Bruno Araujo Lima
Diego Alysson Braga Moreira
Flávio Alves dos Santos

Implementação de Algoritmos Gulosos e Programação Dinâmica

Brasil

Junho de 2016

Exercício 5.23

Dado um Grafo $G(V, E)$ não orientado e um conjunto de vértices U, onde $U \subset V$. Queremos achar uma light spanning tree a partir de G e U. Uma light spanning tree é uma árvore geradora formada a partir de G onde os nós do conjunto U são folhas dessa árvore (entretanto pode haver outras folhas nesta árvore).

Entrada

Grafo $G(V, E)$ não orientado; arestas com pesos w e o subconjunto de vértices $U \subset V$.

Saída

Uma *Lightest Spanning Tree* em que os nós de U são folhas, não necessáriamente sendo uma Árvore Geradora Mínima (*Minimum Spanning Tree*).

Algoritmo de Resolução

```
1  Algoritmo Lightest-Spanning-Tree(G,w,U)
2  Construir Grafo G'' = (V'',E''), onde:
3  V'' = V - U
4  E'' = {(u, v) : u, v pertencente a V - U && (u, v) pertencente a
       E}
5
6  aplica Kruskal para encontrar MST(G'') = T''.
7
8  construir o conjunto de arestas E:
9          para todo (u, v) pertencente a E: u pertencente a U && v
               nao pertencente a U
10 para cada aresta u pertencente a U:
11         ordene as arestas E por peso w.
12 para todas as arestas u,v pertencente a E, em ordem crescente de
       peso:
13         se find(u) != find(v):
14 adicione aresta u,v em T''.
15 union(u,v)
16 retorne T''
```

Caso de Teste A entrada é um arquivo contendo os vértices e as arestas entre os vértices. Também consta no arquivo os nós a serem removidos (nós do conjunto U). No arquivo, há três conjuntos de dados separados por uma linha em branco. Do primeiro

conjunto, o primeiro valor, constam os valores do número de nós do grafo. O segundo, o número de arestas do grafo. E o terceiro, o número de nós a serem removidos. Do segundo conjunto, em cada linha há a entrada dos dois vértices que são ligados pela aresta e o custo desta. O terceiro conjunto são os vértices a serem removidos do conjunto V de entrada.

Exemplo1:
6 8 2

1 2 2
1 4 7
2 3 6
2 4 6
3 4 2
3 5 2
4 6 6
5 6 6

3 2

Saída:
MST
Vertices 3 e 4. Peso: 2
Vertices 1 e 2. Peso: 3
Vertices 2 e 3. Peso: 3
Vertices 4 e 5. Peso: 3
Custo da MST: 11

Vertices Adicionados
Vertices 1 e 6. Peso: 2

LST
Vertices 3 e 4. Peso: 2
Vertices 1 e 2. Peso: 3
Vertices 2 e 3. Peso: 3
Vertices 4 e 5. Peso: 3
Vertices 1 e 6. Peso: 2
Custo da LST: 13

Exemplo2:
6 8 1

1 2 3
1 6 2
2 6 2
2 3 3
3 4 2
4 6 2
4 5 3
5 1 3

6

Saída:

Exemplo3:
6 10 1

1 2 1
2 6 2
5 6 2
4 6 4
2 5 2
1 5 3
1 3 3
3 5 3
4 5 4
3 4 6

2

Saída:
MST
Vertices 5 e 6. Peso: 2
Vertices 1 e 3. Peso: 3
Vertices 1 e 5. Peso: 3
Vertices 4 e 5. Peso: 4
Custo da MST: 12

Vertices Adicionados
Vertices 1 e 2. Peso: 1

LST
Vertices 5 e 6. Peso: 2
Vertices 1 e 3. Peso: 3
Vertices 1 e 5. Peso: 3
Vertices 4 e 5. Peso: 4
Vertices 1 e 2. Peso: 1
Custo da LST: 13

Exercício 6.8

Dadas duas strings x = x1, x2, x3, . . . , xn e y = y1, y2, y3, . . . , ym. Encontre a maior substring comum entre x e y. A substring pode ser qualquer parte da string, inclusive ela toda. Se não houver subsequência comum, a saída deve ser 0. A comparação é case sensitive ('x' != 'X'). Mostre como isso é feito em tempo O(mn).

Entrada

A entrada contém vários casos de teste. Cada caso de teste é composto por duas linhas, cada uma contendo uma string. Ambas strings de entrada contém entre 1 e 50 caracteres ('A'-'Z','a'-'z' ou espaço ' '), inclusive, ou no mínimo uma letra ('A'-'Z','a'-'z').

Saída

O tamanho da maior subsequência comum entre as duas Strings.

Algoritmo de Resolução

```
1  Algoritmo Compara-Substring()
2
3  receba string1 e string2
4  var numeroMaximo = 0
5  var numeroMaximoAtual = 0
6
7  para cada caracter da string1 faca:
8          para caracter da string2 faca:
9                  se o caracter da string1 e string2 forem iguais
10                         incrementa numeroMaximoAtual
11                         analisa proximo caracter da string1
12                         se o numeroMaximoAtual for maior que
                              numeroMaximo
13                                 incrementa numeroMaximo
14                 senao
15                         numeorMaximoAtual = 0
16                         volta posicao da string1
17                 fim-se
18         fim-para
19         escreve numeroMaximo
20 fim
```

Caso de Teste

Dadas duas strings como entrada do programa é retornado o tamanho da maior substring em comum.

Exemplo 1:
Entrada:
abcdabcd
abcabc

Saída:
3
abc

Exemplo 2:
Entrada:
Ordem e progresso
sucesso

Saída:
4
esso

Exemplo 3:
Entrada:
nem sempre foi assim
antes tarde do que nunca

Saída:
2
e
(Neste último caso, o caracter "espaço"também é levado em consideração na cadeia.)