Relatório EP3ED2

Patrícia Rodrigues

Junho 2022

1 Como o código funciona?

NO menu inicial, temos

1 - Testar funcionalidades dos grafos (ler grafo, calcular distancia, ver componentes conexas, verificar se há ciclos, gerar grafos aleatórios de probabilidade p, ler grafos de palavras) 2 - testar experimentos com grafos legais

Ao escolher a opção 1, ele pedirá que você informe o número de vertices e arestas do grafo que quer gerar.

A seguir, você pode: em 1 construir suas próprias conexões para testar as funcionalidades;

em 5 gerar conexões aleatórias de probabilidade p para depois testar novas conexões;

em 6 gerar grafo de palavras (Obs importante: Cada palavra tem seu indice. A primeira palavra é 0 a segunda é 1, a terceira é 2 e assim por diante. Caso queira por exemplo pesquisar a distância da primeira palavra 0.banana digite 0 na pesquisa(opção 2).

2 Funções Implementadas

função Grafo(int V);

A função Grafao(int v) é construtora e inicializa os valores da matriz de adjacência com zeros, o vetor de visitados com false e o vetor de parentesco com -1, o vetor de componentes conexas (cc) com -

1 e o vetor de distancia com -1 também para todos os elementos.

função void Grafo::dfsR(int v)

A função dfsR foi implementada e faz uma busca em profundidade de maneira recursiva na matriz de adjacência. Durante a execução da dfsR os valores dos pais são registrados e os vertices visitados também.

função void Grafo::bfsR(int v)

A função bfsR(int v) faz uma busca em largura de maneira recursiva na matriz de adjacência. Durante a execução da dfsR, os valores de parentesco são registrados e os vertices visitados também. Além disso, os valores de distância a partir do primeiro vertice onde a busca é feita até os outros vértices também são registrados.

Matriz de adjacencia

a matriz de adjacencia trabalha com os índices. Os indices das linhas representam o vértice do qual se "olha" os outros vértices que estariam nos indices das colunas. Quando uma aresta é estabelecida de um vertice u1 até um vertice u2, o valor 1 é registrado.

