UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS – PICOS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
PROFESSOR: Juliana Oliveira de Carvalho

Trabalho de Estruturas de dados 2

Autor:

David Marcos Santos Moura

Picos-Pi, Novembro de 2021

- **Resumo do projeto:**

  Inicialmente vamos falar um pouco sobre Estruturas de dados, **Estrutura de dados** é o ramo da computação que estuda os diversos mecanismos de organização de dados para atender aos diferentes requisitos de processamento. Posto isso , este relatório tem como objetivo demonstrar e analisar como é utilizados na prática conceitos de manipulação de dados para solução de alguns problemas práticos.

- **Introdução:**

  Neste relatório vamos analisar como foi resolvido o problema proposto, para implementar o algoritmo de Dijkstra para o caminho mais curto em grafo. Para solução destes problemas foi utilizada a linguagem de programação C e foram empregados alguns conceitos importantes para resolução de tais problemas, dentre eles estão: ponteiros, listas e vetores.

- **Seções específicas:**

  Neste tópico iremos abordar de maneira resumida como foi solucionado o problema proposto:

  Resumo do problema:

  Implemente em C o algoritmo de Dijkstra para o caminho mais curto em grafo.

  Resumo da solução:

  Para solução do problema proposto foi desenvolvido as seguintes funções:

  Para construção do grafo, foi utilizado um vetor para armazenar os vértices do grafo, e cada vértice possui uma lista onde cada Nó da lista armazena os vértices adjacentes e o peso da aresta entre os vértices do grafo.

  Funções para criação do grafo:

  aloca_adjacencia(): Essa função irá receber como entrada um ponteiro para uma estrutura do tipo lista, que armazenará as informações de um Nó da lista, esse ponteiro irá apontar para um espaço alocado na memória para um Nó e armazena o vértice e o peso da aresta, e esse ponteiro será devolvido pela função por referência.

  insere_l_adjacencia(): Essa função recebe um No que armazena o valor de um vértice e o peso de sua aresta e o ponteiro que aponta para início da lista

e insere o No no fim da lista, essa função é responsável por insere vértices adjacentes a um vértice do grafo na lista do vértice

void cria_vertice(): Recebe um vetor que armazena os vértices do grafo, o valor do vértice que se deseja criar e o valor do vértice que é adjacente ao vértice no grafo que será criado e também o peso da aresta entre os vértice essa função irá inserir o vértice no vetor de vértices e inserir na lista do vértice o seu vértice adjacente.

Funções para implementação do algoritmo de Dijkstra:

Para o cálculo da menor distância é utilizado um vetor de distâncias com o mesmo tamanho do vetor do grafo que se deseja analisar e em cada posição do vetor é armazenada a menor distância do vértice inicial até aquele vértice, o seu vértice predecessor no grafo e o seu estado 'A' = aberto e 'F' fechado, aberto significa que ainda não foi definida a menor distância para o vértice e fechado significa que já foi encontrado a menor distância do vértice inicial até o vértice inicial.

inicializa_distancia(): Recebe o vetor de distâncias, o tamanho do vetor e o valor do vértice inicial que se deseja analisar, percorre o vetor, inicializa todas as distâncias com infinito, seus predecessores com -1, inicializa todos os vértices como aberto e inicializa o vértice inicial com a distância de 0.

existe_aberto(): Recebe o vetor de distâncias e verifica se tem algum vértice que é aberto ou seja que não foi visitado ainda é a função retorna 1 se existe algum vértice aberto e 0 se todos os vértices já estão fechados.

menor_distancia(): Recebe o vetor de distâncias e o tamanho do vetor e busca no vetor de distância o índice da posição do vetor que possui a menor distância e ainda não foi visitado no grafo e retorna a posição do vetor que possui a menor distância.

relaxamento(): Recebe um vértice do grafo e o vetor de distância e o índice da posição do vetor que possui a menor distância e calcula e atualiza o vetor de distância as distâncias dos vértices adjacentes ao vértice recebido.

