Resolução da Lista 1

> Da disciplina de Algoritmos e Estruturas de Dados II

Questão 1

Vértice: a unidade fundamental da qual grafos são formados. Tratam-se de objetos, do ponto de vista da *Teoria dos Grafos*, inexpressivos e indivisíveis os quais podem estar conectados entre si por arestas.

Grau: O número de arestas a incidir sobre determinado vértice, somadas as arestas direcionadas a este e direcionadas à partir deste.

Caminho: A sequência de vértices tal que de cada um destes existe uma aresta para o seguinte.

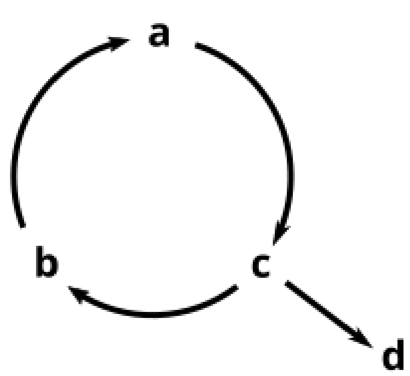
Questão 2

Ciclos são caminhos aqueles que se iniciam e terminam em um mesmo vértice. Estes podem ser distinguidos entre si por suas demais características:

Ciclo Eureliano: O ciclo onde cada **aresta** da sequência ocorre uma única vez.

Ciclo Hamiltoniano: O ciclo onde cada **vértice** da sequência ocorre uma única vez.

Questão 3



Questão 4

Lema do aperto de mão

Seja $G = (V, A)$ um grafo não-direcionado em que V descreve o conjunto de vértices $V = \{v_1, v_2, \ldots, v_n\}$ e A o conjunto de pares ordenados que descreve arestas entre vértices $A = \{(v_i, v_j), (v_l, v_k), \ldots\}$. Temos que:

- Cada **aresta** A incide exatamente sobre **dois vértices** v.
- O **grau** de cada vértice, $\deg_G(v)$ nada mais é do que o número de arestas que incidem sobre ele.
- Logo, a soma dos graus de todos os vértices, S_g, nada mais é que a todas as arestas duas vezes:

$$S_g = \sum_{v \in V} \deg_G(v) = 2|A|$$

Corolário

Ao subtrairmos de S_g os graus de todos os vértices de grau par, resta-nos a soma de todos os graus dos vértices de grau impar em V, o que estamos buscando.

Ou seja,

$$\sum_{v \in V : \deg_G(v) = 2k+1} \deg_G(v) = S_G - \sum_{v \in V : \deg_G(v) = 2k} \deg(v)$$

Sabemos que $S_G = 2|A|$, um valor par e a soma de graus pares não pode ser outra coisa senão outro número par, que denominaremos $2k$. Logo,

$$\sum_{v \in V : \deg_G(v) = 2k+1} \deg_G(v) = 2|A| - 2k = 2(|A| - k)$$

Dado que esta é uma soma composta exclusivamente por graus **impares** para esta resultar em um número par $2(|A| - k)$ necessita haver um número **par** de tais termos para que ambos os lados da equação sejam **pares**. Logo, existe um número par de vértices de grau ímpar em G. $\blacksquare$