Exemplo de grafo e representação de matriz de adjacência

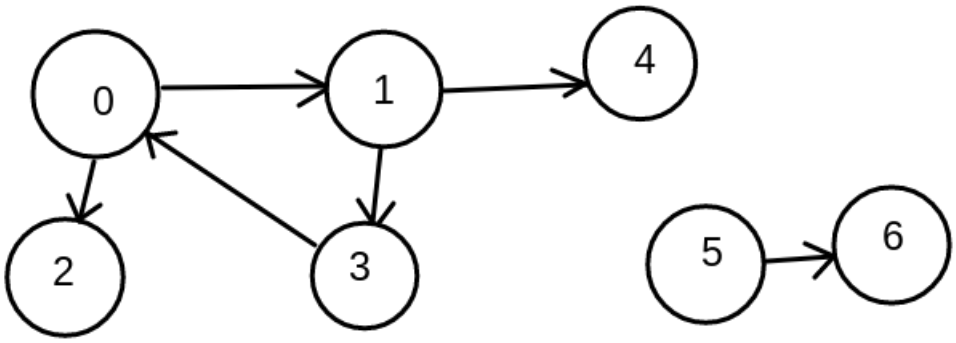


Figure 1: Grafo1

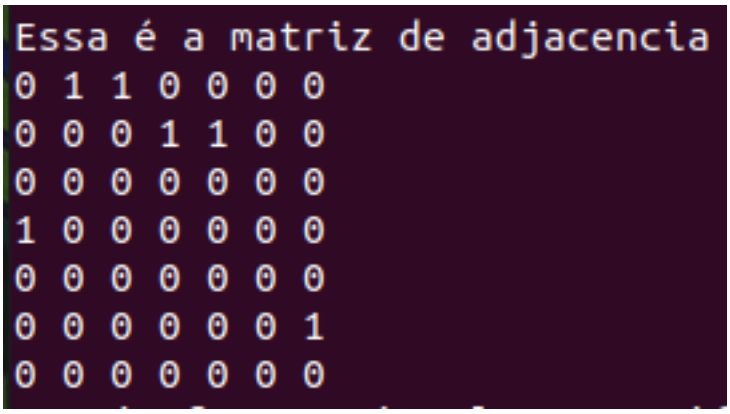


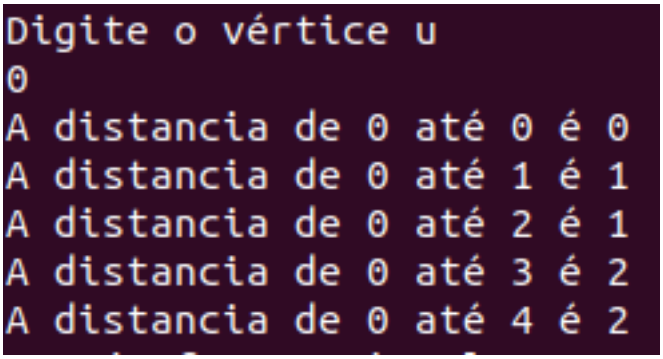
Figure 2: Matriz de adjacência 1

void Grafo::calculaDist(int uI)

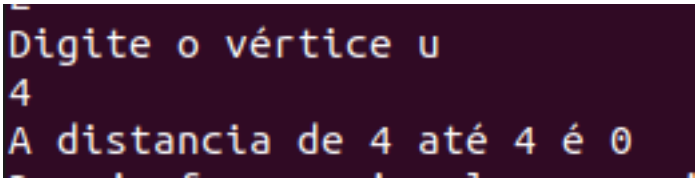
Dado um vértice u calcula a distância de u até todos os vértices de G

Para calcular isso, usamos a função calculaDist(int uI) que roda a bfsR no vertice u e ela irá registrando o valor da distancia de todos os vértices alcançados. O vetor de distância irá registrar nos indices a distancia de cada valor que esse indice teria se fosse um vértice que u alcança.

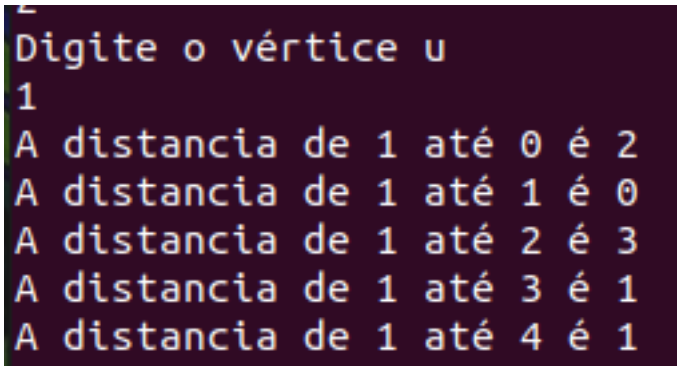
Exemplo: Para o grafo mostrado anteriormente (grafo 1), temos: **A distância do vértice 0 para todos os outros valores**



A distância do vértice 4 para todos os outros valores



A distância do vértice 1 para todos os outros valores



void Grafo::identificaCompCon() Dado um grafo G determina o número de componentes conexas e o tamanho de cada compo- nente.

A função identificaCompCon() verificará quantas componentes conexas existem no grafo. Para isso, rodamos a dfsR no vértice 0 e depois marcamos no vetor cc um mesmo numero para todos os valores que foram visitados. A seguir, rodamos a dfsR para todos os valores que foram visitados quando rodei no primeiro vertice e marcamos de mesma forma todos que ele atingir com o mesmo numero. Depois disso, rodamos da mesma forma que rodamos para o zero para todos os outros valores . No final quem fizar parte de uma mesma componente conexa terão o mesmo valor. A partir dai é possível identificar quantas componentes conexas têm e qual é o tamanho delas.

Componentes conexas do grafo 1

```
A quantidade de componentes conexas é 2
A quantidade de elementos da 1° componente é: 5
A quantidade de elementos da 2° componente é: 2
```

Grafos aleatórios de Erdös e Rényi

Os matemáticos Erdös e Rényi propuseram um modelo de grafos aleatórios, em que cada aresta possıvel que conecta pares de vértices está presente no grafo com probabilidade p. Para simularmos esse grafo, a função void Grafo::probabilidade(int p10) iremos simular uma urna, na qual sempre 100 valores entre 1 e 100 serão sorteados b1, b2, b3, ..., b100. A probabilidade p será multiplicada por cem e um sorteiro será feito nessa urna cada vez que uma possível aresta estiver prestes a ser estabelecida entre um vertice u1 e u2. Se a bola retirada b for menor ou igual ao valor p * 100, uma aresta será estabelecida no grafo.

3 Grafos Legais e Testes

Quais são os teste?

