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

NOME: _____

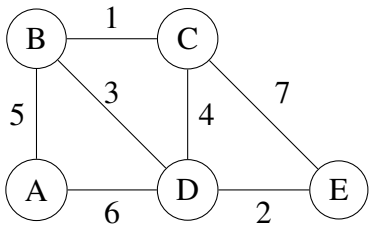
1. {1,5 pt.} Mostre o passo-a-passo da ordenação dos números (nesta ordem) $5, 16, 10, 12, 4, 17, 13, 8$, considerando o algoritmo `mergesort`. Redesenhe o array imediatamente após realizar cada operação de *merge*.

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.

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.

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.

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



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.

	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