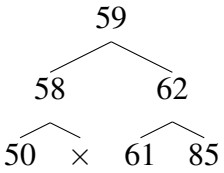
Inserindo 58:



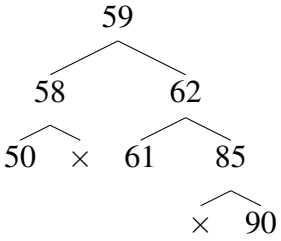
Inserindo 59 (rotação L em 50):



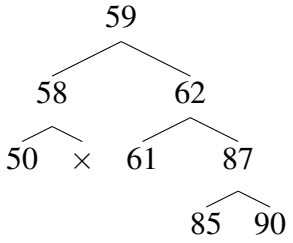
Inserindo 61 (rotação LR em 62):



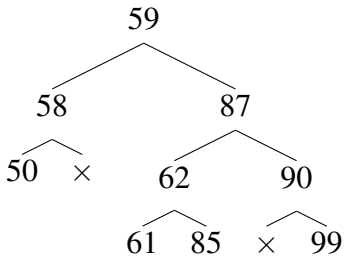
Inserindo 90:



Inserindo 87 (rotação RL em 85):

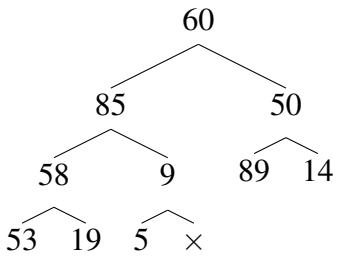


Inserindo 99 (rotação L em 62):

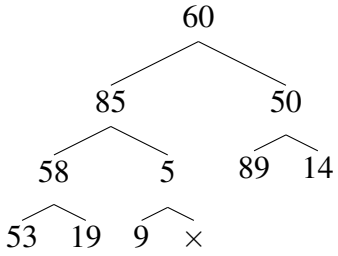


3. {1,5 pt.} Considerando uma inserção *bottom-up*, mostre o passo-a-passo da construção da heap mínima formada pelos números (nesta ordem): 60, 85, 50, 58, 9, 89, 14, 53, 19, 5. Desenhe uma nova heap (como uma árvore) após o processo de *heapify* de cada nó interno.
Resposta:

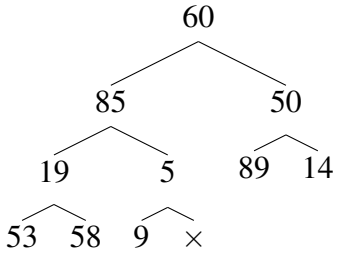
Heap (estado inicial):



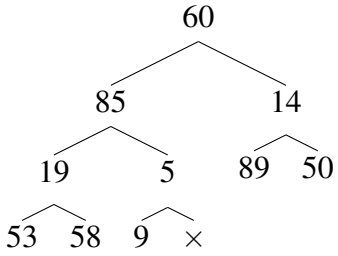
Heap (heapify em 9):



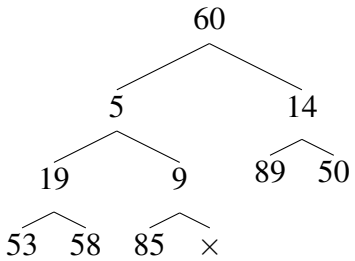
Heap (heapify em 58):



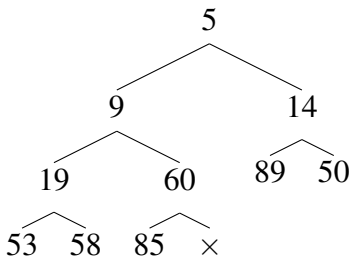
Heap (heapify em 50):



Heap (heapify em 85):



Heap (heapify em 60):

