Relatório trabalho 2

Hill, The Climber

g203(Carlos Palma_46520 e Manuel Cunha_48482)

- ## Algoritmo

1. Pedido o input com recurso a BufferedReader.
2. Com o auxilio de um array de vértices são criados os respetivos vértices no grafo e guardadas as coordenadas dos mesmos(A cada índice corresponde um objeto).
3. As coordenadas dos vértices são ordenadas em relação à coordenada y da menor para maior.
4. É verificado o maior alcance que o utilizador inseriu.
5. Com recurso a dois ciclos for, verificamos se a distancia entre o vértice i e os seguintes(e assim sucessivamente) está entre os valores do alcance máximo, caso seja verdade, calculamos a distancia entre os vértices, caso a mesma seja menor ou igual ao alcance máximo é criada uma aresta bidirecional entre os vértices com o valor da distancia entre os mesmos associado(o ciclo termina quando as distancias entre o vértice i e o vértice j é maior que o alcance máximo e nesse caso passamos para o vértice i+1 e verificamos os vértices seguintes ao mesmo).
6. Após termos o grafo terminado realizamos um percurso em largura.
7. No percurso em largura todos os vértices que podem ser iniciais são inicializados com a cor cinza, a distancia 1 e adicionados à queue.
8. A diferença neste percurso em largura em relação aos que tratamos anteriormente consiste em que apenas somamos 1 no array de distancias caso alem do vértice ainda não ter sido visitado(branco) o valor da distancia entre os vértices seja menor ou igual ao alcance.
9. Por fim verificamos quais os vértices que podem ser finais e vamos guardando numa variável aquela que é a distancia mínima entre um vértice inicial e final.

10. Por fim a nível do output verificamos se o alcance é maior que a altura em caso positivo damos como output 0, caso negativo verificamos se a distancia é igual a Integer.MAX_VALUE(valor retornado pela pesquisa em largura quando não é possível alcançar os nós finais a partir dos iniciais) e damos como output “unreachable”, por ultimo se nenhum destes se verificar retornamos a menor distancia entre os vértices iniciais e finais.