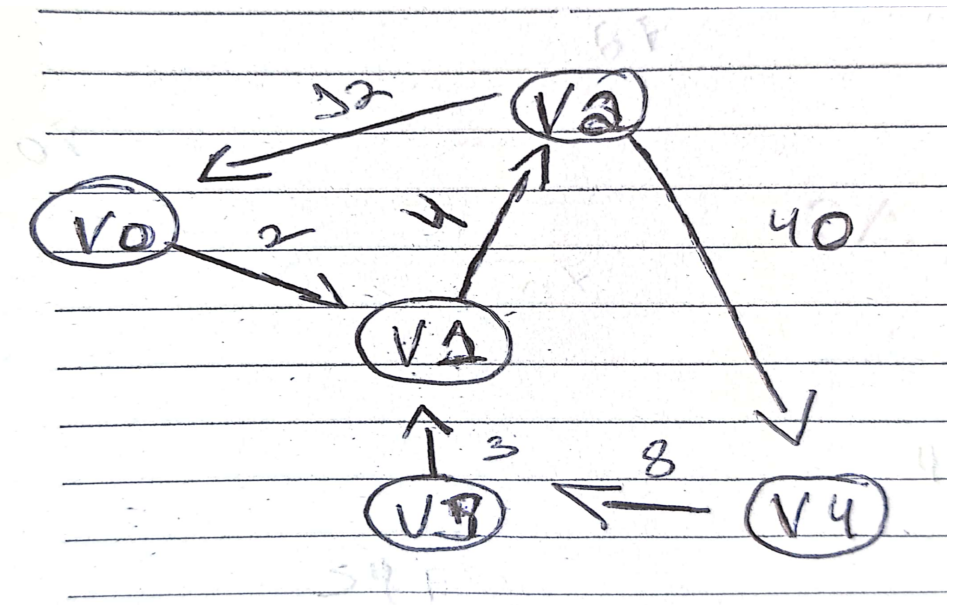
Dijkstra(): Recebe o vetor de vértices que representa o grafo, um vértice inicial que se deseja analisar , o vetor de distância e a quantidade de vértices do vetor, essa função calcula as distâncias do vértice do grafo para seus vértices adjacentes no grafo e esse cálculo é feito de forma que faça no vetor de distâncias fique com as menores distâncias do vértice inicial para os demais vértices do grafo e é devolvido pela função o vetor de distância contendo as menores distâncias do vértice inicial até os demais vértices do grafo.

- **Resultados da Execução do Programa:**

  Neste tópico iremos analisar dado algumas entradas como os códigos dão suas saídas:

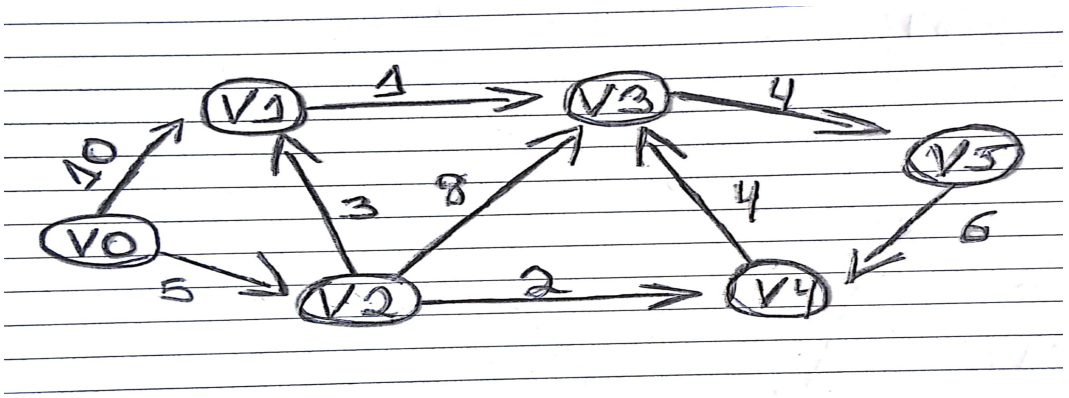
  O código a seguir foi executado pelo terminal linux.

