

Relatório EP3 - MAC0323 Algoritmos e Estruturas de Dados II

Sabrina Araújo - nºUSP 12566182

Tarefas básicas de grafos

Entrada do arquivo de texto

O algoritmo lê o arquivo de texto para a entrada da estrutura do grafo na função `leArquivo`. O nome do arquivo deve ser "grafo.txt" e estar no mesmo diretório do programa. A lista de adjacência deve ter inteiros de 0 e V-1.

Implementação do Grafo

O grafo foi implementado utilizando a classe `vector` para criar V (número de vértices) listas ligadas. Cada lista contém os adjacentes de um dado vértice.

Função `distancia()`

Dado um vértice `u`, a função calcula a distância de `u` até todos os vértices do grafo utilizando a função `bfs(int i)` que consiste na busca em largura. A ideia dessa função é visitar um vértice `i`, em seguida seus vizinhos e assim por diante, desse modo, é possível encontrar o menor caminho entre `i` e os demais vértices.

Função `compConexas()`

Determina o número de componentes conexas e o tamanho de cada componente de um grafo utilizando a função `dfs()` que consiste na busca em profundidade. A ideia dessa função é em cada passo examinar um vértice, marcar que a busca já o examinou e visitar cada um de seus vizinhos que ainda não foi visitado, assim, é possível determinar as componentes conexas.

Exemplo

Seja a seguinte lista de adjacência:

```
10 8
1 3
2 5
6 9
3 5
4 6
2 7
0 8
5 8
```

O programa gera a saída:

```
Distancia de 0 ate 8: 1
Distancia de 0 ate 5: 2
Distancia de 0 ate 2: 3
Distancia de 0 ate 3: 3
Distancia de 0 ate 7: 4
Distancia de 3 ate 1: 1
Distancia de 3 ate 5: 1
Distancia de 3 ate 2: 2
Distancia de 3 ate 8: 2
Distancia de 3 ate 7: 3
Distancia de 3 ate 0: 3
Distancia de 4 ate 6: 1
Distancia de 4 ate 9: 2
Distancia de 5 ate 2: 1
Distancia de 5 ate 3: 1
Distancia de 5 ate 8: 1
Distancia de 5 ate 7: 2
Distancia de 5 ate 1: 2
Distancia de 5 ate 0: 2
Distancia de 6 ate 9: 1
Distancia de 6 ate 4: 1
Distancia de 7 ate 2: 1
Distancia de 7 ate 5: 2
Distancia de 7 ate 3: 3
Distancia de 7 ate 8: 3
Distancia de 7 ate 1: 4
Distancia de 7 ate 0: 4
```

```
Distancia de 8 ate 0: 1
Distancia de 8 ate 5: 1
Distancia de 8 ate 2: 2
Distancia de 8 ate 3: 2
Distancia de 8 ate 7: 3
Distancia de 8 ate 1: 3
Distancia de 9 ate 6: 1
Distancia de 9 ate 4: 2
```

```
Numero de componentes conexas: 2
Tamanho das componentes
componente 1: 7
componente 2: 3
```

Os grafos legais: geradores

No arquivo geradores.cpp há 3 geradores de entrada para o programa principal.

`simples()`

A função `simples()` gera um número aleatório para V (número de vértices) e para E (número de arestas) e a partir disso gera uma lista de adjacência aleatória.

`palavras()`

A função `palavras()` recebe uma lista com n palavras de um arquivo `palavras.txt` e a partir disso cria uma lista de adjacência com números entre 0 e $V-1$.

`erdosRenyi()`

A função `erdosRenyi()` recebe um número de vértices V e uma probabilidade p e a partir disso conecta cada vértice com os outros com probabilidade p .

Propriedades de grafos: testes

Grafos de palavras

Nestes grafos, cada vértice é uma palavra de k letras, e duas palavras estão ligadas se diferem em exatamente uma letra.

Teste 1

Utilizando as palavras tears-sears-stars-stare-stale-stile-smile como entrada do programa gerador, temos a seguinte lista de adjacência:

```
7 8
0 1
1 2
1 3
2 3
2 4
3 4
4 5
5 6
```

