TP 01 - Trabalho Prático 01

Algoritmos I

Entrega: 23/11/2021

1 Objetivos do trabalho

O objetivo deste trabalho é modelar o problema computacional descrito a seguir utilizando uma estrutura de dados que permita resolvê-lo de forma eficiente com os algoritmos estudados nesta disciplina.

Serão fornecidos alguns casos de teste bem como a resposta esperada para que o aluno possa verificar a corretude de seu algoritmo. Não obstante, recomenda-se que o aluno crie casos de teste adicionais a fim de validar sua própria implementação.

O código-fonte da solução e uma documentação sucinta (relatório contendo não mais do que 5 páginas) deverão ser submetidos via *moodle* até a data limite de 23/11/2021. A especificação do conteúdo do relatório e linguagens de programação aceitas serão detalhadas nas seções subsequentes.

2 Definição do problema

No mês de Novembro acontece uma das datas mais esperadas pelo varejo nacional, o Black Friday. E este ano não poderia ser diferente, principalmente pelo baixo volume de vendas dos últimos meses devido ao isolamento social e às restrições de funcionamento enfrentadas pelos lojistas.

Uma das restrições que continua vigente é a quantidade total de pessoas que podem acessar o interior das lojas simultaneamente. Por isso, para otimizar o volume de vendas e evitar grandes filas e aglomerações, a principal empresa do varejo nacional decidiu implementar uma estratégia de agendamento para atendimento de seus melhores clientes no dia da promoção Black Friday. A empresa possuí múltiplas lojas espalhadas pela cidade. Cada loja possui em seu estoque uma quantidade já definida dos produtos da empresa e, para evitar que as pessoas desistam da compra devido à distância de sua residência, foi definido que cada cliente deve ser agendado o mais próximo possível do endereço de sua casa.

O sistema de cadastro da empresa possui as informações de cada cliente e loja disponíveis. Onde cada cliente c possui as informações de número de identificação id_c, idade i_c, estado de origem uf_c, tipo de pagamento mais frequente u_c e a localização geográfica (x_c, y_c) da sua residência. O sistema também possui as informações de cada loja l, como o número de identificação id_l, localização geográfica (x_l, y_l) e o estoque l_e de produtos disponíveis.

Devido a regras tributárias, foi definido pela equipe de negócios que clientes de diferentes estados deveriam possuir prioridades diferentes e, por isso, foi necessário a definição de um valor de score para cada estado do país cadastrado no sistema. Pelo mesmo motivo, também foi criado um valor de score para cada tipo de pagamento aceito pelas lojas.

O score de cada estado é dado pela tabela abaixo:

ESTADO	**SCORE**
MG	0
PR	10
SP	20
SC	30
RJ	40
RN	50
RS	60

O score de cada tipo de pagamento é dado pela seguinte tabela:

PAGAMENTO	**SCORE**
DINHEIRO	1
DEBITO	2
CREDITO	3

A sua empresa foi selecionada para desenvolver um software que seja capaz de automatizar o agendamento dos clientes para uma determinada loja. A priorização dos clientes deve ser realizada de acordo com o número de ticket de cada um. Este ticket deverá ser calculado de acordo com a fórmula abaixo.

$$ticket = \frac{(abs(60 - idade\_cliente) + score\_estado)}{score\_tipo\_pagamento}$$

.

Onde *idade_cliente* é a idade do cliente obtida do sistema da empresa, *score_estado* é o valor do score do estado de origem do cliente e *score_tipo_pagamento* é o score do tipo de pagamento mais frequente utilizado pelo cliente.

* *abs* é o valor absoluto

Este software deverá:

- Ler o id, a localidade geográfica, a idade, o estado de origem e o tipo de pagamento geralmente utilizado nas compras de cada cliente;
- Ler o id, a localidade geográfica e o estoque de cada uma das lojas da empresa;
- Agendar as pessoas para serem atendidas em cada uma das lojas. As lojas devem priorizar clientes com o maior número de ticket e a quantidade em estoque deve ser respeitada para que agendamentos desnecessários não sejam realizados. Em caso de empate, deve ser priorizado o cliente com maior número de id (posição sequencial no arquivo de entrada);

O seu programa deve se atentar para as seguintes condições:

