INATEL - INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

Projeto de Algoritmos e Estrutura de Dados III

C204 - 2022/1 - L2

Nomes:

Arthur Bueno Silva

Marcos Guerra Soares

Pedro Augusto Barbosa Aparecido

Sumário

Introdução

O objetivo é mostrar a solução de um problema de entregas para um supermercado. E também para conseguir fazer as entregas em menos tempo e mostrar qual caminho cada entregador terá que percorrer.

O supermercado que irá implementar esse serviço onDemand é o UniNorte para a cidade Santa Rita do Sapucaí. E com isso foi elaborada uma forma de mostrar a melhor rota possível para se entregar a compra e o menor tempo possível. Considerando que possui vários caminhos e várias compras para entregar.

Descrição do Projeto

O projeto foi feito na linguagem C++ e com isso foi utilizado várias estruturas de busca e ordenação para conseguir resolver o problema. O algoritmo começa lendo a quantidade de compras, entregadores e casas, após isso lê a capacidade do entregador, o ponto inicial, o tempo de cada entregador até a casa inicial e por fim as casas e seus pesos para montar criar as arestas.

Após toda a leitura dos dados ordenamos por meio do quicksort os entregadores que estão mais próximos do ponto inicial:

```
163  void quickSort(Entregadores vetor[], int i, int j)
164  {
165      Entregadores trab, pivo;
166      int esq = i;
167      int dir = j;
168      pivo.distanciaOrigem = vetor[(int)round((esq + dir) / 2.0)].distanciaOrigem;
169      do
170      {
171          while (vetor[esq].distanciaOrigem < pivo.distanciaOrigem)
172              esq++;
173          while (vetor[dir].distanciaOrigem > pivo.distanciaOrigem)
```

Por apresenta uma complexidade menor em relação ao método força bruta e um resultado bem parecido optamos por utilizar o método guloso para selecionar as compras que cada entregador vai levar:

```
125     for (int i = (qtdCompras - 1); i >= 0; i--) {
126         for (int j = 1; j <= capacidadeMax; j++) {
127             int pega, naoPega;
128             naoPega = pd[i + 1][j];
129             if ((j >= compras[i].peso) && !(compras[i].selected))
130                 pega = pd[i + 1][j - compras[i].peso] + compras[i].id;
131             else
132                 pega = 0;
133             if ((pega > naoPega) /*&& (buscaIDS(ignoreIndex, 0, (tamIgnoreIndex - 1), compras[i].id) != -1)*/) {
134                 pd[i][j] = pega;
135                 caminhoComprasSelecionadas[i][j] = 1;
136             } else {
137                 pd[i][j] = naoPega;
138                 caminhoComprasSelecionadas[i][j] = 0;
139             }
140         }
141     }
```

Por meio do algoritmo de Dijkstra nós verificamos o caminho de custo mínimo que aquele entregador pode fazer para realizar suas entregas:

```
33  // Verifica o melhor caminho
34  void dijkstra(list<no>adj[], int nVertices, int start, int end, Entregadores &entregador, int &c)
35  {
36      // Declaração de variáveis
37      int u = 0;
38      int v = 0;
39      list<no>::iterator p;
40      int destino = 0;
41      float weight = 0;
42      float dist = 0;
43      bool intree[nVertices];
44      float distance[nVertices];
45      int parent[nVertices];
```

Análise de Complexidade

Obs :Os logs do gprof para cada teste estão anexados junto ao projeto, por algum motivo o tempo de execução está dando 0 em todos os casos de teste.

```
B = numero de ligações entre cidades
funcao: buscaIDS
O(n): n/2
funcao: verificaCompras
O(n): n
funcao: djikstra
O(n): n²
funcao: cria_aresta
O(n): 1
funcao: selecionaComprasParaEntregadores
O(n): nlog(n)
funcao: quickSort
O(n): nlog(n)
funcao: main
O(n): nCompra/nEntregador + O(n log n) + (nEntregadores * ((n log n) + (n²)) + (B * n²)/2 + n + nEntregadores
```

Big-O: n² * (nEntregadores + B÷2)

Testes e Resultados

TESTE 1, é o teste que foi cedido como exemplo para o desenvolvimento do projeto:

```
5
4
18
6
5
1 7
2 5
3 10
4 9
6 4
10
9
9
3
1 2 1
1 3 5
1 4 7
1 5 4
1 6 9
2 1 1
2 3 14
2 4 6
2 5 8
2 6 11
3 1 5
3 2 14
3 4 8
3 5 10
3 6 10
4 1 7
4 2 6
4 3 8
4 5 12
4 6 4
5 1 4
5 2 8
5 3 10
5 4 12
5 6 8
6 1 9
6 2 11
6 3 10
6 4 4
```

```
6 5 8
-1 -1 -1
```