A partir dessa lista criamos um grafo com o programa principal, que gera a saída:

```
Distancia de 0 ate 1: 1  Distancia de 3 ate 0: 2
Distancia de 0 ate 2: 2  Distancia de 3 ate 5: 2
Distancia de 0 ate 3: 2  Distancia de 3 ate 6: 3
Distancia de 0 ate 4: 3  Distancia de 4 ate 2: 1
Distancia de 0 ate 5: 4  Distancia de 4 ate 3: 1
Distancia de 0 ate 6: 5  Distancia de 4 ate 5: 1
Distancia de 1 ate 0: 1  Distancia de 4 ate 1: 2
Distancia de 1 ate 2: 1  Distancia de 4 ate 6: 2
Distancia de 1 ate 3: 1  Distancia de 4 ate 0: 3
Distancia de 1 ate 4: 2  Distancia de 5 ate 4: 1
Distancia de 1 ate 5: 3  Distancia de 5 ate 6: 1
Distancia de 1 ate 6: 4  Distancia de 5 ate 2: 2
Distancia de 2 ate 1: 1  Distancia de 5 ate 3: 2
Distancia de 2 ate 3: 1  Distancia de 5 ate 1: 3
Distancia de 2 ate 4: 1  Distancia de 5 ate 0: 4
Distancia de 2 ate 0: 2  Distancia de 6 ate 5: 1
Distancia de 2 ate 5: 2  Distancia de 6 ate 4: 2
Distancia de 2 ate 6: 3  Distancia de 6 ate 2: 3
Distancia de 3 ate 1: 1  Distancia de 6 ate 3: 3
Distancia de 3 ate 2: 1  Distancia de 6 ate 1: 4
Distancia de 3 ate 4: 1  Distancia de 6 ate 0: 5
```

```
Numero de componentes conexas: 1
Tamanho das componentes
componente 1: 7
```

Teste 2

Utilizando as palavras word-woad-road-read-real-peal-peat-plat-play como entrada do programa gerador, temos a seguinte lista de adjacência:

```
9 8
0 1
1 2
2 3
3 4
4 5
5 6
6 7
7 8
```

A partir dessa lista criamos um grafo com o programa principal, que gera a saída:

```
Distancia de 0 ate 1: 1  Distancia de 2 ate 0: 2  Distancia de 4 ate 1: 3
Distancia de 0 ate 2: 2  Distancia de 2 ate 4: 2  Distancia de 4 ate 7: 3
Distancia de 0 ate 3: 3  Distancia de 2 ate 5: 3  Distancia de 4 ate 0: 4
Distancia de 0 ate 4: 4  Distancia de 2 ate 6: 4  Distancia de 4 ate 8: 4
Distancia de 0 ate 5: 5  Distancia de 2 ate 7: 5  Distancia de 5 ate 4: 1
Distancia de 0 ate 6: 6  Distancia de 2 ate 8: 6  Distancia de 5 ate 6: 1
Distancia de 0 ate 7: 7  Distancia de 3 ate 2: 1  Distancia de 5 ate 3: 2
Distancia de 0 ate 8: 8  Distancia de 3 ate 4: 1  Distancia de 5 ate 7: 2
Distancia de 1 ate 0: 1  Distancia de 3 ate 1: 2  Distancia de 5 ate 2: 3
Distancia de 1 ate 2: 1  Distancia de 3 ate 5: 2  Distancia de 5 ate 8: 3
Distancia de 1 ate 3: 2  Distancia de 3 ate 0: 3  Distancia de 5 ate 1: 4
Distancia de 1 ate 4: 3  Distancia de 3 ate 6: 3  Distancia de 5 ate 0: 5
Distancia de 1 ate 5: 4  Distancia de 3 ate 7: 4  Distancia de 6 ate 5: 1
Distancia de 1 ate 6: 5  Distancia de 3 ate 8: 5  Distancia de 6 ate 7: 1
Distancia de 1 ate 7: 6  Distancia de 4 ate 3: 1  Distancia de 6 ate 4: 2
Distancia de 1 ate 8: 7  Distancia de 4 ate 5: 1  Distancia de 6 ate 8: 2
Distancia de 2 ate 1: 1  Distancia de 4 ate 2: 2  Distancia de 6 ate 3: 3
Distancia de 2 ate 3: 1  Distancia de 4 ate 6: 2  Distancia de 6 ate 2: 4
```

```
Distancia de 6 ate 1: 5
Distancia de 6 ate 0: 6
Distancia de 7 ate 6: 1
Distancia de 7 ate 8: 1
Distancia de 7 ate 5: 2
Distancia de 7 ate 4: 3
Distancia de 7 ate 3: 4
Distancia de 7 ate 2: 5
Distancia de 7 ate 1: 6
Distancia de 7 ate 0: 7
Distancia de 8 ate 7: 1
Distancia de 8 ate 6: 2
Distancia de 8 ate 5: 3
Distancia de 8 ate 4: 4
Distancia de 8 ate 3: 5
Distancia de 8 ate 2: 6
Distancia de 8 ate 1: 7
Distancia de 8 ate 0: 8
```

```
Numero de componentes conexas: 1
Tamanho das componentes
componente 1: 9
```


- Seis graus de separação

Teste 3

Utilizando uma [lista](#) de 1000 palavras de 4 letras em português, temos uma lista de adjacência com 1000 vértices e 2793 arestas (obs: a lista da imagem não está completa):

```
1000 2793
0 55
0 201
0 591
1 119
1 142
1 217
1 241
1 268
1 344
1 394
1 403
1 535
1 592
1 619
```