1. Componente Gigante:
2. Seis graus de separação
3. Como o grau de conextividade se comporta para grafos de diferentes tamanhos desde redes mais esparsas até redes mais densas

Teste hipótese 1: Componente Gigante

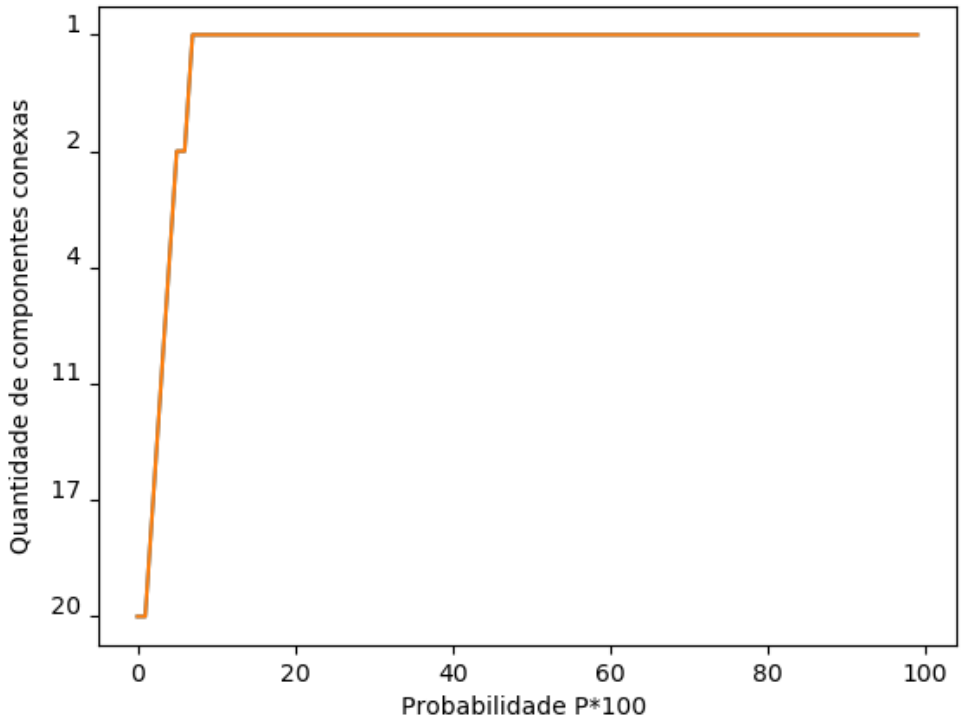
Os matemáticos Erdös e Rényi propuseram um

modelo de grafos aleatórios, em que cada aresta possıvel que conecta pares de vértices está presente no grafo com probabilidade p. Eles mostram que, se $p \leq \left(\frac{1-e}{n}\right)$, então, com alta probabilidade as componentes conexas são pequenas, com O(log n) elementos. Mas, $p \geq \left(\frac{1+e}{n}\right)$ surge uma componente gigante no greafo.

Adotando e = 1/n = 0.05, e um grafo G de n = 20, temos que:

1. $p \leq \left(\frac{1-0.05}{20}\right) \rightarrow$p ≤ 0.0475
2. $p \geq \left(\frac{1-0.05}{20}\right) \rightarrow$p ≥ 0.0525

Então para testar isso gerei um código que gera uma componente conexa de tamanho n = 20 (numero de vértices) e nessa componente geramos conexões de probabilidade p iterativamente e computamos os valores da quantidade de componentes conexas para aquele p. Começando em p = 0.00 e indo até p = 1.00 (2 casas decimais) e a cada grafo G(20,p) a quantidade de componente conexas foi contabilizada e depois de tudo o código gera um arqCompCom.txt com esses dados.