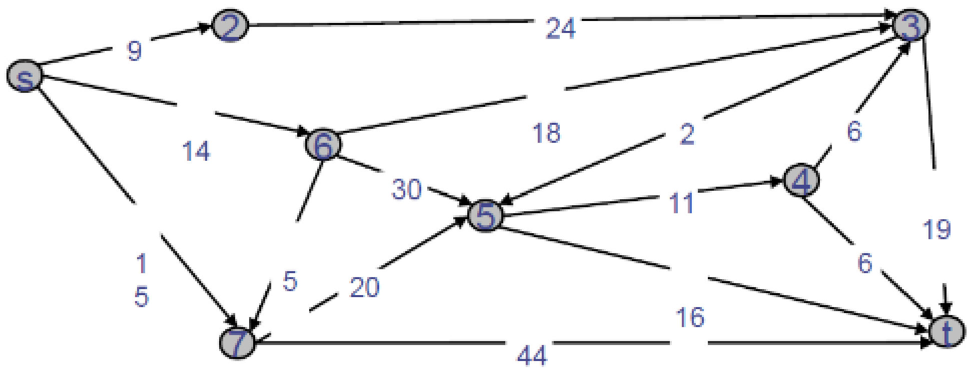
Questão 5

O algoritmo de Dijkstra pode ser descrito pelo seguinte pseudo-código:

```
Dijkstra (grafo, pesos, s)
    inicializaEstimativa(G, s)
    S ← ∅
    Q ← filaDePrioridade(grafo.vértices)
    enquanto Q ≠ ∅
        u ← extrairMínimo(Q)
        S ← S ∪ {u}
        para cada vértice v ∈ grafo.adjacentes[u]
            se d[v] > d[u] + pesos(u,v)
                relaxar(u,v,pesos)
```

Sendo que,

- `inicializaEstimativa` associa à cada vértice em `G` uma estimativa da distância destes em relação ao vértice `s`, inicialmente como `∞` senão de `s` para `s`, que é tida como `0`.
- `filaDePrioridade` gera uma fila em ordem dos vértices com menor distância com relação à `s`.
- `extrairMínimo` retira do início desta fila o índice de um vetor `u`.
- `relaxar` ajusta a estimativa de distância se a distância percorrida até este desde o vértice inicial, e passando pelo atual vértice, é menor que a estimativa anterior.



Assim sendo, dado o grafo acima, a operação do algoritmo de Djikstra se dá da seguinte maneira:

- Inicia-se em `s` todas as estimativas de distância para os vértices adjacentes são atualizadas.
- Identifica-se `2` enquanto o vértice mais próximo, atualiza-se as estimativas de distância da aresta 2 à 3. A menor estimativa de distância para se chegar à `3` fica posta em 33.
- O próximo vértice mais próximo é `6`, a estimativa de distância para `3` é atualizada para 32, a estimativa de distância para `5` fica posta em 44.
- O próximo vértice mais próximo é `7`, a estimativa de distância para `5` é atualizada para 35, a distância para `t` é posta em 59.
- O próximo vértice mais próximo é `3` a estimativa de distância para `t` é atualizada para 52.
- O próximo vértice mais próximo é `5`, a estimativa de distância para `4` fica posta em 46, a estimativa de distância para `t` é atualizada para 51.
- O próximo vértice mais próximo é `4`, caminhos mais breves não são encontrados.

Logo, encontra-se que o caminho mais curto para `t` é `s → 2 → 3 → 5 → 4 → t`, como um peso total de 51.

Questão 7

```c
/* Tests if a non-directed graph represented in a matrix is fully connected */

int countVertices(graph *g, int vector, int vectorCount, bool *foundVertices) {
    int i;
    foundVertices[vector] = true;
    vectorCount++;

    for (i = 1; i < g->vertices; i++)
        if (g->weights[vector][i] != EOF && foundVertices[i] == false)
            vectorCount = countVertices(g, i, vectorCount, foundVertices);
    return vectorCount;
}

bool isConnected(graph *g) {
    return (g && countVertices(g, 0, 0, calloc(g->vertices, sizeof(bool))) ==
                  g->vertices);
}
```

Questão 9

Verdadeiro. Uma ordenação topológica consiste na ordenação linear dos vértices em um grafo dirigido em uma sequência tal que, partindo de um vértice inicial, cada vetor seguinte é conectado à um vértice anterior por uma aresta a ele direcionada. Assim sendo, este grafo não pode ser cíclico pois, o fosse, o vértice inicial teria um predecessor posterior a este na sequência, rompendo assim o parâmetro de ordenamento.

Questão 10