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

1. {2,0 pt.} Seja G um grafo dirigido acíclico com n nós, representado como uma matriz de adjacências (0 indica a ausência de aresta e 1 a presença de aresta), escreva um código para: `void toposort(int[][] G, int n)`. Este código deve imprimir no final uma ordenação topológica do grafo G.
Resposta: Abaixo, o código de `toposort` e da função auxiliar `topo_aux`

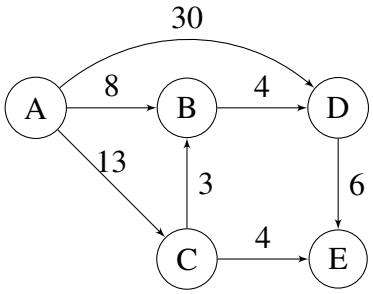
Algoritmo: void toposort(int[][] G, int n)

```
1 for i ← 0 to n − 1 do v[i] ← false ;
2 s ← create_stack();
3 for i ← 0 to n − 1 do
4     if ¬ v[i] then topo_aux(G, n, i, s, v);
5 while ¬ empty(s) do print(pop(s)) ;
```

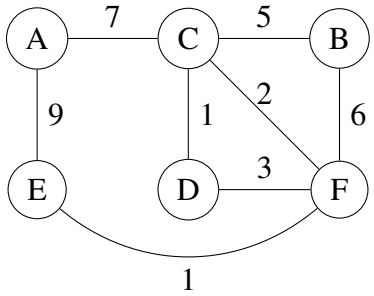
Algoritmo: void topo_aux(int[][] G, int n, int i, Stack s, bool[] v)

```
1 v[i] ← true;
2 for j ← 0 to n − 1 do
3     if G[i][j] = 1 ∧ ¬ v[j] then topo_aux(G, n, j, s, v) ;
4 push(s, i)
```

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



3. {2,0 pt.} Considerando o algoritmo de Kruskal (usando uma *union-find*: *quick-union* sem compressão), calcule a árvore geradora de peso mínimo para o grafo abaixo. 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: A seguir, resposta da questão 2 – evolução abaixo:

Inicialmente:

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

	0	**1**	**2**	**3**
Heap	–	(A,0)	–	–

Após visitar A:

	A	**B**	**C**	**D**	**E**
Dist.	0	8	13	30	∞

	0	**1**	**2**	**3**
Heap	–	(B,8)	(C,13)	(D,30)

Após visitar B:

	A	**B**	**C**	**D**	**E**
Dist.	0	8	13	12	∞

	0	**1**	**2**	**3**
Heap	–	(D,12)	(D,30)	(C,13)

Após visitar D:

	A	**B**	**C**	**D**	**E**
Dist.	0	8	13	12	18

	0	**1**	**2**	**3**
Heap	–	(C,13)	(D,30)	(E,18)

Após visitar C:

	A	**B**	**C**	**D**	**E**
Dist.	0	8	13	12	17

	0	**1**	**2**	**3**
Heap	–	(E,17)	(D,30)	(E,18)

Após visitar E:

	A	**B**	**C**	**D**	**E**
Dist.	0	8	13	12	17

	0	**1**	**2**	**3**
Heap	–	(E,18)	(D,30)	–

Resposta: A seguir, resposta da questão 3 – evolução abaixo:

Inicialmente:

A	**B**	**C**	**D**	**E**	**F**
–	–	–	–	–	–

Após processar (C,D)

A	B	C	D	E	F
–	–	–	C	–	–

Após processar (E,F)

A	B	C	D	E	F
–	–	–	C	–	E

Após processar (C,F)

A	B	C	D	E	F
–	–	–	C	C	E

Após processar (D,F)

A	B	C	D	E	F
–	–	–	C	C	E

Após processar (B,C)

A	B	C	D	E	F
–	–	B	C	C	E

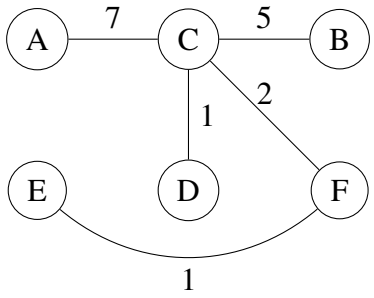
Após processar (B,F)