A partir dessa lista criamos um grafo com o programa principal, que gera a saída (as imagens não representam a saída completa):

```
Distancia de 7 ate 970: 13
Distancia de 7 ate 971: 13
Distancia de 7 ate 712: 14
Distancia de 7 ate 594: 14
Distancia de 7 ate 944: 15
Distancia de 7 ate 875: 16
Distancia de 8 ate 19: 1
Distancia de 8 ate 310: 1
Distancia de 8 ate 878: 1
Distancia de 8 ate 175: 2
Distancia de 8 ate 705: 2
Distancia de 8 ate 97: 2
Distancia de 8 ate 402: 2
Distancia de 8 ate 744: 2
Distancia de 8 ate 970: 3
```

É possível observar que há diferentes distâncias entre os vértices, no qual podem estar acima ou abaixo de 6. Diante disso, não dá para afirmar que todos os vértices estão a seis ou menos conexões de outro vértice. Porém, a distância média é menor ou igual a 6:

```
Distancia media: 5
```

Grafos aleatórios de Erdős-Renyi

- Componente gigante

Se $p \leq (1-\epsilon)/n$ então, com alta probabilidade as componentes conexas são pequenas, com $O(\log n)$ elementos, mas se $p \geq (1+\epsilon)/n$ surge uma componente gigante no grafo.

Teste 4

Seja $p=0.2$ e $\epsilon=0.01$. Ao criar um grafo com 10 vértices e probabilidade p , temos a seguinte lista de adjacência:

```
10 14
0 8
0 9
1 3
0 2
3 6
3 9
2 4
4 7
5 7
5 9
6 7
0 7
1 8
7 8
```

A partir dessa lista criamos um grafo com o programa principal, que gera a saída:

Distancia de 0 ate 8: 1	Distancia de 3 ate 8: 2	Distancia de 6 ate 0: 2
Distancia de 0 ate 9: 1	Distancia de 3 ate 7: 2	Distancia de 6 ate 8: 2
Distancia de 0 ate 2: 1	Distancia de 3 ate 0: 2	Distancia de 6 ate 2: 3
Distancia de 0 ate 7: 1	Distancia de 3 ate 5: 2	Distancia de 7 ate 4: 1
Distancia de 0 ate 1: 2	Distancia de 3 ate 4: 3	Distancia de 7 ate 5: 1
Distancia de 0 ate 3: 2	Distancia de 3 ate 2: 3	Distancia de 7 ate 6: 1
Distancia de 0 ate 5: 2	Distancia de 4 ate 2: 1	Distancia de 7 ate 0: 1
Distancia de 0 ate 4: 2	Distancia de 4 ate 7: 1	Distancia de 7 ate 8: 1
Distancia de 0 ate 6: 2	Distancia de 4 ate 0: 2	Distancia de 7 ate 2: 2
Distancia de 1 ate 3: 1	Distancia de 4 ate 5: 2	Distancia de 7 ate 9: 2
Distancia de 1 ate 8: 1	Distancia de 4 ate 6: 2	Distancia de 7 ate 3: 2
Distancia de 1 ate 6: 2	Distancia de 4 ate 8: 2	Distancia de 7 ate 1: 2
Distancia de 1 ate 9: 2	Distancia de 4 ate 9: 3	Distancia de 8 ate 0: 1
Distancia de 1 ate 0: 2	Distancia de 4 ate 3: 3	Distancia de 8 ate 1: 1
Distancia de 1 ate 7: 2	Distancia de 4 ate 1: 3	Distancia de 8 ate 7: 1
Distancia de 1 ate 5: 3	Distancia de 5 ate 7: 1	Distancia de 8 ate 9: 2
Distancia de 1 ate 2: 3	Distancia de 5 ate 9: 1	Distancia de 8 ate 2: 2
Distancia de 1 ate 4: 3	Distancia de 5 ate 4: 2	Distancia de 8 ate 3: 2
Distancia de 2 ate 0: 1	Distancia de 5 ate 6: 2	Distancia de 8 ate 4: 2
Distancia de 2 ate 4: 1	Distancia de 5 ate 0: 2	Distancia de 8 ate 5: 2
Distancia de 2 ate 8: 2	Distancia de 5 ate 8: 2	Distancia de 8 ate 6: 2
Distancia de 2 ate 9: 2	Distancia de 5 ate 3: 2	Distancia de 9 ate 0: 1
Distancia de 2 ate 7: 2	Distancia de 5 ate 2: 3	Distancia de 9 ate 3: 1
Distancia de 2 ate 1: 3	Distancia de 5 ate 1: 3	Distancia de 9 ate 5: 1
Distancia de 2 ate 3: 3	Distancia de 6 ate 3: 1	Distancia de 9 ate 8: 2
Distancia de 2 ate 5: 3	Distancia de 6 ate 7: 1	Distancia de 9 ate 2: 2
Distancia de 2 ate 6: 3	Distancia de 6 ate 1: 2	Distancia de 9 ate 7: 2
Distancia de 3 ate 1: 1	Distancia de 6 ate 9: 2	Distancia de 9 ate 1: 2
Distancia de 3 ate 6: 1	Distancia de 6 ate 4: 2	Distancia de 9 ate 6: 2
Distancia de 3 ate 9: 1	Distancia de 6 ate 5: 2	Distancia de 9 ate 4: 3

```

Numero de componentes conexas: 1
Tamanho das componentes
componente 1: 10

