Projeto Xablau

Douglas Martins [1] **Ivens Diego Müller**[2] **Tiago Funk**[3]

[1]CEAVI - Universidade do Estado de Santa Catarina(UDESC) Ibirama - SC - Brasil

ivens.muller@edu.udesc.br, martins.douglas.souza@gmail.br, tiagoff.tf@gmail.com

Informações Gerais

O Projeto Xablau tem como objetivo, a partir de um grafo informado, no qual o grafo deve contar a identificação das arestas, vértices e indivíduos que estarão posicionados em cima de um vértice, definir o ponto de encontro onde o custo total de movimentação dos indivíduos é o mínimo possível.

Para isso, o usuário que utilizará o software deve, por meio de um documento de texto com um padrão pré definido, definir os dados referente ao grafo, onde os dados devem ser:

- Vértices: contendo um rótulo, um valor para x e um valor para y.
- Arestas: contendo um vértice de origem, um vértice de destino, comprimento, se a aresta é bidirecional e um nome.
- Indivíduos: contendo o vértice de localização e um nome.

Com estes dados, a partir do algoritmo de Floyd Warshall, o sistema identificará qual será o vértice de destino de todos os indivíduos. A partir deste vértice, o sistema utilizará o algoritmo de Dijkstra para definir o caminho que cada indivíduo utilizará para chegar ao ponto de encontro.

O passo a passo para execução do sistema é:

- Criar um novo mapa: para isso, o usuário que estiver na tela inicial clicará em Criar Mapa no menu de opções.
- Carregar um mapa pelo arquivo de texto: para isso, após o mapa criado, o usuário clicará no menu de opções e em carregar mapa, escolherá um grafo a partir de um arquivo de textos escolhido, após o mapa definido será desenhado na tela. Sendo as arestas cinzas, os vértices azuis e os indivíduos vermelhos.
- Calcular o ponto de encontro: o usuário clicará em Calcular Ponto de Encontro, e após o cálculo, o sistema informará na tela qual o ponto de encontro e, caso não tenha nenhum indivíduo nesse vértice, o vértice será pintado de laranja.
- Iniciar encontro: o usuário clicará em Iniciar Encontro e, neste momento, o sistema definirá o caminho que cada indivíduo percorrerá e irá mostrar os indivíduos caminhando até o ponto de encontro.
- Limpar o mapa: o usuário clicará em Limpar Mapa e, neste momento, o mapa será limpo, permitindo carregar outro mapa na mesma tela. Porém, caso o usuário queira ter dois mapas abertos ao mesmo tempo, basta o mesmo clicar no menu opções da página inicial e depois em Novo Mapa, assim estarão abertos duas telas com mapas completamente distintos.

Dijkstra

Através do algoritmo de Dijkstra, podemos realizar o cálculo do caminho mais curto entre dois pontos. Para isso, utilizaremos Dijkstra em conjunto com o nosso grafo, que é definido por um arquivo texto. O Algoritmo de Dijkstra, a partir de um ponto de origem, que no nosso caso é um vértice, realiza o cálculo do caminho que contém o menor custo do ponto de origem para todos os outros vértices do grafo.

O algoritmo de Floyd Warshall, explicado na próxima seção, é utilizado para definirmos qual o ponto de encontro entre todos os vértices do grafo, e baseando-se nesse ponto de encontro como o destino de todos os indivíduos, aplicamos o algoritmo de Dijkstra para definir o caminho que contém o menor custo, que nesse caso seria a menor distância, entre o vértice onde o indivíduo está posicionado, e o vértice de destino, que é o ponto de encontro.

"Este Algoritmo parte de uma estimativa inicial para a distância mínima, que é considerada infinita, e vai sucessivamente ajustando-a"[**?**]. No início da execução do algoritmo, partimos de que todos os vértices estão a uma distância infinita do vértice de posicionamento do indivíduo e, o vértice onde o indivíduo está posicionado, contém uma distância zero. Após isso, verifica-se os vértices laterais que contém a menor distância, ou seja, os vértices ligados ao do indivíduo e salva-se essa distância. A partir desse momento, escolhe-se o vértice de menor distância para ser o posicionado, e calcula-se a sua distância para os vértices adjacentes, sempre somando a distância do vértice anterior. É feito esta iteração até que todos os vértices tenham sido verificados, ou, no nosso caso, até o vértice de destino.