A	B	C	D	E	F
–	–	B	C	C	E

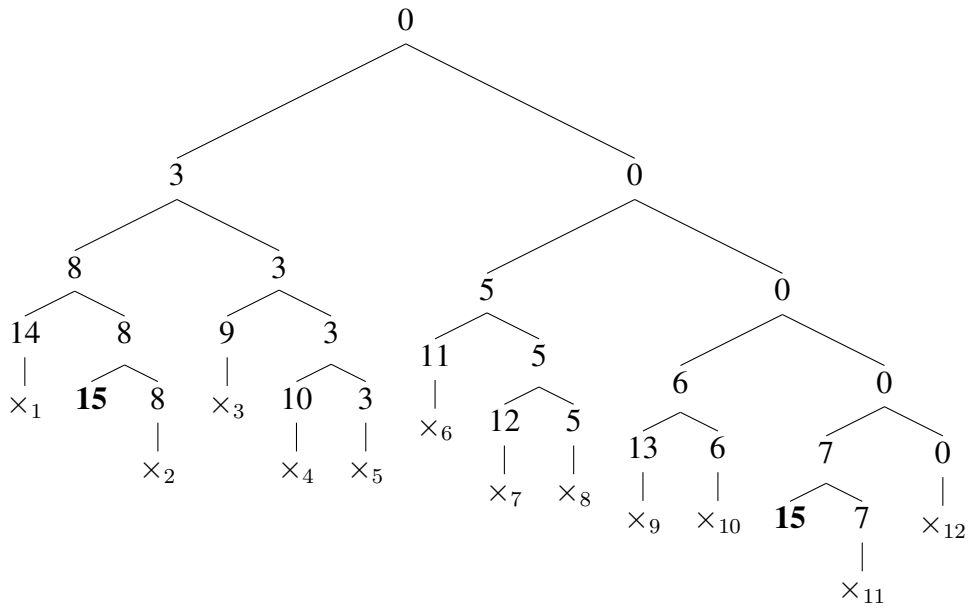
Após processar (A,C)

A	B	C	D	E	F
–	A	B	C	C	E

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



4. {2,0 pt.} Seja $A = \{3, 5, 6, 7, 8\}$, encontre *todos* os $A' \subseteq A$, tal que a soma de seus elementos seja igual a 15. Resolva o problema usando *backtracking* (filho à esquerda/direita = inclusão/exclusão do i-ésimo item, respectivamente). Desenhe a árvore de espaço de estados. Os nós devem ser rotulados com o somatório atual. Ao não continuar a busca em certo nó, explique o porquê.
Resposta: Ver abaixo a árvore de espaço de estados:



Soluções:

- $3 + 5 + 7 = 15$
- $7 + 8 = 15$

A busca não foi continuada em alguns estados pelos seguintes motivos:

- $\times_1 = 14 + 7 > 15$
- $\times_2 = 8 + 8 > 15$
- $\times_3 = 9 + 7 > 15$
- $\times_4 = 10 + 8 > 15$
- $\times_5 = 3 + 8 < 15$
- $\times_6 = 11 + 7 > 15$
- $\times_7 = 12 + 8 > 15$
- $\times_8 = 5 + 8 < 15$
- $\times_9 = 13 + 8 > 15$
- $\times_{10} = 6 + 8 < 15$
- $\times_{11} = 7 + 0 < 15$
- $\times_{12} = 0 + 8 < 15$

5. {2,0 pt.} Considerando uma mochila com capacidade de 4 kg, e os itens (peso, valor): $i_1 = (2, 20), i_2 = (1, 10), i_3 = (3, 20), i_4 = (2, 15)$, encontre o subconjunto de itens mais valioso que cabe na mochila. 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: $v_1 + v_4 = 35$.

–	0	1	2	3	4
0	0	0	0	0	0
1	0	0	20	20	20
2	0	10	20	30	30
3	0	10	20	30	30
4	0	10	20	30	35