RELATÓRIO DO TRABALHO DE PROBLEMA DE TRANSPORTE

DIFICULDADES:

1. Compreender que no código de ler arquivo, fornecido pelo professor, eu poderia usar recursividade ou não;
2. Fazer com que todos os valores não repassados pelos arquivos de criminalidade.txt, distância.txt e transito.txt preencham as suas respectivas matrizes sem ter lixo de memória e para que as rotas sem caminho ficassem com o valor de -1;
3. Comunicação e compreensão entre os raciocínios da dupla;
4. Entender, a partir das instruções fornecidas do exercício, se seria dado um ponto de partida e de chegada , e assim achar a rota de menor custo, ou se deveríamos achar a rota de menor custo que passa em todos os pontos;
5. Calcular, levando em consideração os pesos dados a criminalidade, distância e trânsito, a rota de menor custo;
   1. Na função “menor\_possivel()”, tive dificuldade de entender que a variável “int menor\_valor” deveria inicializar com o maior valor possível do tipo da variável int, pois não poderia iniciar com o primeiro valor da linha, já que eu não sabia se ele estava no vetor de rota e caso ele já estivesse e fosse o menor valor possível o seu índice iria para o vetor rota e iria ficar errado, visto que ele já teria aparecido;
   2. Na função”int menor\_possivel()” preciso verificar se o valor já apareceu no vetor de rota ou não, sendo assim, a função “existe\_vetor()” irá verificar isso.Graças a aula de quinta-feira(23/11/2023) em que estudamos na prática a recursividade foi possível criar o raciocínio da função int “existe\_vetor()”, tendo em vista que não foi usado recursividade propriamente dita, em que retorna 1 se o valor já apareceu no vetor e 0 se não.
   3. Como inicializamos a matriz com todos os valores sendo -1, para não termos lixo de memória e para que as rotas sem caminho ficassem com o valor de -1, foi feita uma comparação na hora de inserir o valor ponderado, que é o resultado da ponderação entre a matriz de trânsito, de criminalidade e de distância, na matriz ponderada, que é uma matriz auxiliar para achar o valor ponderado.O problema encontrado foi que na hora da comparação feita os valores de trânsito criminalidade e distância deveriam ser maiores que -1, no entanto, essa comparação estava invertida.

MATERIAIS UTILIZADOS:

1. Vídeo para começarmos a montar o raciocínio

<https://www.youtube.com/watch?v=RGaGsUf_KC8>;

1. Explicação do algoritmo guloso: [https://www.ime.usp.br/~pf/analise\_de\_algoritmos/aulas/guloso.html#:~:text=Um%20algoritmo%20guloso%20é%20míope,em%20cada%20iteração%20são%20definitivas](https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/guloso.html#:~:text=Um%20algoritmo%20guloso%20%C3%A9%20m%C3%ADope,em%20cada%20itera%C3%A7%C3%A3o%20s%C3%A3o%20definitivas).
2. Conversamos sobre o trabalho entre amigos para podermos montar o raciocínio e o código dentro das especificações exigidas pelo professor;

JUSTIFICATIVA DA FÓRMULA:

matriz\_ponderada[i][j] = (transito[i][j] \* 2) + (criminalidade[i][j] \* criminalidade[i][j]) + (distancia[i][j] \* 2);

Fizemos uma breve pesquisa entre amigos sobre o que desmotiva a utilizar certo trajeto. Nesse estudo foi dito em grande maioria pelos nossos colegas que grandes distâncias desmotivam de maneira igual ao intenso trânsito, o que justifica ambas terem o mesmo peso. Já quando comparamos rotas de alta criminalidade com caminhos de elevados congestionamento e distância foi dito por colegas que dirigem que preferem utilizar rotas de maior distância e de intenso trânsito do que rotas com alta criminalidade.Com tudo, demos o mesmo peso para distância e trânsito por serem igualmente desfavoráveis e botamos a criminalidade ao quadrado por ser algo decisivo entre pegar uma rota ou não.

RACIOCÍNIO PARA CALCULAR A ROTA DE MENOR CUSTO

* Raciocínio base para achar a rota de menor custo foi o do algoritmo guloso. Primeiramente preenchemos a matriz ponderada, matriz na qual contém os valores de custo real de cada trajeto utilizando seus respectivos pesos. Em seguida pegamo “int início”,que contém o ponto de partida, e varremos a linha correspondente a esse “int inicio”, e a coluna que continha o menor valor seria o trajeto mais favorável.Com isso em mente, o valor da coluna que contém o trajeto com o menor custo indicará a próxima linha que irei varrer e assim sucessivamente até eu chegar no meu ponto final representado por “int final”. Nesse viés, a cada varredura de linha foi necessário checar se o trajeto já estava no vetor de rota, se sim pegamos o trajeto que não estava no vetor de rota e que tinha o menor valor como o trajeto de menor custo. Tudo isso foi feito em um loop for que irá rodar 20 vezes até chegarmos ao destino.
* Sequência de processos do código:
  + 1. Iniciar a todas as matrizes 20x20 com todos os valores -1;
  + 2. Ler arquivos .txt para obtermos os dados de distância, criminalidade e trânsito;
  + 3. Preencher a matriz ponderada com os valores de custo ponderados dos trajetos;
  + 4. Percorrer todos os pontos que podem assumir a posição inicial e posição final. A partir do ponto de início iremos varrer a linha da matriz ponderada, matriz onde contém os valores dos custos ponderados, na função “ void analisa\_menor()” achar o trajeto com o menor custo e inserir no vetor rota.Em seguida vamos pegar a coluna com o trajeto de menor custo e varrer a sua respectiva linha, sempre lembrando de evitar pontos do trajeto que já foram visitados ou que não possuem rota entre si, e armazenar no vetor rota e assim sucessivamente até acharmos o trajeto total de menor custo;
  + 5. Pegar o vetor rota e jogar na função de “int calcula custo()” para calcular o custo do trajeto e atribuir o retorno da função a variável “int valor” ,assim, podemos achar a rota de menor custo comparando os valores que serão retornados da função”int calcula\_custo()” com o menor custo de rota até então que é atribuído a variavel “int aux”.
  + 6. A cada comparação em que o custo da rota é menor que o antecessor, que estará atribuído a “int aux”, iremos armazenar o vetor de menor custo de rota com a função “void copia\_vetor()”.Essa função armazena o vetor de menor custo no “int vetor\_aux”.