Foi realizado uma alteração, para que no final da execução do algoritmo, ele retorne uma lista encadeada de arestas, que será o caminho por onde o indivíduo deverá percorrer para chegar até o ponto de encontro.

Floyd Warshall

o algoritmo de Floyd-Warshall é um algoritmo que resolve o problema de calcular o caminho mais curto entre todos os pares de vértices em um grafo orientado (com direção) e valorado (com peso). O algoritmo de Floyd-Warshall recebe como entrada uma matriz de adjacência que representa um grafo orientado e valorado. O valor de um caminho entre dois vértices é a soma dos valores de todas as arestas ao longo desse caminho. As arestas do grafo podem ter valores negativos, mas o grafo não pode conter nenhum ciclo de valor negativo. O algoritmo calcula, para cada par de vértices, o menor de todos os caminhos entre os vértices. Por exemplo, e no nosso caso, o caminho de menor custo.

Utilizamos esse algoritmo em nosso trabalho para calcular o ponto de encontro que minimiza a distância que individuos deveriam percorrer pelo grafo para que todos devessem se encontrar em um mesmo ponto.

Na forma como implementamos ele em nosso trabalho, ao criar a matriz de adjacência, consideramos que vértices adjacentes tem com valor na matriz o peso da aresta

entre esses vértices, não sendo adjacentes, colocamos o valor de infinito na matriz, como não existe o valor de infinito em java, então foi inserido o maior valor possível de double na matriz.

Com essa matriz é realizado o cálculo do algoritmo de floyd warshall, a matriz resultante possue os valores em cada célula que representam a melhor distância entre dois vértices, para achar o vértice que é o resultado final, basta achar a linha da matriz que possue a melhor soma de valores.

Cálculos matemáticos para desenho do arco em tela

No trabalho foi necessário implementar um código para desenhar uma representação do grafo e do individuos que irião se movimentar pelo grafo para mostrar o seu andamento pelo grafo.

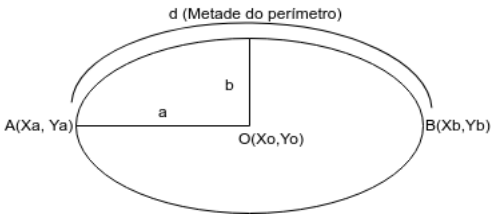

Figura 1. Elipse e suas propriedades

Para desenhar as arestas foi levado em consideração o seguinte raciocínio: toda aresta é considerada como uma semi-elipse, ou seja, apenas como se fosse a metade de uma elipse, aonde uma reta é um caso que o valor de b é igual a zero (ver fígura 1).

Primeiro é preciso observar que ao inicio teriamos apenas os pontos A e B e o valor do arco entre eles (são esses valores que o usuário vai informar), se por acaso esse valor do arco for maior que a distância em linha reta entre os pontos, será desenhado um arco, e se for igual é desenhada uma linha reta.

Para iniciar termos a equação da elipse paralela ao eixo das abcissas.

$$\frac{x^2}{a} + \frac{y^2}{b} = 1 \tag{1}$$

Temos a equação do perímetro de elipse, a equação que sera apresentada é uma que possui aproximação do cálculo com erro de 5%.

$$P = 2\pi\sqrt{\frac{a^2 + b^2}{2}} \tag{2}$$

Mas como a elipse é possui a caracteristica de ser simétrica, o ponto O é tem as cordenadas dadas pelo ponto médio entre A e B:

$$O(\frac{x_a + x_b}{2}, \frac{y_a + y_b}{2}) \tag{3}$$

Assim descobrimos que o valor de a é a distância entre o ponto A entre o ponto O.

$$a = \sqrt{(x_o - x_a)^2 + (y_o + y_a)^2} \tag{4}$$

Agora, levando em conta a equação 4, podemos calcular o valor de b com a equação 2. Lembrando que o valor do Perímetro é igual a duas vezes o valor do arco que o usuário deverá informar.

$$P = 2d \tag{5}$$

A equação 2 fica:

$$2d = 2\pi\sqrt{\frac{a^2 + b^2}{2}} \tag{6}$$

Quando vamos isolar o b na equação 6 temos:

$$b = \sqrt{\frac{2d^2}{\pi^2} - a^2} \tag{7}$$

Para facilitar na hora de cálcular os pontos, vamos desenhar a elipse na origem do plano cartesiano e vamos realizar uma rotação para deixa-la na inclinação correta e somar as coordenadas originais do centro da elipse para inseri-las no local correto.

