UFPE/CIn – ENGENHARIA DA COMPUTAÇÃO
IF672 – AED 2018.2 – 2ª CHAMADA
PROFESSOR: GUSTAVO CARVALHO
NOME: _____

1. {1,5 pt.} Seja *A* um array ordenado crescentemente com *n* inteiros, escreva um código *não-recursivo* de busca binária em *A*: `bool binSearch(int[] A, int n, int k)`; retorne *true* se *k* estiver em *A*; *false*, caso contrário.
Resposta: Abaixo, o código de `binSearch`.

Algoritmo: bool binSearch(int[] A, int n, int k)

```
1  l ← 0;
2  r ← n − 1;
3  while l ≤ r do
4  |   m ← ⌊(l + r)/2⌋;
5  |   if k = A[m] then return true;
6  |   else if k < A[m] then r ← m − 1;
7  |   else l ← m + 1;
8  return false;
```

2. {1,5 pt.} Seja G um grafo com n nós (uma matriz de adjacências, onde $0/1$: ausência/presença de aresta), escreva um código para: `void bfs(int[][] G, int n)`. Este código deve percorrer todos os nós do grafo em largura.
Resposta: Abaixo, o código de `bfs` e da função auxiliar `bfs_aux`.

Algoritmo: void bfs(int[][] G, int n)

```
1  for i ← 0 to n − 1 do v[i] ← false ;
2  for i ← 0 to n − 1 do
3      if ¬ v[i] then bfs_aux(G, n, i, v);
```

Algoritmo: void bfs_aux(int[][] G, int n, int i, bool[] v)

```
1  q ← create_queue();
2  enqueue(q, i);
3  v[i] ← true;
4  while length(q) > 0 do
5      i ← dequeue(q);
6      for j ← 0 to n − 1 do
7          if G[i][j] = 1 ∧ ¬ v[j] then
8              v[j] ← true;
9              enqueue(q, j);
```

3. {2,0 pt.} Seja uma tabela hash com 6 posições, $h(k) = k - (6 * \lfloor k/6 \rfloor)$ a função hash ($/$ denota a divisão entre números reais), 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): 8, 4, 2, -16, 16, e 5. Desenhe uma nova tabela após cada inserção. Exiba seus cálculos de $h(k)$ e $p(k,i)$.
Resposta:

0	**1**	**2**	**3**	**4**	**5**
		8			
		8		4	
		8	2	4	
		8	2	4	-16
	16	8	2	4	-16
5	16	8	2	4	-16

Cálculos:

- $h(8) = 8 - (6 * \lfloor 8/6 \rfloor) = 2$
- $h(4) = 4 - (6 * \lfloor 4/6 \rfloor) = 4$
- $h(2) = 2 - (6 * \lfloor 2/6 \rfloor) = 2$
  - Resolvendo colisão por *quadratic probing*:
    tentativa 1: $p(k,1) = \frac{1^2+1}{2} = 1$
    nova posição: $(h(2) + p(k,1)) \bmod 6 = 3$
- $h(-16) = -16 - (6 * \lfloor -16/6 \rfloor) = 2$
  - Resolvendo colisão por *quadratic probing*:
    tentativa 1: $p(k,1) = \frac{1^2+1}{2} = 1$
    nova posição: $(h(-16) + p(k,1)) \bmod 6 = 3$
    tentativa 2: $p(k,2) = \frac{2^2+2}{2} = 3$
    nova posição: $(h(-16) + p(k,2)) \bmod 6 = 5$
- $h(16) = 16 - (6 * \lfloor 16/6 \rfloor) = 4$
  - Resolvendo colisão por *quadratic probing*:
    tentativa 1: $p(k,1) = \frac{1^2+1}{2} = 1$
    nova posição: $(h(16) + p(k,1)) \bmod 6 = 5$
    tentativa 2: $p(k,2) = \frac{2^2+2}{2} = 3$
    nova posição: $(h(16) + p(k,2)) \bmod 6 = 1$

- $h(5) = 5 - (6 * \lfloor 5/6 \rfloor) = 5$
  - Resolvendo colisão por *quadratic probing*:
    tentativa 1: $p(k,1) = \frac{1^2+1}{2} = 1$
    nova posição: $(h(5) + p(k,1))$ mod $6 = 0$

Comentários:

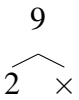
- A primeira linha da tabela representa os índices do array;
- A segunda linha da tabela representa a tabela sem nenhum valor;
- Uma nova linha é criada para cada inserção.