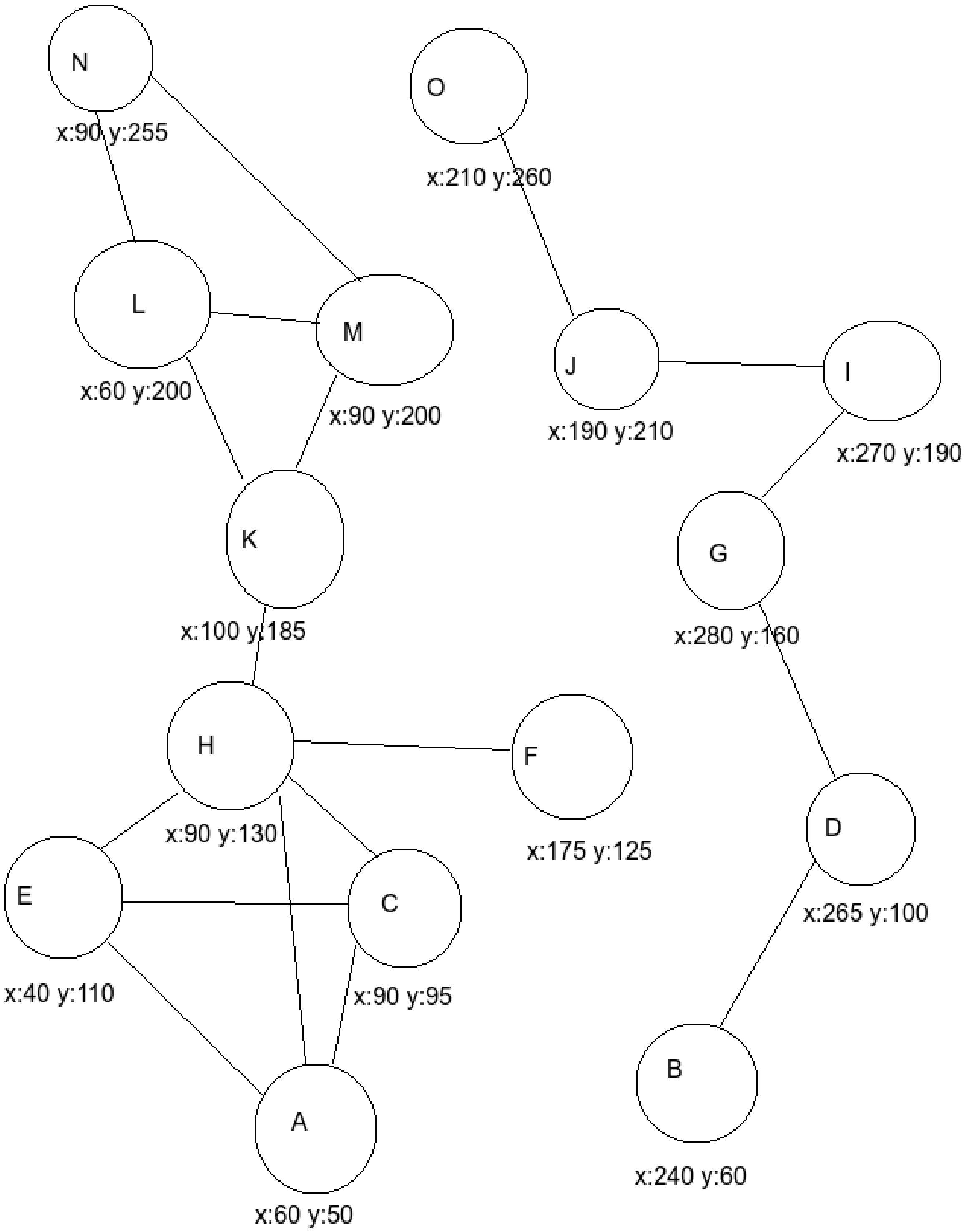
- ## Grafo construido

Foi criado um grafo não orientado que usa um HashMap e uma LinkedList para guardar os vértices adjacentes, cada vértice tem um objeto referente onde são guardadas as coordenas x e y e cada aresta tem um objeto referente onde é guardado o vértice de destino e o valor da aresta.

Na verdade o nosso grafo pode até ser considerado um grafo orientado pois o que fazemos é adicionar uma aresta entre vértices tanto do vértice de partida para o vértice de chegada como vice-versa.

As arestas apenas serão adicionadas caso o valor da distancia entre os vértices tratados seja inferior ou igual ao alcance máximo pretendido pelo utilizador.

Para melhor exemplificação está ilustrado na figura seguinte qual seria o grafo obtido tendo em conta o exemplo apresentado no enunciado, para melhor perceção não estão presentes dois membros do nosso grafo, o valor das arestas e duas arestas orientadas uma em cada sentido, ao invés das arestas não orientadas apresentadas.


N
x:90 y:255
O
x:210 y:260
L
M
x:60 y:200
x:90 y:200
J
x:190 y:210
I
x:270 y:190
K
x:100 y:185
G
x:280 y:160
H
F
x:90 y:130
x:175 y:125
D
x:265 y:100
E
C
x:40 y:110
x:90 y:95
B
A
x:240 y:60
x:60 y:50

- **Análise da complexidade**

1. Na criação dos vértices utilizamos um ciclo for que termina quando atingimos o numero máximo de vértices pretendidos logo a complexidade temporal será $\Theta(V)$ tal como a complexidade espacial,sendo V o numero de vértices que se pretende adicionar ao grafo.
2. Na criação das arestas utilizamos dois ciclos for que irão percorrer os vértices, logo no pior dos casos terá complexidade temporal $O(V^2)$, sendo V o numero de vértices, no entanto dependendo dos valores das coordenadas y dos vértices a complexidade poderá ser bastante mais reduzida pois o segundo ciclo for apenas correrá enquanto a distancia entre os valores das coordenadas y dos vértices for menor ou igual ao alcance. A complexidade espacial será $\Theta(2E)$ sendo E o numero de arestas pois teremos que guardar as arestas 2 vezes, tanto para uma direção como para outra.
3. No percurso em largura iremos ter uma complexidade espacial no pior caso de $O(V+2E)$ pois teremos de percorrer todos os vértices(V) e duas vezes as arestas(2E) pois trata-se de um grafo não orientado. A nível de complexidade espacial teremos $\Theta(V+2E)$ pois teremos de percorrer todos os vértices e as arestas duplicadas.

Como tal podemos concluir que a complexidade temporal do nosso programa será $O(V+2E)$ e que a complexidade espacial será $\Theta(V+2E)$.

Notas:

-O nosso grafo tem como base um grafo genérico apresentado no site Geeksforgeeks(https://www.geeksforgeeks.org/implementing-generic-graph-in-java/) implementado e alterado por nós tendo em conta as necessidades apresentadas no problema em questão.

-O percurso em largura utilizado foi o mesmo que utilizámos tanto nas aulas práticas como teóricas mas apresentando as alterações necessárias para o problema em questão.