Aluno: Allan Cordeiro Rocha de Araújo

Questão 1

Item 1

O algoritmo de Dijkstra serve para achar o caminho mais curto entre dois vértices de um grafo dirigido ou não-dirigido com arestas de peso positivo com tempo computacional O(V log(V + E)).

Item 2

```
#include <stdio.h>

#include <stdlib.h>


//A partir daqui será para criar e adicionar valores ao Grafo

//-------------------------------------------------------------------------------------------------------------------------//

struct grafico{

	int v;//numero de vertices

	int a;//numero de arcos do grafo

	int **adj;


};


//Um Graph é um ponteiro para graph, um Graph tem um ponteiro para graph

typedef struct grafico *Graph;


int** inicializarMatriz(int linhas,int colunas,int valor){

		int **m = (int**)malloc(linhas * sizeof(int*));//aloca um vetor de ponteiros

		int i,j;

		for(i=0;i<linhas;i++){
```

```
                m[i] = (int*)malloc(colunas * sizeof(int));//aloca um vetor de inteiros´para cada posição do vetor de ponteiros
                for(j=0;j<colunas;j++){
                        m[i][j] = valor;
                }
        }

        return m;
}

Graph inicioMatriz(int v){
        Graph G = malloc(sizeof *G);
        G->v = v;
        G->a = 0;
        G->adj = inicializarMatriz(v,v,0);

        return G;
}

void inserirValorArcos( Graph G, int vertice1, int vertice2,int custo) {
  if (G->adj[vertice1][vertice2] == 0) {
    G->adj[vertice1][vertice2] = custo;
    G->adj[vertice2][vertice1] = custo;
    G->a++;
  }
}
//-----------------------------------------------------------------------------------------//

//Agora o algoritmo de Dijkstra
void dijkstra (Graph G,int Vi,int *dis){//calcula as distancias e armazena no VETOR dis[]
```

```
	/*
	Ele irá calcular a distancia da raiz Vi até todos os outros vértices armazenados no vetor dis[]
	*/

	char vis[G->v];
	memset (vis, 0, sizeof (vis));
	memset (dis, 0x7f, sizeof (dis));

	dis[Vi] = 0;
	int t, i;

	for (t = 0; t < G->v; t++){
		int v = -1;
		for (i = 0; i < G->v; i++){
			if (!vis[i] && (v < 0 || dis[i] < dis[v]) )
				v = i;
		}
		vis[v] = 1;

		for (i = 0; i < G->v; i++){
			if (G->adj[v][i] > (dis[v] + G->adj[v][i]) && dis[i] > (dis[v] + G->adj[v][i]) )
				dis[i] = dis[v] + G->adj[v][i];
		}

	}
}

int main(void){
```

```
	Graph g;
	g = inicioMatriz(4);


	int valorDistanciaCadaVertice[g->v];
	int raiz;


	dijkstra(g,raiz,valorDistanciaCadaVertice); // a distancia da raiz ate cada vértice armazenada no vetor valorDistanciaCadaVertice[]




	return 0;
}
```

Item 4

Arestas A,B,C,D.

Um arco de A para B com custo -1.

Um arco de A para C com custo -2.

Um arco de B para D com custo 1.

Um arco de C para D com custo 1.