4. {1,5 pt.} Considerando uma árvore AVL inicialmente vazia, mostre o passo-a-passo da inserção dos valores (nesta ordem): 9, 2, 15, 18, 20, 19, 17, e 16. 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:

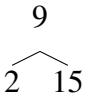
Inserindo 9:

9

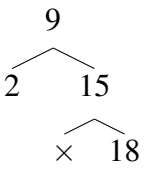
Inserindo 2:



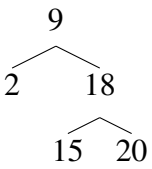
Inserindo 15:



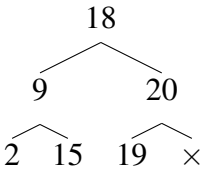
Inserindo 18:



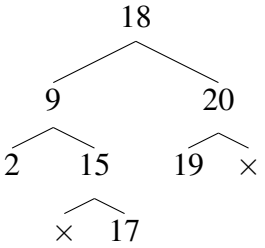
Inserindo 20 (rotação L em 15):



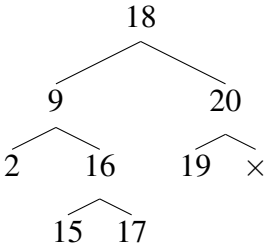
Inserindo 19 (rotação L em 9):



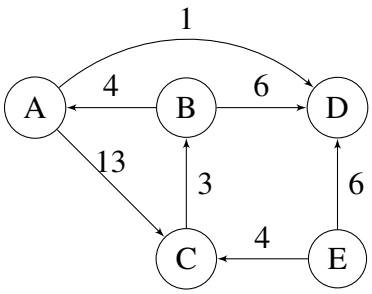
Inserindo 17:



Inserindo 16 (rotação RL em 15):



5. {2,0 pt.} Considerando o algoritmo de Dijkstra (usando uma *heap*), e o grafo ao lado, calcule os menores caminhos a partir do nó B. Mostre a evolução do array de distâncias (inicialmente, com ∞ para todos os nós exceto B) e da heap como um array (inicialmente, só com o par $(B, 0)$) após a visita de cada nó do grafo.



Resposta: A seguir, resposta da questão – evolução abaixo:

Inicialmente:

	A	**B**	**C**	**D**	**E**
Dist.	∞	0	∞	∞	∞

	0	**1**	**2**	**3**
Heap	–	(B,0)	–	–

Após visitar B:

	A	**B**	**C**	**D**	**E**
Dist.	4	0	∞	6	∞

	0	**1**	**2**	**3**
Heap	–	(A,4)	(D,6)	–

Após visitar A:

	A	**B**	**C**	**D**	**E**
Dist.	4	0	17	5	∞

	0	**1**	**2**	**3**
Heap	–	(D,5)	(C,17)	(D,6)

Após visitar D:

	A	**B**	**C**	**D**	**E**
Dist.	4	0	17	5	∞

	0	**1**	**2**	**3**
Heap	–	(D,6)	(C,17)	–

Após visitar C:

	A	**B**	**C**	**D**	**E**
Dist.	4	0	17	5	∞

	0	**1**	**2**	**3**
Heap	–	–	–	–

6. {1,5 pt.} Considerando uma mochila com capacidade de 5 kg, e os itens (peso, valor): $i_1 = (1, 6), i_2 = (2, 10), i_3 = (3, 12)$, encontre o subconjunto de itens mais valioso que cabe na mochila. Só existe uma unidade de cada item e não é possível dividir um item. Use programação dinâmica (*bottom-up*) e apresente a matriz (*item* $\times$ *capacidade*) construída na busca.
Resposta: O subconjunto mais valioso que cabe na mochila é: $v_2 + v_3 = 22$.

–	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	6	6	6	6	6
2	0	6	10	16	16	16
3	0	6	10	16	18	22