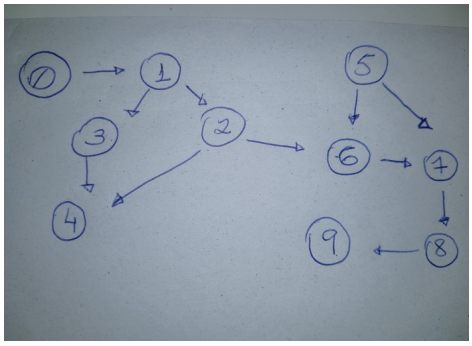
Relatório EP3

Ísis Ardisson Logullo – 7577410

1)Tarefa Básica de Grafos

Nesse EP3, primeiramente construi toda a estrutura necessária para a manipulação de Grafos. Nessa mesma classe fiz uma função que insere arcos. Tudo funciona com digrafos, basta que na inserção dos arcos você leve em consideração exatamente essa conexão A→B (A-B) e B→A (B-A). Então um arco duplo tem que ser especificado nas duas direções com a ordem dos vértices ali fazendo diferença. Usei várias classes de construção de tipos, apenas. Então tudo basicamente se divide em 2 classes. Na classe de Grafos também construi funções que calculam componentes conexas, dfs e bfs. A partir dessa estrutura, na classe principal do ep fiz tudo que era pedido nos enunciados.

Usei de teste inicial o seguinte grafo (grafo1.txt):



Tendo 2 componentes conexas, uma apenas com o 5 e a outra com o restante dos vértices. No total contém 10 vértices. E as distâncias de cada vértice seguem abaixo:

Distâncias do vértice 0:
0: 0
1: 1
2: 2
3: 2
4: 3
5: -1
6: 3
7: 4
8: 5
9: 6

Distâncias do vértice 1:
0: -1
1: 0
2: 1
3: 1
4: 2
5: -1
6: 2
7: 3
8: 4
9: 5

Distâncias do vértice 2:
0: -1
1: -1
2: 0
3: -1
4: 1
5: -1
6: 1
7: 2
8: 3
9: 4

Distâncias do vértice 3:
0: -1
1: -1
2: -1
3: 0
4: 1
5: -1
6: -1
7: -1

Distâncias do vértice 4:
0: -1
1: -1
2: -1
3: -1
4: 0
5: -1
6: -1
7: -1

Distâncias do vértice 5:
0: -1
1: -1
2: -1
3: -1
4: -1
5: 0
6: 1
7: 1

8: -1
9: -1
Distâncias do vértice 6:
0: -1
1: -1
2: -1
3: -1
4: -1
5: -1
6: 0
7: 1
8: 2
9: 3

8: -1
9: -1
Distâncias do vértice 7:
0: -1
1: -1
2: -1
3: -1
4: -1
5: -1
6: -1
7: 0
8: 1
9: 2

8: 2
9: 3
Distâncias do vértice 8:
0: -1
1: -1
2: -1
3: -1
4: -1
5: -1
6: -1
7: -1
8: 0
9: 1

Distâncias do vértice 9:
0: -1
1: -1
2: -1
3: -1
4: -1
5: -1
6: -1
7: -1
8: -1
9: 0

Para visualizar tudo isso, apenas rode o makefile em anexo, que o grafo já está ali no arquivo.
A saída com todas as infos fica gravada num arquivo “saidaEp3.txt”, gerado no seu diretório atual.

2) Propriedades de Grafos

2.1) Grafos Aleatórios – ErdosRenyi

Foi feito um programa que gera uma entrada de grafo aleatório baseado na probabilidade de 2 vértices estarem conectados. Rode “make erdos” para compilar. Um exemplo de uso é:
java ErdosRenyi 0.05 100 (cria um grafo aleatório de probabilidade 0.05 com 100 vértices)
O arquivo será gerado no seu diretório atual com o nome de random.txt. Esse arquivo é usado para ser a entrada do Ep3. Apenas retira-se a última linha e insiro ela no formato certo na primeira linha. Dessa maneira altero o makefile com esse novo arquivo de entrada e rodo o programa. Todos os arquivos gerados e suas entradas estão em anexo com o ep.

Gerei 4 tipos de grafos aleatórios, sempre com 100 vértices:

- Probabilidade: 0,01 (random1.txt):

  Vértices - 100 e Arestas - 108
  46 componentes conexas: 1 de 24 elementos, 2 de 11 elementos, 2 de 3 elementos, 7 de 2 elementos e 34 de 1 elemento.

  Tempos de execução:
  0m1,097s / 0m1,262s / 0m1,106s / 0m1,057s / 0m1,087s / 0m1,081s / 0m1,173s / 0m1,232s / 0m1,065s / 0m1,064s → média: 0m1,122s

- Probabilidade: 0,05 (random2.txt)

  Vértices - 100 e Arestas – 410
  2 componentes conexas: 1 de 99 elementos e 1 de 1 elemento.