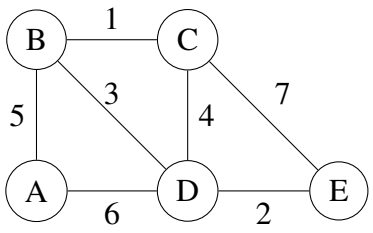


4. {2,0 pt.} Seja G um grafo representado como uma matriz de adjacências (0 indica ausência de aresta e $p > 0$ indica a presença de aresta de peso p) com n nós, escreva um pseudocódigo para: `void dijkstra(int[][] G, int n, int s, int[] D)`, usando uma heap mínima. O código deve atualizar o array de distâncias D com o menor caminho a partir de s. Não é preciso detalhar o corpo das funções de manipulação da heap.
Resposta:

Algoritmo: void dijkstra(int[][] G, int n, int s, int[] D)

```
1  H[0] ← (s, 0);
2  for i ← 0 to n − 1 do
3  |    D[i] ← ∞;
4  |    V[i] ← UNVISITED;
5  D[s] ← 0;
6  i ← 0;
7  while i < n ∧ ¬ empty(H) do
8  |    repeat
9  |    |    v ← vertex(removemin(H));
10 |    until V[v] = UNVISITED;
11 |    V[v] ← VISITED;
12 |    for j ← 0 to n − 1 do
13 |    |    if G[v][j] ≠ 0 ∧ V[j] ≠ VISITED ∧ D[j] > D[v] + G[v][j] then
14 |    |    |    D[j] ← D[v] + G[v][j];
15 |    |    |    insert(H, (j, D[j]));
16 |    i++;
```

5. {1,5 pt.} Considerando o algoritmo de Kruskal (usando uma *union-find*: *quick-union* sem compressão), calcule a árvore geradora de peso mínimo para o grafo ao lado. Mostre também a evolução da *union-find* como um array (inicialmente, um conjunto para cada nó). Ao fazer uma união, a raiz deve ser o nó representativo lexicograficamente menor



Resposta:

Inicialmente:

A	**B**	**C**	**D**	**E**
–	–	–	–	–

Após processar (B,C)

A	**B**	**C**	**D**	**E**
–	–	B	–	–

Após processar (D,E)

A	**B**	**C**	**D**	**E**
–	–	B	–	D

Após processar (B,D)

A	**B**	**C**	**D**	**E**
–	–	B	B	D

Após processar (C,D)

A	**B**	**C**	**D**	**E**
–	–	B	B	D

Após processar (A,B)

A	**B**	**C**	**D**	**E**
–	A	B	B	D

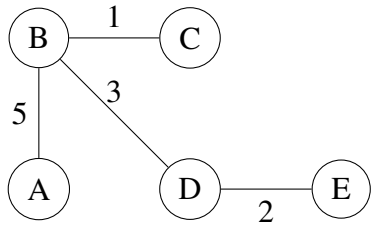
Após processar (A,D)

A	**B**	**C**	**D**	**E**
–	A	B	B	D

Após processar (C,E)

A	**B**	**C**	**D**	**E**
–	A	B	B	D

Portanto, a árvore geradora de peso mínimo é a seguinte:



6. {1,5 pt.} Seja C a matriz ao lado ($0/1$ indica a ausência/presença de uma moeda), considerando um robô saindo da posição (1,1) até (5,6), andando para à direita *ou* para baixo, e coletando as moedas no caminho, encontre a maior quantidade de moedas que podem ser coletadas. Use programação dinâmica (*bottom-up*) e apresente a matriz ($M[5][6]$) construída na busca.
Resposta:

	1	**2**	**3**	**4**	**5**	**6**
1	0	0	0	0	1	0
2	0	1	0	1	0	0
3	0	0	0	1	0	1
4	0	0	1	0	0	1
5	1	0	0	0	1	0

	1	**2**	**3**	**4**	**5**	**6**
1	0	0	0	0	1	1
2	0	1	1	2	2	2
3	0	1	1	3	3	4
4	0	1	2	3	3	5
5	1	1	2	3	4	5

Portanto, a maior quantidade de moedas que podem ser coletadas é 5.