IST - LMAC, MEBiom

Elementos de Programação

16 de Dezembro de 2016 Ficha 3A Duração: 15m

Número: _____ **Nome:** _____

Considere grafos dirigidos em que cada aresta tem associado um custo (positivo), com as seguintes operações:

- `grvazio(n)`: grafo sem arestas com `n` nós (identificados pelos números $0, \ldots, n-1$);
- `novaresta(g,o,d,k)`: adiciona ao grafo `g` uma aresta do nó `o` para o nó `d` com custo associado `k`;
- `dim(g)`: número de nós do grafo `g`;
- `custos(g,o,d)`: lista dos custos associados a cada uma das arestas do nó `o` para o nó `d` no grafo `g`;
- `semsaidaQ(g,o,k)`: `True` se não existe em `g` uma aresta a partir do nó `o` para algum outro nó cujo custo não seja superior a `k`, e `False` caso contrário.

Pretende-se, em *Python*, representar cada grafo dirigido com custos por um par da forma $(n, [(o_1, d_1, k_1), ..., (o_j, d_j, k_j)])$ onde cada triplo (o_i, d_i, k_i) representa uma aresta do grafo (do nó o_i para o nó d_i com custo k_i).

a) Apresente implementações eficientes apenas para as operações `novaresta` e `semsaidaQ`.

(vsff)

b) Desenvolva, sobre a camada de abstracção obtida acima e assegurando a independência da implementação, uma função `itinerarioQ` que recebendo um grafo dirigido com custos `g`, nós `o` e `d` do grafo e um valor `k` devolve `True` se existe um caminho no grafo `g` (sequência de nós, cada um ligado ao seguinte por uma aresta), que comece no nó `o` e termine no nó `d`, cujo custo total acumulado não ultrapasse o valor `k`, e `False` caso contrário.

IST - LMAC, MEBiom

Elementos de Programação

16 de Dezembro de 2016 Ficha 3B Duração: 15m

Número: _____ **Nome:** _____

Considere grafos dirigidos em que cada aresta tem associado um custo (positivo), com as seguintes operações:

- `grvazio(n)`: grafo com `n` nós (identificados pelos números $0, \ldots, n-1$) e sem arestas;
- `novaresta(g,o,d,k)`: adiciona ao grafo `g` uma aresta do nó `o` para o nó `d` com custo associado `k`;
- `dim(g)`: número de nós do grafo `g`;
- `caminho(g,o,d,k)`: devolve, se existir, um caminho no grafo `g` (lista de nós, cada um ligado ao seguinte por uma aresta), começando no nó `o` e terminando no nó `d`, cujo custo acumulado não ultrapasse o valor `k`, e devolve `[]` caso tal caminho não exista.

Pretende-se, em *Python*, representar cada grafo dirigido com custos por um par da forma $(n, [(o_1, d_1, k_1), ..., (o_j, d_j, k_j)])$ onde cada triplo (o_i, d_i, k_i) representa uma aresta do grafo (do nó o_i para o nó d_i com custo k_i).

a) Apresente implementações eficientes apenas para as operações `novaresta` e `caminho`.

(vsff)

b) Desenvolva em *Python*, sobre a camada de abstracção obtida acima e assegurando a independência da implementação, uma função `fazescala` que recebendo um grafo dirigido com custos `g`, nós `o`, `m` e `d` do grafo e um valor `k` devolve, se existir, um caminho no grafo do nó `o` para o nó `d`, que passe também pelo nó intermédio `m`, cujo custo total não exceda `k`, e devolve `[]` caso contrário.

IST - LMAC, MEBiom

Elementos de Programação

16 de Dezembro de 2016 Ficha 3C Duração: 15m

Número: _____ **Nome:** _____

Considere grafos dirigidos em que cada aresta tem associado um custo (positivo), com as seguintes operações:

- `semarestas(n)`: grafo sem arestas com `n` nós (identificados pelos números $0, \ldots, n-1$);
- `jaresta(g,o,d,k)`: adiciona ao grafo `g` uma aresta do nó `o` para o nó `d` com custo associado `k`;
- `dim(g)`: número de nós do grafo `g`;
- `arestaQ(g,o,d)`: `True` se existe no grafo `g` alguma aresta do nó `o` para o nó `d`, e `False` caso contrário;
- `mincusto(g,o,d)`: custo mínimo associado a uma aresta do nó `o` para o nó `d` no grafo `g` (se existir).

Pretende-se, em *Python*, representar cada grafo dirigido com custos por uma lista da forma $[\texttt{ars}_0, \ldots, \texttt{ars}_{n-1}]$ onde `n` é o número de nós do grafo e cada $\texttt{ars}_o$ é uma lista da forma $[(\texttt{d}_1, \texttt{k}_1), (\texttt{d}_2, \texttt{k}_2), \ldots]$ em que cada par $(\texttt{d}_j, \texttt{k}_j)$ representa uma aresta do nó `o` para o nó $\texttt{d}_j$ com custo $\texttt{k}_j$.

a) Apresente implementações eficientes apenas para as operações `jaresta` e `mincusto`.

(vsff)

b) Desenvolva em *Python*, sobre a camada de abstracção obtida acima e assegurando a independência da implementação, uma função `fazcaminho` que recebendo um grafo dirigido com custos `g`, e nós `o` e `d` do grafo devolve, se existir, um par `(cam,tot)` onde `cam` é um caminho no grafo `g` (lista de nós, cada um ligado ao seguinte por uma aresta), começando no nó `o` e terminando no nó `d`, e `tot` é o custo acumulado desse caminho, e devolve `False` caso tal caminho não exista.

IST - LMAC, MEBiom

Elementos de Programação

16 de Dezembro de 2016 Ficha 3D Duração: 15m

Número: _____ **Nome:** _____

Considere grafos dirigidos em que cada aresta tem associado um custo (positivo), com as seguintes operações:

- `semarestas(n)`: grafo com `n` nós (identificados pelos números $0, \ldots, n-1$) e sem arestas;
- `jaresta(g,o,d,k)`: adiciona ao grafo `g` uma aresta do nó `o` para o nó `d` com custo associado `k`;
- `dim(g)`: número de nós do grafo `g`;
- `acessQ(g,o,d,k)`: `True` se existe um caminho no grafo `g` (sequência de nós, cada um ligado ao seguinte por uma aresta), começando no nó `o` e terminando no nó `d`, cujo custo acumulado não ultrapasse o valor `k`, e `False` caso contrário.

Pretende-se, em *Python*, representar cada grafo dirigido com custos por uma lista da forma $[\texttt{ars}_0, \ldots, \texttt{ars}_{n-1}]$ onde `n` é o número de nós do grafo e cada $\texttt{ars}_o$ é uma lista da forma $[(d_1, k_1), (d_2, k_2), \ldots]$ em que cada par (d_j, k_j) representa uma aresta do nó `o` para o nó d_j com custo k_j.

a) Apresente implementações eficientes apenas para as operações `jaresta` e `acessQ`.

(vsff)

b) Desenvolva em *Python*, sobre a camada de abstracção obtida acima e assegurando a independência da implementação, uma função `semvolta` que recebendo um grafo dirigido com custos `g`, um nó `o` do grafo e um valor `k`, calcula a lista de todos os nós `d` do grafo que são acessíveis a partir de `o` por algum caminho com custo acumulado não superior a `k`, mas tais que partindo de `d` não seja possível regressar a `o` sem ultrapassar o custo máximo `k`.