Universidade Federal do Rio de Janeiro (UFRJ) - 2021
Curso: Ciência da Computação **Período:** 2021.2 (Remoto)
Disciplina: Algoritmos e Grafos **Professor:** Mitre Costa Dourado

Detalhes sobre arquivos da implementação - 2º trabalho

Arquivos do programa

Vertex.java: Onde temos a implementação dos vértices do grafo. Cada vértice possui um id (valor inteiro) e uma lista de vértices vizinhos.

Graph.java: É a classe em que implementamos o grafo. Possui um conjunto de vértices. Possui também métodos básicos tais como:

- printGraph() → imprime os vértices e vizinhos do grafo;
- addVertex() → adiciona um vértice no grafo usando. Pode ser usada addVertex(int id) que adiciona usando um id ou addVertex(Vertex v) que adiciona um vértice v ao grafo;
- hasEdge(Integer v1, Integer v2) → dado dois inteiros verifica se há uma aresta entre eles. Para isso verifica se cada um contém o outro em sua vizinhança, em caso positivo a aresta existe;
- isUndirected() → Verifica se o grafo é direcionado. É utilizado no início para fazer essa verificação. Em caso positivo, informa com uma mensagem e termina o programa. Para verificar se é não direcionado para cada vértice eu checa-se o vizinho, se o vizinho não é vizinho do vértice, então o grafo é direcionado.

  Os outros métodos da classe são mais específicos para auxiliar na verificação se o grafo é cografo ou não e para gerar/mostrar o subgrafo que comprova o contrário. Segue as funções com mais detalhes abaixo:
- combinaArray() → Basicamente dado um array de inteiros e um valor r, realiza todas as possíveis combinações r a r no array e salva em uma lista de lista de inteiros, em que cada lista de inteiros é uma combinação r a r do array de entrada. Realiza tudo isso de forma recursiva;
- combinaArrayNumANumElementos(int[] array, int quantElemCombinar) → Dados um array de inteiros e uma quantidade de elementos para combinar, chama a função descrita acima para realizar as combinações r a r dos elementos. Ao longo do processo, guarda e retorna em uma lista de lista de inteiros.

- generatePermutations(List<Integer> list) → Dado uma lista de inteiros, retorna uma lista de listas de inteiros com todas as possiveis permutações do conjunto de entrada;
- getSubgraphFromSetVertexIntegers(List<Integer> setVertexIntegers) → Dado um conjunto de vértices inteiros, retorna um subgrafo desses vértices dados como entrada
- isCograph() → Verifica se o grafo do objeto criado é um cografo. Em caso positivo, informa em uma simples mensagem e caso negativo, mostra o subgrafo que comprova isso com seus vértices e vizinhança. De acordo com a ideia de uma das referências bibliográficas abaixo:

  Teorema 4 (Corneil, Lerchs e Burlingham (1981)). *Um grafo $G = (V, E)$ é um cografo se e somente se não contém um P_4 como um subgrafo induzido.*

  Podemos pensar em uma maneira de buscar por grafos P4 de forma mecânica. A forma pensada foi gerar todas as combinações 4 a 4 elementos dos vértices do grafo e para cada combinação (conjunto de 4 vértices) gerar as permutações desse conjunto. Para cada permutação, verificar entre os 4 elementos 2 a 2, se existe uma aresta por meio do método hasEdge(Integer v1, Integer v2 até formar um P4. Enfim, de maneira geral, se não forem encontradas P4 então o grafo é um cografo assim como demonstrado no Teorema da imagem acima.

FileController.java: É o controlador de arquivos. Serve basicamente o grafo do arquivo de entrada na pasta myfiles e guarda os vértices e seus vizinhos num objeto do tipo grafo

AlgGrafos.java: Basicamente cria um Grafo, abre o arquivo de entrada “grafo.txt” na pasta myfiles e lê os vértices e a vizinhança para dentro do Grafo usando o FileController. Em seguida, é verificado se o grafo é direcionado e caso positivo o algoritmo se encerra. Caso negativo o próximo passo do algoritmo é verificar se o grafo dado como entrada é cografo assim como descrito mais acima

Bibliografia

BUSCA EM LARGURA LEXICOGRÁFICA E ALGORITMOS DE SOLUÇÃO EXATA PARA O PROBLEMA DA CLIQUE MÁXIMA - Dissertação de Mestrado

D. G. Corneil, Y. Perl, L. K. Stewart, A linear recognition algorithm for cographs, SIAM Journal on Computing, Vol. 14, No. 4 : pp. 926-934, 1985