RESULTADO 1:

```
Tempo que o entregador 4 gastou: 18
Caminho que o entregador 4 percorreu: 5 1 2 4 6
Compras entregues pelo entregador 4 : 2 4 6
==========================================
Tempo que o entregador 3 gastou: 18
Caminho que o entregador 3 percorreu: 5 1 3
Compras entregues pelo entregador 3 : 1 3
==========================================
```

TESTE 2, usamos 10 compras para 4 entregadores:

```
10
4
18
6
5
1 7
2 5
3 10
4 9
6 4
1 9
2 7
3 5
4 12
6 7
6
4
8
12
1 2 1
1 3 5
1 4 7
1 5 4
1 6 9
2 1 1
2 3 14
2 4 6
2 5 8
2 6 11
3 1 5
3 2 14
3 4 8
3 5 10
3 6 10
```

```
4 1 7
4 2 6
4 3 8
4 5 12
4 6 4
5 1 4
5 2 8
5 3 10
5 4 12
5 6 8
6 1 9
6 2 11
6 3 10
6 4 4
6 5 8
-1 -1 -1
```

RESULTADO 2:

```
Tempo que o entregador 2 gastou: 32
Caminho que o entregador 2 percorreu: 5 6 3 6
Compras entregues pelo entregador 2 : 6 3 6
==========================================
Tempo que o entregador 1 gastou: 23
Caminho que o entregador 1 percorreu: 5 1 4 2
Compras entregues pelo entregador 1 : 4 2
==========================================
Tempo que o entregador 3 gastou: 19
Caminho que o entregador 3 percorreu: 5 1 2 4
Compras entregues pelo entregador 3 : 2 4
==========================================
Tempo que o entregador 4 gastou: 21
Caminho que o entregador 4 percorreu: 5 1 3
Compras entregues pelo entregador 4 : 1 3
==========================================
Tempo que o entregador 1 gastou: 12
Caminho que o entregador 1 percorreu: 5 1
Compras entregues pelo entregador 1 : 1
==========================================
```

TESTE 3, usamos 10 compras para 2 entregadores que suportam apenas 10 kg:

```
10
2
10
6
5
1 7
2 5
3 10
4 9
6 4
1 9
2 7
3 5
4 10
6 7
6
4
1 2 1
1 3 5
1 4 7
1 5 4
1 6 9
2 1 1
2 3 14
2 4 6
2 5 8
2 6 11
3 1 5
3 2 14
3 4 8
3 5 10
3 6 10
4 1 7
4 2 6
4 3 8
4 5 12
4 6 4
5 1 4
5 2 8
5 3 10
5 4 12
5 6 8
6 1 9
6 2 11
6 3 10
6 4 4
6 5 8
-1 -1 -1
```

RESULTADO 3:

```
Tempo que o entregador 2 gastou: 20
Caminho que o entregador 2 percorreu: 5 6 3
Compras entregues pelo entregador 2 : 6 3
==========================================
Tempo que o entregador 1 gastou: 14
Caminho que o entregador 1 percorreu: 5 6
Compras entregues pelo entregador 1 : 6
==========================================
Tempo que o entregador 1 gastou: 19
Caminho que o entregador 1 percorreu: 5 1 4
Compras entregues pelo entregador 1 : 4
==========================================
Tempo que o entregador 2 gastou: 21
Caminho que o entregador 2 percorreu: 5 1 4 3
Compras entregues pelo entregador 2 : 4
==========================================
Tempo que o entregador 2 gastou: 19
Caminho que o entregador 2 percorreu: 5 1 3
Compras entregues pelo entregador 2 : 3
==========================================
Tempo que o entregador 1 gastou: 17
Caminho que o entregador 1 percorreu: 5 1 2
Compras entregues pelo entregador 1 : 2
==========================================
Tempo que o entregador 1 gastou: 13
Caminho que o entregador 1 percorreu: 5 1 2
Compras entregues pelo entregador 1 : 2
==========================================
Tempo que o entregador 2 gastou: 14
Caminho que o entregador 2 percorreu: 5 1
Compras entregues pelo entregador 2 : 1
==========================================
Tempo que o entregador 2 gastou: 8
Caminho que o entregador 2 percorreu: 5 1
Compras entregues pelo entregador 2 : 1
==========================================
```

Conclusão

Este projeto tem como objetivo utilizar todos os conceitos aprendidos até o momento da matéria de Algoritmos, para conseguir a devida resolução que se espera. Os conceitos utilizados foram: dijkstra, quicksort e método guloso. E até mesmo os conceitos de struct, vetor, estruturas de repetição, matriz, funções, dentre outros. Por conta disso, conseguimos realizar o projeto e nos aperfeiçoamos.

Referências

- Documentação do Microsoft C/C++ | Microsoft Docs