- Condição de estabilidade: se um cliente não for agendado em uma loja mais próxima a ele, não deve haver estoque disponível naquela loja e nem clientes com tickets menores que o dele agendados para esta mesma loja;
- Regra de desempate de clientes: no caso de clientes com mesmo número de ticket de priorização, terá prioridade aquele com o menor número de id (posição sequencial no arquivo de entrada);
- Regra de desempate de lojas: se duas lojas com capacidade de atendimento desejam agendar um cliente equidistante, ele deverá ser agendado a loja com menor id (posição sequencial no arquivo de entrada);

3 O que fazer?

O objetivo deste trabalho será agendar, com a melhor configuração possível de acordo com as regras definidas, os principais clientes nas lojas mais próximas a cada um deles. Considere um processo de agendamento no qual n clientes devam ser agendados a m lojas. Dessa forma, um cliente c deverá ser designado para a loja com a menor distância da sua residência (a) se existir estoque disponível para atendimento a este cliente ou (b) se a loja, mesmo sem estoque disponível, tiver um agendamento de algum cliente q com ticket menor que c.

O mapa de localização deve ser interpretado como um grid de $N \times M$ posições e a localização geográfica será definida como a posição (x, y) do elemento no grid. O cálculo da distância deverá ser realizado considerando o número minimo de posições no grid que se deve caminhar para atingir a localização desejada, podendo realizar movimentos diagonais, verticais e horizontais.

Para o cálculo da distância deve ser aplicado a regra para obter o número mínimo de quadrados dentro do grid, representando a distância mínima que um cliente deve percorrer para chegar até a loja (não é considerado nenhum tipo de obstruçcão no cálculo das distâncias, ou seja, se houver um cliente/loja em meio do caminho, não deve ser desconsiderado como obstrução).

A condição essencial que deve ser satisfeita para um agendamento é que ele não apresente **"instabilidades"**. Em outras palavras, queremos que o agendamento seja o melhor possível dentro das regras definidas, ou seja, respeitando todas as restrições do problema (número do ticket, localização e estoque de cada loja). Uma alocação será instável e inválida quando alguma das seguintes situações ocorrer:

- Um cliente c_1 está agendado para uma loja l_1, mas há uma loja l_2 mais próximo que possui estoque disponível para atender sua demanda ou possui um agendamento para um cliente c_2 de ticket menor que c_1.
- Há um cliente sem agendamento mas ainda existe estoque disponível para atendê-lo em alguma loja.

Além disso, o agendamento deve ser **simultaneamente a melhor possível para todas as lojas**, respeitando as restrições descritas anteriormente. Ou seja, o agendamento será **ótimo do ponto de vista das lojas**. Sendo assim, a loja mais próxima ao cliente de maior ticket sempre conseguirá agendamento para este cliente se existir estoque disponível para sua necessidade. Os clientes possuem lista de prioridades de cada loja definida pela distância de cada um. Cada cliente possui um identificador id, $0 \leq id \leq n$ e cada loja possui um identificador l, $0 \leq l \leq m$. **Será adotada como premissa que, quando outros aspectos forem iguais, o cliente ou loja com menor identificador terá prioridade**.

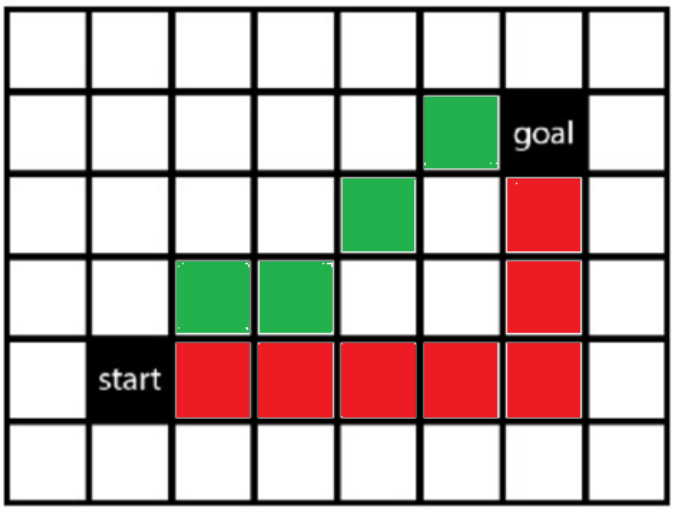

Figura 1: Exemplo de cálculo da distância $d = 4$, o caminho verde representa a distância mínima. O caminho vermelho representa outro possível caminho, mas com maior distância

4 Exemplo do problema

A seguir é apresentado um exemplo de alocação estável para um caso de teste. Observe na Figura 2 na qual existem 2 lojas disponiveis (L0, L1), ambas com quantidade de 3 produtos em estoque no dia da BLACK FRIDAY. Existem 7 clientes disponíveis para agendamento (P0, P1, P2, P3, P4, P5, P6). Considere ainda a quantidade de produtos em estoque de cada loja e o ticket de prioridade de cada cliente listados na Figura 2.