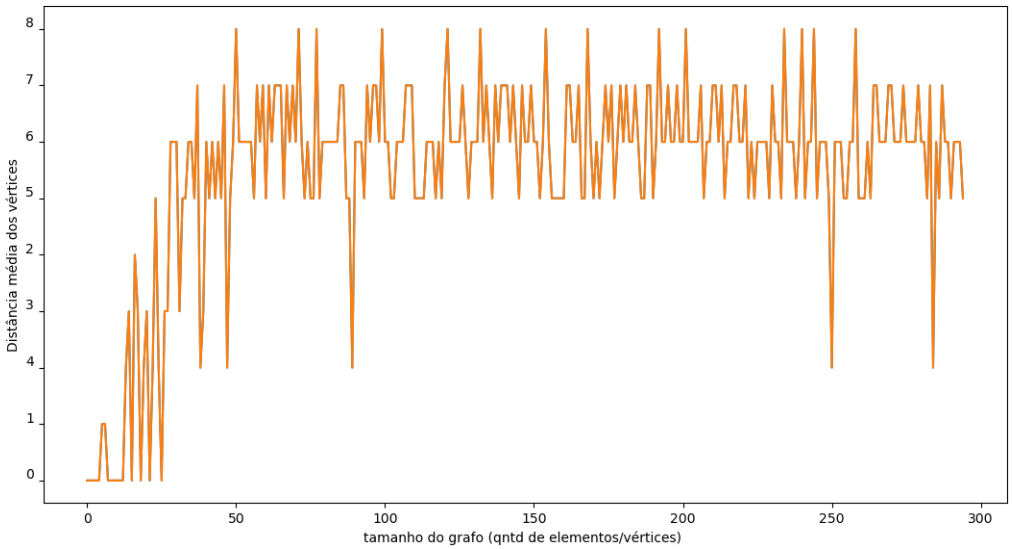
O gráfico acima representa a quantidade de componentes conexas em (y) em relação a probabilidade p multiplicada por 100 (para ser mais fácil de visualizar) em (x).

Podemos observar quando $p \leq 0.0475$ a quantidade de componentes conexas vai caindo no gráfico, pois seu valor vai aumentando até chegar em 20 (para p = 0). Ou seja, a quantidade de componentes conexas vai aumentando até chegar em uma situação na qual nenhum vértice é conectado com

nenhum outro. Por outro lado, para $p \geq 0.0525$ vemos que a quantidade de componentes conexas vai caindo até 1 e depois de chegar em um para todos os outros valores de p ela vai ficando cada vez mais conexa e densa para no final (p = 1) se tornar uma componente gigante super conexa e densa, no qual todos ous vérteces se conectam com todos os outros. Nessa situção a distância dos vértices de um ao outro é 1 para todos os elementos. Quanto maior a quantidade de componente conexas, menores elas provavelmente são são, pois elas tm que se dividir mais. Em contra partida, quanto menor a quantidade de componente conexas, menores elas são. Portanto, pelo gráfico, podemos concluir que se inicia para p = 0 20 componentes conexas e termina para p = 1 com 1 componente conexa.

Teste hipótese 2: Seis Graus de Separação

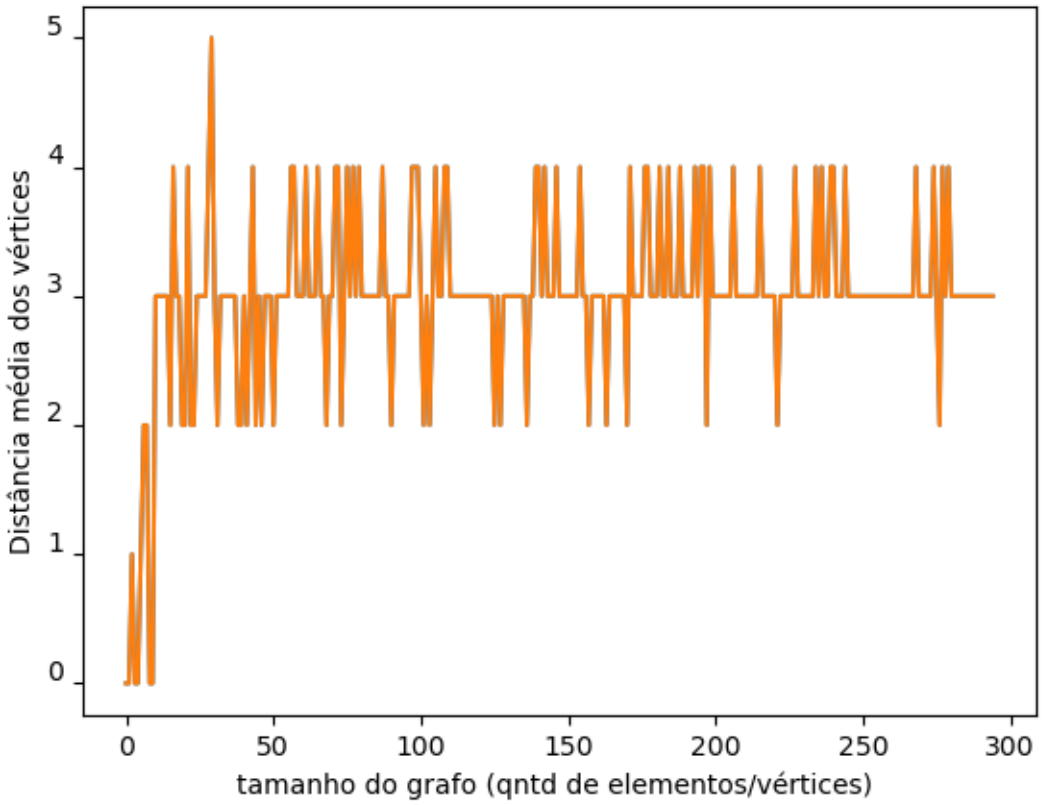
Este conceito surgiu em 1929, devido a F. Karinthy. A ideia é que todas as pessoas estão a seis ou menos conexões sociais de qualquer outra, ou seja, uma cadeia de "amigos de amigos" pode ser feita para conectar qualquer par de pessoas. Para testar essa ideia, gerarei grafos que vão de n = 5 até n = 300 e a cada iteração conputamos o valor da distância média dos vértices. O gráico a seguir repesenta em x a distância média do grafo de tamanho y.