  Tempos de execução:
  0m1,250s / 0m1,331s / 0m1,227s / 0m1,372s / 0m1,257s / 0m1,269s / 0m1,254s /
  0m1,207s / 0m1,210s / 0m1,308s → média: 0m1,268s

- Probabilidade: 0,5 (random3.txt)

  Vértices - 100 e Arestas – 4994
  1 componente conexa com os 100 elementos

  Tempos de execução:
  0m1,864s / 0m1,914s / 0m1,938s / 0m1,869s / 0m1,897s / 0m1,928s / 0m1,888s /
  0m2,125s / 0m2,505s / 0m2,119s → média: 0m2,004s

- Probabilidade: 1 (random4.txt)

  Vértices - 100 e Arestas – 9900
  1 componente conexa com os 100 elementos

  Tempos de execução:
  0m2,516s / 0m2,359s / 0m2,435s / 0m2,802s / 0m2,253s / 0m2,571s / 0m2,355s /
  0m2,362s / 0m2,531s / 0m2,313s → média: 0m2,449s

Com esses resultados podemos notar que quanto menor a probabilidade mais componentes conexas aparecem e o tempo de execução é mais rápido por ter várias componentes pequenas. Conforme a probabilidade vai aumentando, componentes gigantes vão aparecendo. Percebe-se pela quantidade de arcos e por existir apenas 1 componente conexa. O tempo aumenta muito também, por conta de O(V+A) o tempo praticamente dobra. Outro ponto é que numa probabilidade pequena, aumentando um pouco apenas (0,01 para 0,05), percebemos a diminuição brusta de 46 componentes conexas para apenas 2. Gerei um random com probabilidade 0,06 e outro com 0,07. O 0,06 manteve 2 componentes conexas. O 0,07 mudou pra 1 componente conexa. Isso mostra exatamente a propriedade de 6 graus de separação. Exatamente de 0,01 → 0,07 vamos de 46 para apenas 1 componente conexa.

2.2) Grafos de palavras - Word Ladders

Para gerar os grafos de word ladders, usei a classe WordLadder.java do alg4 do Sedgewick and Wayne com algumas pequenas modificações. Fiz ela usar uma lista de palavras e com isso fazer todas as conexões entre as palavras com apenas 1 letra de diferença. Fiz a saída no formato necessário para entrada do Ep3. Primeiro fiz uma saída com os arcos das palavras, para ficar visível as conexões de fato e as word ladders que apareciam. Depois alternei para a saída apenas dos indexes das palavras, para poder usar no Ep3, de fato, esses arcos. Tanto o arquivo com a lista de palavras, a saída de arcos com as palavras e a saída de arcos com os indexes estão anexados ao ep.

Para executar o programa entre na pasta WL e rode:
javac WordLadder.java
java WordLadder words100.txt > EntradaWL100.txt (o número aqui pode ser 50, 100, 200 ou 500)

Com essas entradas prontas fiz os testes, sempre se baseando numa listagem de palavras com 5 letras em inglês:

- Lista 50 palavras:

  Vértices - 50 Arestas - 224
  24 componentes conexas: 1 de 8 elementos, 1 de 5 elementos, 1 de 4 elementos, 4 de 3 elementos, 4 de 2 elementos e 13 de 1 elemento.

  Tempos de execução:
  0m1,192s / 0m1,196s / 0m1,127s / 0m1,139s / 0m1,156s / 0m1,200s / 0m1,129s / 0m1,123s / 0m1,123s / 0m1,156s → média: 0m1,154s

- Lista 100 palavras:

  Vértices - 100 Arestas – 3140
  40 componentes conexas: 1 de 24 elementos, 1 de 14 elementos, 1 de 13 elementos, 1 de 4 elementos, 1 de 3 elementos, 7 de 2 elementos, 28 de 1 elemento.

  Tempos de execução:
  0m2,011s / 0m2,280s / 0m2,048s / 0m2,187s / 0m2,250s / 0m2,157s / 0m2,352s / 0m2,121s / 0m2,293s / 0m2,090s → média: 0m2,178s

- Lista 200 palavras:

  Vértices - 200 Arestas – 3756
  118 componentes conexas: 1 de 21 elementos, 1 de 20 elementos, 1 de 11 elementos, 1 de 7 elementos, 1 de 5 elementos, 1 de 4 elementos, 3 de 3 elementos, 14 de 2 elementos e 95 de 1 elemento.

  Tempos de execução:
  0m2,532s / 0m2,694s / 0m2,797s / 0m2,472s / 0m2,468s / 0m2,596s / 0m2,469s / 0m2,464s / 0m2,748s / 0m2,419s → média: 0m2,565s

- Lista 500 palavras:

  Vértices - 500 Arestas - 141958
  216 componentes conexas: 1 de 115 elementos, 1 de 20 elementos, 1 de 13 elementos, 1 de 9 elementos, 2 de 8 elementos, 2 de 7 elementos, 5 de 6 elementos, 1 de 5 elementos, 8 de 4 elementos, 13 de 3 elementos, 26 de 2 elementos e 155 de 1 elemento.

  Tempos de execução:
  0m17,929s / 0m18,418s / 0m17,674s / 0m17,982s / 0m17,977s / 0m17,864s / 0m18,087s / 0m18,146s / 0m18,260s / 0m18,608s → média: 0m18,094s

Nas word ladders, começam a aparecer algumas componentes gigantes conforme a listagem vai aumentando. Em 200 palavras a maior componente era de 21 elementos. Em 500 isso deu um salto para uma componente conexa de 115 elementos. Então a tendência é das componentes gigantes irem aparecendo conforme os vértices de entrada aumentam. Por conta dessas componentes gigantes o tempo também sofre um impacto. Também de 200 para 500 vinha mantendo algo não longe dos 2,5s e salta para 18s. Isso por conta de O(V+A), por aparecerem muitas arestas da componente gigante isso fica muito evidente no tempo.

Sobre os 6 graus de separação, podemos ver que está bem esparço. As componentes estão longe de se tornar uma só. Isso porque o grau de separação aqui é bem distante também. Não tem como associar vários graus na listagem, por não ser uma listagem com palavras controladas com densidade de ocorrência, então o grau é muito baixo. Por isso acaba evidenciando esses graus de distância da propriedade.