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \times \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} x_o \\ y_o \end{pmatrix} \tag{8}$$

Lembrando que o valor de do ângulo teta é dado por:

$$\theta = \arctan\left(\frac{y_b - y_a}{x_b - x_a}\right) \tag{9}$$

Que deriva de do cálculo do coeficiente angular (ver equação 10) da reta (no nosso caso não necessáriamente uma reta, mas podemos apenas considerar para os pontos A e B para achar a inclinação do arco).

$$\tan\theta = \frac{y_b - y_a}{x_b - x_a} \tag{10}$$

Para cálcular os pontos no software, iremos ultilizar um laço de repetição para dar os valores de x para no intervalo de -a e a (lembre que estamos cálculando a elipse na origem e depois será feita uma rotação para coloca-la no local correto) e apenas precisamos cálcular o valor de y da coordenada. Isolando y na equação 1 temos:

$$y = \sqrt{b^2\left(1 - \frac{x^2}{a^2}\right)} \tag{11}$$

Com essas equações podemos cálcular os pontos do arco que deve ser desenhado em tela, e os pontos que indivíduos devem percorrer pelo grafo.

Operador de Arquivos

O Algoritmo que executa a leitura e escrita do arquivo texto é bastante simples e de fácil manutenção. Abaixo estará dois sub-tópicos que descreverão, cada um, o método escrever() (responsável por armazenar os dados no arquivo texto), seguido do método ler() (método responsável por ler os dados do arquivos texto e utilizá-los no código).

Método escrever

Este método não possui segredo, ele recebe uma lista de desenháveis via parâmetro, bem como o nome do arquivo que irá ser registrado as informações vindas de código. Apenas abre-se uma instrução try-catch() para poder realizar a escrita no arquivo. Dentro do try, instância-se a classe FileWriter que se encarregará de escrever os dados no arquivo e logo abaixo percorre-se a lista de desenháveis, onde para cada posição que há na lista, registra-se uma linha dentro do arquivo texto. O método retorna true caso tudo ocorra como o planejado e false caso ocorra algum error neste processo. **Obs: Cada valor posto no arquivo, no que se refere ao comprimento da aresta, equivale a 1 metro, logo, se o usuário colocar uma medida 5 como comprimento da aresta, a mesma será transcrita como 5 metros dentro do mapa.**

Método ler

Este método é um pouco mais complexo do que o método escrever() visto anteriormente. Este método não recebe parâmetro porém possui o retorno de uma lista de Desenhavel(eis). Inicia-se com a criação de duas listas, uma de desenháveis e outra de vértices, logo abaixo encontra-se a instrução try-catch(), que se escarregadará de tratar possíveis erros que possam vir a ocorrer em meio ao processo. Dentro do try, instancia-se a classe BufferedReader que se encarregará de ler os dados no arquivo texto. Um laço de repetição while() é necessário para a leitura de cada linha do arquivo texto, logo, quando chegar na última linha do arquivo, o laço de condição estourará e a execução do método termina retornando a lista de desenhaveis que o mesmo populou ao longo do processo. Dentro do laço while() não há segredo, o algoritmo trata as possíveis imperfeições que usuários desavisados possam vir a alterar. Para reconhecer quem é vertices, aresta e indivíduo dentro do arquivo, o algoritmo pega a primeira letra de cada linha e verifica se é uma letra ?v? (de vértice), uma letra ?a? (de aresta) ou uma letra ?i? (de individuo). Ao localizá-los, ele realiza as verificações necessárias e adiciona o desenhavel encontrado detro da lista de desenhaveis que será retornada pelo método. **Obs: Cada valor posto no arquivo, no que se refere ao comprimento da aresta, equivale a 1 metro, logo, se o usuário colocar uma medida 5 como comprimento da aresta, a mesma será transcrita como 5 metros dentro do mapa.**