Após obter a distância no grid de cada cliente para cada loja, teremos a seguinte lista de prioridades para cada cliente (desempatando pelo identificador das lojas, caso apresentem a mesma distância):

Lista de proximidade por cliente (Figura 2)

```
P0 - {L0, L1}                          lista de proximidade para primeira pessoa
P1 - {L0, L1}
P2 - {L0, L1}
P3 - {L1, L0}
P4 - {L1, L0}
P5 - {L0, L1}
P6 - {L1, L0}
```

Já para as lojas teremos a seguinte lista de prioridade de atendimento em relação ao ticket de cada cliente (novamente desempatando pelo identificador dos clientes):

Lista de prioridade de clientes em cada loja (Figura 2)

```
L0 - {P4, P2, P1, P5, P3, P0, P6}      lista de prioridade para primeira loja
L1 - {P4, P2, P1, P5, P3, P0, P6}
```

Com base nestas informações teremos uma alocação estável de clientes nas lojas da seguinte forma:

Lista de atendimento de clientes em cada loja (Figura 2)

```
L0 - {P2, P1, P5}
L1 - {P4, P3, P0}
```

Nesta alocação a pessoa P6 não foi agendada para atendimento devido a falta de estoque disponível. Note que a alocação é a melhor possível do ponto de vista das lojas, visto que a loja mais próxima foi agendada para a pessoa com maior ticket de prioridade. Note também que todos os critérios definidos anteriormente são mantidos nesta alocação.

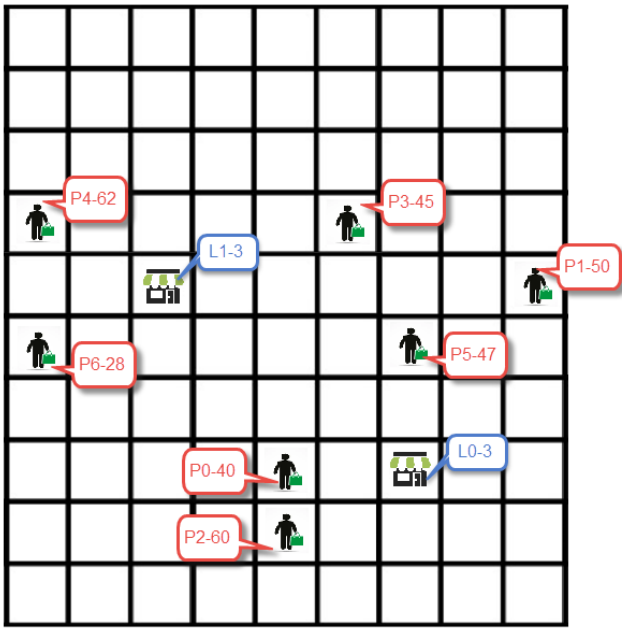

Figura 2: Exemplo de situação com duas lojas com 3 produtos cada uma e sete clientes e seus respectivos tickets de prioridade

No segundo exemplo, ilustrado na Figura 3, a solução dada envolve a observância de uma regra de desempate pelo id dos clientes. A alocação para o exemplo da Figura 3 deverá ser:

Lista de atendimento de clientes nas lojas (Figura 3)

```
L0 - {P0}
L1 - {P1, P2}
```

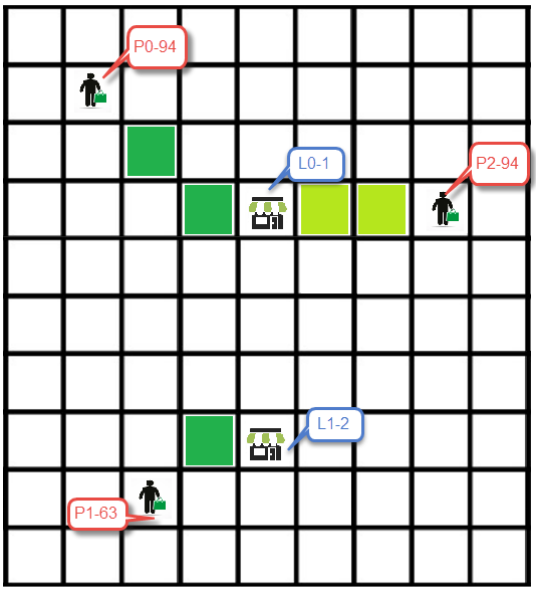

Figura 3: Exemplo de alocação com duas lojas e três clientes

No terceiro exemplo, ilustrado na Figura 4, a solução dada também envolve a observância de uma regra de desempate, mas agora pelo id das lojas. A alocação para o exemplo da Figura 4 deverá ser:

