Introdução

Para que possa haver uma solução, é necessário que se entenda o problema. O problema abordado por esse trabalho prático é que se deve determinar a ordem em que as cidades devem ser visitadas para que os caminhões da empresa percorram o menor caminho possível.

Isso porque a empresa deseja reduzir os gastos com gasolina. A partir disso um algoritmo é criado, fazendo o uso de grafos é possível chegar a uma solução para ele.

A solução utiliza a função permuta para fazer trocas em um vetor, e assim calcular os caminhos possíveis para a rota do caminhão, elas são feitas para definir o melhor caminho. A lógica passa pela recursividade que a partir dela os caminhos são gerados e comparados com caminhos já testados anteriormente, e ao fim teremos o melhor.

Solução do Problema

A solução foi concretizada a partir do uso de algumas funções.

Fazendo o uso de grafos, foi necessário definir duas estruturas uma chamada ‘vertice’ com dois inteiros e outra ‘precedencia’ também com dois inteiros. Sendo que esses valores são definidos pelo que seria o peso no vértice e a prioridade de precedência. Eles são definidos pelo .h e utilizados na função main e a partir deles são criados dois vetores e usados diretamente para receber a entrada do arquivo.

Nesse algoritmo em particular, o uso da função ‘main’ foi de grande importância sendo que nela acontece a inicialização. A função abre o arquivo de entrada e faz a leitura, em seguida todos os testes serão feitos na própria função como: existe uma chance de que não seja possível chegar na solução porque existem algumas restrições de precedência e assim será exibido na tela a palavra Deadlock; esse teste consiste na comparação do vetor de precedência, como ele é um tipo definido pelo programador, ele utiliza seus dois inteiros e faz comparações entre eles dentro do vetor, se esses valores

forem iguais, uma variável será acionada para que o programa imprima a palavra e finalize o programa com return 0. Entre estes testes feitos a também o cálculo da distância após a leitura do arquivo, o tipo vértice tem um vetor que recebe a entrada de dados, assim a distância é definida pela formula padrão raiz de $(x1 *x0)^2+ (y1 *y0)^2$, sendo assim a solução é diretamente definida por isso porque o caminhão da empresa deve percorrer as cidades determinadas pela ordem, pelo menor caminho possível. Após isso, a função permuta é chamada. Depois que a função permuta termina, será exibido na tela o valor da menor distância.

A função ‘permuta’ usando recursividade, trocando os elementos do vetor para fazer as permutações, essas permutações são os caminhos possíveis para os caminhões percorrerem, essas trocas ocorrem como uma busca, pois assim o menor caminho possível pode ser encontrado, e com o uso dessa solução resolver o problema da gasolina. O uso da recursividade para fazer as trocas, no caso base ele verifica se o começo e o final do vetor estão preenchidos por 0, se isso ocorrer, a função calcula a distância de todo o percurso. Mas o problema consiste em que existem algumas restrições para definir esses caminhos, com o uso de grafo, a ideia é que cada caminho, ou seja, cada haste tem um peso diferente, e assim cada caminho pode haver diferença entre o peso destas hastes, e isso entra na resolução do cálculo do melhor caminho, pois essa prioridade pode afetar os resultados finais. Depois de encontrar a distância final, é feita uma comparação para que se a distância mínima já encontrada, for maior que a distância encontrada na chamada atual, então teremos uma nova distância mínima. Assim a chamada é finalizada com o caso base e retorna para a main e finaliza o programa.

.Pseudocódigo:

```
Vetorpermutacao[n+1];
Vetorprecedencia[nprecedencia]
If(vetorpermuta[0] ==  0 e vetorpermuta[n] == 0)
        Enquanto contador <= n
                For(i = 0; i < nprecedencia; i++){
                        Se (vetorpermutacao[contador] ==
vetorprecedencia[i](entrega prod)){
                                Se (vetorpermutacao[contador] != 0)
                                        Proxima permutação;
                        Se (vetorpermutação[contador] ==
vetorprecedencia[i](pegar prod){
                                Vetorprecedencia[i](entrega produto) = 0;
                                Vetorprecedencia[i](pegar produto) = 0;
                        }
                }
        contrador++;
        }
        Distancia = Σdistancias entre caminhos;
        Se distancia < distanciaminima
                Distanciaminima = ditancia;
Else
Próxima permutação;
```

Considerações finais

O trabalho prático consistiu em averiguar se as permutações de ordem das cidades se encaixava nas especificações de entrega, sendo uma delas a cidade de origem e a cidade final, e as precedências de entrega, ter que passar em uma cidade para pegar um produto que tem que ser entregue, e esse foi uma das maiores dificuldades do trabalho, verificar se o caminhão fez o ciclo de passar por todas as cidades e voltar a origem e atender a ordem de precedência, assim podando o problema quando ele não respeitar a ordem de precedência.