CHECKLIST fiz 🟩 incompleto 🟧 não entendi 🟥

- [x] 1. descrever uma classe para representar grafos
- [x] 2. diferença do exc. 1 para digrafos
- [x] 3. escrever função achaFonte
- [ ] 4. escrever função testaCaminho
- [ ] 5. escrever função Kcaminho
- [ ] 6. escrever função - monumentos de um cavalo
- [ ] 7. escrever função ancestralComum
- [ ] 8. escrever função, falar complexidade
- [ ] 9. percorrer digrafo em profundidade
- [ ] 10. justificar
- [ ] 11. algoritmo de Dijkstra
- [ ] 12. escrever função (pais)
- [ ] 13. escrever função (clique)
- [ ] 14. usar função do 13, falar complexidade

1. Descreva uma classe para representar `grafos`, implementando:

   - construtor: devolve um grafos com V vértices, sem arestas; ✓
   - inclui aresta: inclui uma aresta $u - v$ no grafo; ✓
   - remove aresta: remove a aresta $u - v$ (se existir) do grafo; ✓

```
class Grafo {

    private:
        int V;                      // número de vértices
        int E;                      // número de arestas
        vector<int> *adj;           // vetor de adjacência

    public:
        Grafo(int V) {
            this->V = V;
            E = 0;
            adj = new vector<int>[V];
        }

        void add(int u, int v) {
            adj[u].push(v)
            adj[v].push(u)
        }

        void remove(int u, int v) {
            adj[u].pop(v);
            adj[v].pop(u);
        }

};
```

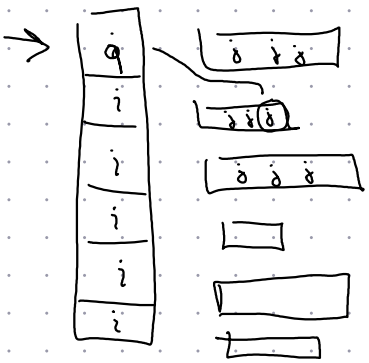
2. O que muda nas funções acima para digrafos?

Para adicionar uma aresta u-v em um digrafo usando a função add() apenas inserimos a aresta dirigida de u até v (adj[u].push(v)) ao invés de adicionar nas adjacências de u e v. Assim como na função remove, removemos apenas de adj[u], pois as arestas são dirigidas.

3. Escreva uma função `achaFonte` que devolve, se existir, um vértice fonte no grafo e -1, caso contrário.

Uma fonte (= source) é um vértice que tem grau de entrada nulo.

```
int achaFonte() {
    for (int q = 0; q < V; q++) {
        bool fonte = false
        for (int i = 0; i < V && q != i; i++) {
            if (std::count(adj[i].begin(), adj[i].end(), q) {
                fonte = true;
            }
        }
        if (fonte == false) return q;
    }
    return -1;
}
```



4. Escreva uma função `testaCaminho` que recebe um vetor `seq` e verifica se seq é um caminho no grafo.

prof fez

5. Escreva uma função `kcaminho` que recebe dois vértices u e v e um inteiro k e verifica se existe no grafo um caminho de comprimento menor ou igual a k.

6. Escreva uma função que recebe um inteiro n e constroi o grafo correspondente aos movimentos de um cavalo no tabuleiro de xadrez $n \times n$, ou seja, os vértices são as posIcões do tabuleiro e dois vértices são ligados por uma aresta se é possível que um cavalo vá de uma posição para a outra. Observe que este grafo é bipartido :)

   Faça um programa que determine, neste grafo, a distância do vértice correspondente ao canto superior esquerdo do tabuleiro a todas as outras posições do tabuleiro.

