Universidade Federal da Fronteira Sul - Ciência da Computação

Disciplina GEN254: Grafos

2025.1

Trabalho Nº 1 - Zoológico

Prazo de entrega: **Consultar a página do trabalho.**

Linguagem para implementação: **C++**.

Professor: Andrei Braga

Zoológico

Considere o seguinte problema. Os funcionários de um zoológico estão criando um novo espaço de convivência para os animais. Este espaço consistirá em várias áreas isoladas, cada uma destas áreas isoladas abrigando um ou mais grupos de animais. Dois grupos de animais podem ser acomodados em uma mesma área isolada se os animais destes grupos não se atacam. Para criar um espaço de convivência pacífico e reduzir custos, os funcionários do zoológico devem resolver o seguinte problema: É possível acomodar todos os grupos de animais em apenas duas áreas isoladas e evitar que aconteçam ataques?

Neste trabalho, você deve resolver o problema acima utilizando um grafo. Para isso, você deve determinar o que representam os vértices e as arestas do grafo.

---

1. Implementação - Entrega pelo Moodle

Para realizar o trabalho, você deve implementar uma classe que representa um **grafo simples** como uma **matriz de adjacências**. Você deve escrever os métodos a seguir, cujo objetivo é retornar `verdadeiro` se o grafo é bipartido e `falso` caso contrário:

- `eh_bipartido_1`
    - Utiliza o seguinte algoritmo recursivo: Remova um vértice **v** do grafo e, recursivamente, tente particionar os vértices restantes em dois conjuntos independentes. Se **v** não tem vizinhos no primeiro conjunto independente, então adicione **v** a este conjunto. Caso contrário, se **v** não tem vizinhos no segundo conjunto independente, então adicione **v** a este conjunto. Se nenhum dos casos ocorre, então retorne que o grafo não é bipartido. (**Atenção:** Este algoritmo não é um algoritmo garantidamente correto para testar se um grafo é bipartido.)
    - No algoritmo acima, a cada etapa da recursão, deve ser removido o vértice existente com o menor índice.
    - O algoritmo acima deve ser executado e a sua resposta deve ser retornada.
    - **Dica:** Na implementação do algoritmo acima, você não precisa remover, de fato, vértices do grafo. Em vez disso, você pode apenas registrar quais foram os vértices removidos.
- `eh_bipartido_2`
    - Utiliza a seguinte modificação do algoritmo de busca em profundidade: A cada novo vértice visitado **v**, adicione **v** a um segundo conjunto caso o vértice visitado anteriormente tenha sido adicionado a um primeiro conjunto e vice-versa. Além disso, caso **v** tenha um vizinho que já tenha sido visitado anteriormente e que esteja no mesmo conjunto, retorne que o grafo não é bipartido.
    - O algoritmo acima deve ser executado até que todos os vértices do grafo tenham sido visitados.

Você deve escrever um programa que constrói um grafo, executa operações no grafo e depois, se necessário, explicitamente o destrói. O seu programa deve processar informações que determinarão as operações a serem executadas no grafo, o que deve ser feito de acordo com as **Seções Entrada** e **Saída** abaixo.

Entrada

A primeira linha da entrada contém dois inteiros **A** (**A** > 0) e **R** (**R** ≥ 0), sendo **A** o número de grupos de animais do zoológico e **R** o número de relações entre os grupos de animais. Cada uma das **R** linhas seguintes contém um inteiro **X**, um inteiro **Y** e um caractere **Z** separados por espaços em branco. Estes valores indicam o seguinte:

- Se **Z** é `A`, os grupos de animais **X** e **Y** se atacam;
- Se **Z** é `N`, os grupos de animais **X** e **Y** não se atacam.

Saída

A saída deve consistir em duas linhas, de acordo com o seguinte:

- O seu programa deve executar o método `eh_bipartido_1` (descrito acima) e imprimir o seguinte:
  - caso o resultado do método seja `verdadeiro`, uma linha contendo o texto `SIM`
  - caso contrário, uma linha contendo o texto `NAO`
- O seu programa deve executar o método `eh_bipartido_2` (descrito acima) e imprimir o seguinte:
  - caso o resultado do método seja `verdadeiro`, uma linha contendo o texto `SIM`
  - caso contrário, uma linha contendo o texto `NAO`

Exemplos de execução

Entrada	Saída
6 9 2 5 A 0 4 A 3 5 A 4 1 N 1 3 A 0 5 A 0 3 N 1 4 N 2 4 A	SIM SIM

Entrada	Saída
7 10 0 1 A 0 3 A 0 4 A 0 5 A 1 2 A 1 3 N 2 3 A 4 5 N 4 6 A 5 6 A	NAO SIM

Observações:

- Para a realização dos testes automáticos, a compilação se dará da seguinte forma:
  `g++ -pedantic -Wall *.cpp -o main -lm -lutil`

2. Análise - Entrega pelo SIGAA

Você deve enviar um arquivo PDF contendo o seu nome, a sua matrícula e as respostas do item abaixo.

Selecione pelo menos três dos testes cadastrados no Moodle para os quais o resultado do método `eh_bipartido_1` é `falso` e o resultado do método `eh_bipartido_2` é `verdadeiro`. Para cada um dos testes selecionados, faça o seguinte:

- Procure entender por que o resultado do método `eh_bipartido_1` é `falso` (deveria ser `verdadeiro`). Apresente uma explicação para o resultado do método.