Podemos verificar que a partir de um tamanho n do grafo (por volta de 23 vértices) a distância

média fica ao redor de 6 graus de separação.

Agora testaremos para p = 0.2. Veremos se esse graus de separação mudará.



Verificamos que ao dobrar a probabilidade (de 0.1 para 0.2) a distância média cais pela metade. Isso aconteceu pois a quantidade de vértices estabelecidos aumentou, logo a probabiidade de haver um caminho menor de u a v aumentou proporcionalmente e naturalmente a média dessas distâncias também. Quanto a ideia de que cada pessoa estariam separadas por 6 graus de separação, neste teste verificamos como se cada pessoa tivesse a probabilidade 1/10 de conhecerem umas as outras. Em ambientes maiores com grafos ainda maiores que 300 pessoas essa probabilidade pode ser ainda menor. Nesse relatório conseguimos verificar que com uma probabilidade 0.1 de redes de aproximadamente 10 pessoas até 300 a distância média é 6. No meu código infelizmente quando tento colocar tamanhos maiores do que 300 o sistema morre (Morto) e não consigo fazer testes gerando iterativamente grafos para tamanhos maiores de 300.

Teste hipótese 3: Como o grau de conextividade de comporta de redes mais esparsas para redes mais densas em diferentes tamanhos

Verificaremos se, quanto maior for p (probabilidade de haver conexão entre os vértices u1 e u2,

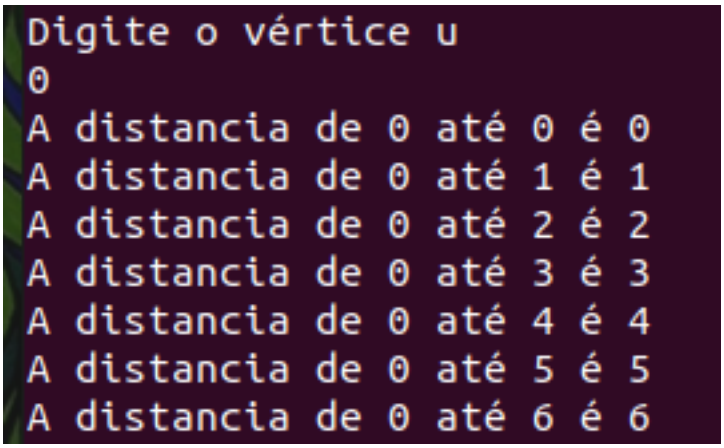
menor será a distância média entre um vértice qualquer u pertencente ao grafo e todos os outros vértices existentes ali. Ademais, também atestaremos o tópico 2 de seis graus de separação, pois v. Verificaremos como a probabilidade de conexão desses vértices influencia essa média de separação (ou seja, como essa distância varia de redes mais esparsas (menor probabilidade de conexão) para redes mais densas (maior probabilidade de conexão).

Para p = 0.1 vimos que a disntância média gira ao redor de 6 e que para p = 0.2 a distância média gira ao redor de 3 (cai pela metade). O que é interesante verificar aqui é que essa distância se estabilida ao redor de um número mesmo que o tamanho do grafo cresça bastante.

4 Grafo de palavras

Testes:

Para o teste do enunciado:
tears – sears – stars – stare – stale – stile – smile
tivemos o seguinte resultado



0 = tears 1 = sears 2 = stars 3 = stare 4 = stale 5 = stile 6 = smile

Podemos ver que a distância de tears para smile é 6 e que elas estão ligadas por 7 palavras (de 0 a 6)