   Exemplo: Para o tabuleiro 4×4, as distâncias entre a posição do canto e as posições do tabuleiro são:

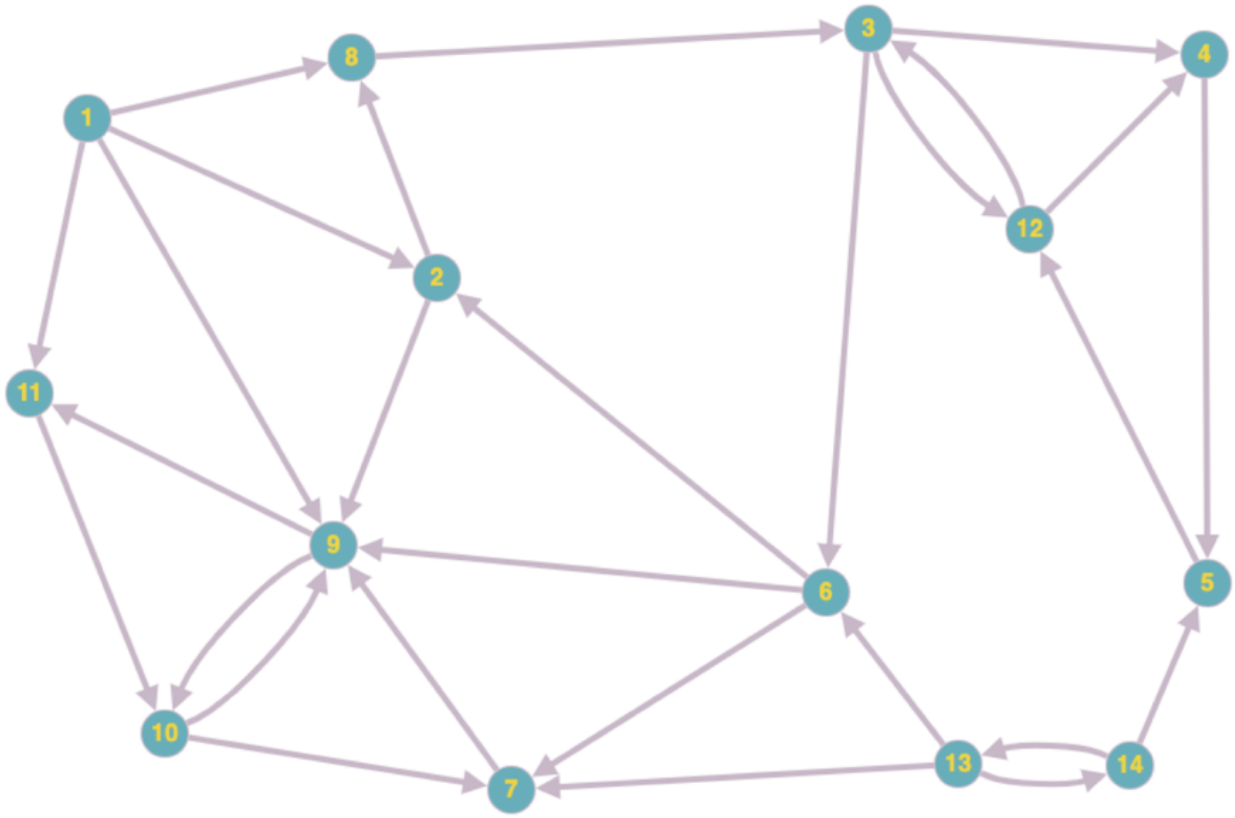
$$\begin{matrix} 0 & 3 & 2 & 5 \\ 3 & 4 & 1 & 2 \\ 2 & 1 & 4 & 3 \\ 5 & 2 & 3 & 2 \end{matrix}$$