```

Neste exemplo, $p \geq (1+e)/n$, assim, como proposto pelo modelo surgiu uma componente gigante no grafo.

Teste 5

Seja $p=0.01$ e $e=0.01$. Ao criar um grafo com 20 vértices e probabilidade p , temos a seguinte lista de adjacência:

```

20 4
6 18
0 14
3 16
12 17

```

A partir dessa lista criamos um grafo com o programa principal, que gera a saída:

```
Distancia de 0 ate 14: 1
Distancia de 3 ate 16: 1
Distancia de 6 ate 18: 1
Distancia de 12 ate 17: 1
Distancia de 14 ate 0: 1
Distancia de 16 ate 3: 1
Distancia de 17 ate 12: 1
Distancia de 18 ate 6: 1
```

```
Numero de componentes conexas: 16
Tamanho das componentes
componente 1: 2
componente 2: 1
componente 3: 1
componente 4: 2
componente 5: 1
componente 6: 1
componente 7: 2
componente 8: 1
componente 9: 1
componente 10: 1
componente 11: 1
componente 12: 1
componente 13: 2
componente 14: 1
componente 15: 1
componente 16: 1
```

Neste exemplo, $p \leq (1-e)/n$, assim, como proposto pelo modelo surgiram componentes conexas pequenas com $O(\log 20)$ elementos.

- Seis graus de separação

Teste 6

Seja $p=0.1$. Ao criar um grafo com 20 vértices e probabilidade p , temos uma lista de adjacência com 42 arestas.

A partir dessa lista criamos um grafo com o programa principal, que gera a saída (a imagem não está completa, pois a saída é grande):


```
Distancia de 0 ate 1: 1
Distancia de 0 ate 16: 1
Distancia de 0 ate 17: 1
Distancia de 0 ate 18: 1
Distancia de 0 ate 7: 2
Distancia de 0 ate 4: 2
Distancia de 0 ate 19: 2
Distancia de 0 ate 9: 2
Distancia de 0 ate 2: 2
Distancia de 0 ate 15: 2
Distancia de 0 ate 3: 2
Distancia de 0 ate 5: 2
Distancia de 0 ate 12: 2
Distancia de 0 ate 14: 2
Distancia de 0 ate 13: 2
Distancia de 0 ate 6: 3
Distancia de 0 ate 8: 3
Distancia de 0 ate 10: 3
Distancia de 0 ate 11: 3
```

```
Numero de componentes conexas: 1
Tamãho das componentes
componente 1: 20
Distancia media: 2
```

Neste exemplo, todos os pares de vértices possuem uma distância menor ou igual a 6, com média igual a 2, assim, validando a ideia dos seis graus de separação.

Teste 7

Seja $p=0.1$. Ao criar um grafo com 50 vértices e probabilidade p , temos uma lista de adjacência com 253 arestas.

A partir dessa lista criamos um grafo com o programa principal, que gera a saída (a imagem não está completa, pois a saída é grande):

```
Distancia de 30 ate 5: 2
Distancia de 30 ate 18: 2
Distancia de 30 ate 23: 2
Distancia de 30 ate 38: 2
Distancia de 30 ate 0: 2
Distancia de 30 ate 4: 2
Distancia de 30 ate 14: 2
Distancia de 30 ate 33: 2
Distancia de 30 ate 37: 2
Distancia de 30 ate 39: 2
Distancia de 30 ate 41: 2
Distancia de 30 ate 44: 2
Distancia de 30 ate 48: 2
Distancia de 30 ate 20: 2
Distancia de 30 ate 26: 2
Distancia de 30 ate 8: 2
Distancia de 30 ate 28: 2
Distancia de 30 ate 34: 2
Distancia de 30 ate 35: 2
Distancia de 30 ate 46: 2
Distancia de 30 ate 32: 2
```

```
Numero de componentes conexas: 1  
Tamanho das componentes  
componente 1: 50  
Distancia media: 2
```

Neste exemplo, todos os pares de vértices possuem uma distância menor ou igual a 3, com média igual a 2, assim, validando a ideia dos seis graus de separação.