  Grafo teste 1:



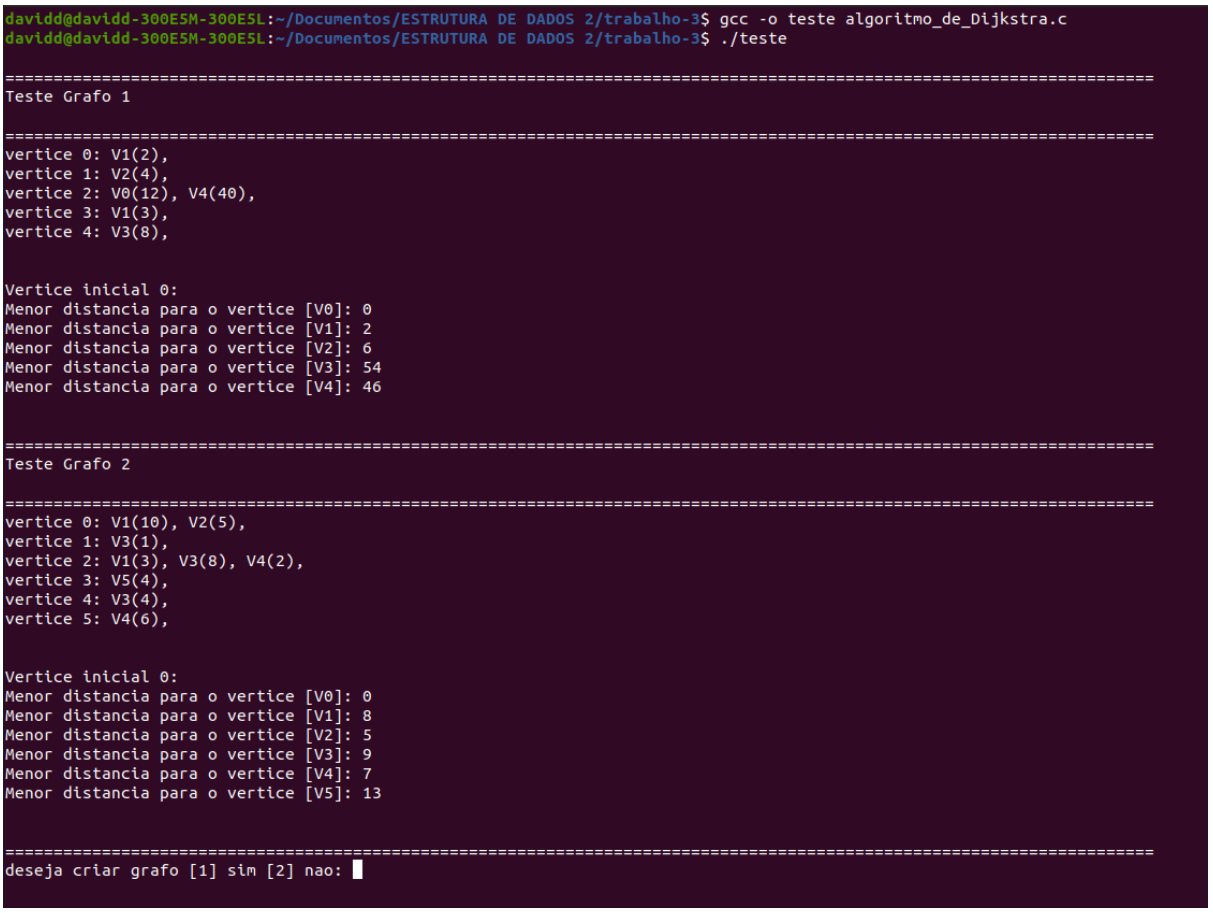
  Na imagem acima mostra o grafo que foi utilizado para o teste 1 do algoritmo de Dijkstra.

  Grafo teste 2:



  Na imagem acima mostra o grafo que foi utilizado para o teste 2 do algoritmo de Dijkstra.

Execução do algoritmo de Dijkstra mostrando o processamento dos grafos de teste 1 e 2 mostrado acima:



Na imagem acima mostra o como o código da sua saída, e como o algoritmo de Dijkstra da sua saída analisando o grafo de teste 1 e o grafo de teste 2, podemos ver que o código imprime o grafo criado e mostra em ambos os casos a menor distância do vértice inicial 0 até os demais vértices do grafo o código também dá a opção para o usuário também criar seu próprio grafo e executar o algoritmo de Dijkstra.

Conclusão:

Por fim, podemos concluir que este trabalho teve como objetivo proporcionar para os estudantes um aprofundamento em conceitos importantes para a disciplina como manipulação de dados ,processamento de dados, manipulação de ponteiros e estruturas de dados como grafos, então podemos ver que através desta atividade prática foi possível aumentar o nosso conhecimento sobre a linguagem de programação C e sobre conceitos importantes para a disciplina de estruturas de dados 2.