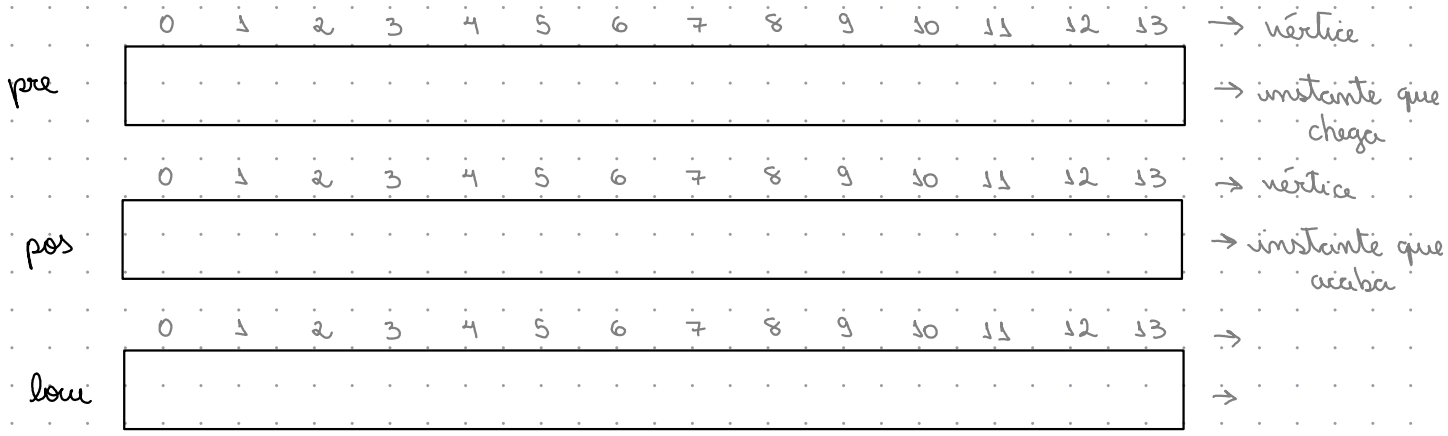
7. Você pode representar uma genealogia por um grafo dirigido acíclico, em que os vértices representam as pessoas e existe um arco $u - v$ se u é pai ou mãe de v. Faça uma função `ancestralComum` que recebe três vértices u, v e w e devolve 1 se w é ancestral de u e v. Qual a complexidade da função?

8. Usando a função do item anterior, faça uma função que recebe dois vértices u e v que desejam se casar, e devolve a lista dos ancestrais comuns deles. Qual a complexidade da função?

9. Percorra o digrafo abaixo em profundidade (a partir do vértice de menor índice e supondo que as listas de adjacências estão ordenadas) fazendo o seguinte:

   - anote o instante de tempo em que começa e termina o percurso dfs de cada vértice;
   - identifique cada arco como arco da arborescência dfs, descendente, retorno, ou cruzado.
   - identifique as componentes fortemente conexas do digrafo.





10. Considere agora o percurso bfs e a mesma classificação dos arcos. Podem existir arcos descendetes, de retorno e cruzados? Justifique.

dois tipos de cruzado e nenhum descendente

11. Mostre que o algoritmo de Dijkstra pode dar resultados incorretos na presença de arcos de custo negativo.

12. Considere um país situado em um arquipélago em que o presidente mandou construir diversas pontes de mão única ligando as diferentes ilhas do arquipélago. O ministro da infraestrutura está, agora, preocupado com a segurança do país frente a desastres naturais. Faça uma função que, dadas duas ilhas s e t, encontre, se existir, uma ponte no grafo, cuja remoção aumenta a distância entre s e t. Qual a complexidade da sua função?

13. Chamamos de *clique* de um grafo, um conjunto S de vértices em que todos são vizinhos de todos, ou seja, para todo par $u, v \in S$, $u-v$ é aresta do grafo. Faça uma função que receba um inteiro k e devolva, se existir, uma clique com k vértices de um grafo. Qual a complexidade de sua função?

14. Considere agora um grafo formado a partir de uma rede social, em que dois vértices estão ligados se as pessoas correspondentes são "amigas" nesta rede social. Use a função acima para encontrar o tamanho da maior clique do grafo. Qual a complexidade de seu algoritmo?