Lista de atendimento de pessoa nas lojas (Figura 4)

```
L0 - {P0}
L1 - {P1, P2}
L2 - {P3}
```

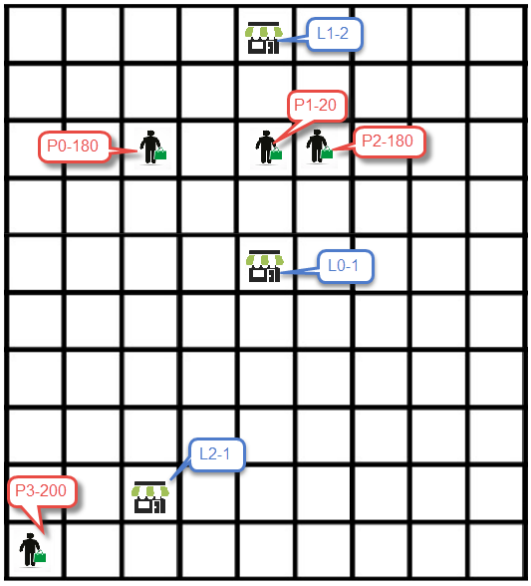

Figura 4: Exemplo de alocação com três lojas e quatro pessoas com seu score final

No quarto exemplo, ilustrado na Figura 5, temos uma demanda de alocação de quatro pessoas para três lojas. A alocação para o exemplo da Figura 5 deverá ser:

Lista de atendimento de pessoa nas lojas (Figura 5)

```
L0 - {P3, P0}
L1 - {P2}
L2 - {P1}
```

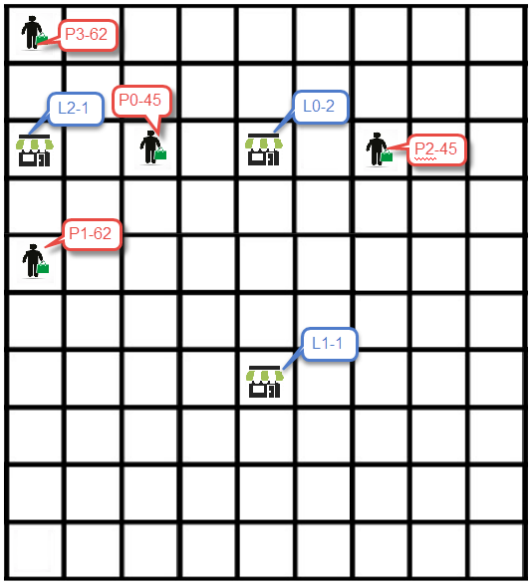

Figura 5: Exemplo de alocação com três lojas e quatro pessoas

5 Arquivos de entrada

O problema terá como entrada um arquivo contendo as informações dos clientes e das lojas da empresa. Os alunos terão que ler o arquivo e armazená-lo em uma estrutura de dados para utilizar na resolução do problema. O arquivo está organizado da seguinte forma:

A primeira linha contem dois numeros N e M, relacionado ao tamanho do grid que contém as lojas e os clientes. A proxima linha contém um número m inteiro, $0 < m \leq 10^4$, relacionado ao número total de lojas da empresa. Em sequência, cada uma das m linhas a seguir deverá conter o estoque da loja E, $0 < E \leq 10^2$, e sua localização $x < N$ e $y < M$, $0 \leq x_i, y_i \leq 10^5$, todos separados por espaço.

A próxima linha conterá um número n inteiro, $0 < n \leq 10^5$, relacionado ao número total de clientes. Em sequência, cada uma das n linhas a seguir deverá conter uma sequência de inteiros e cadeias representando, respectivamente, a idade I_i do cliente i, $0 < I_i \leq 100$, seu estado de origem Uf_i, $0 < Uf_i \leq 60$, o tipo de pagamento u_i mais frequente, u_i, $0 < u_i \leq 3$, e sua localização $x_i < N$ e $y_i < M$, $0 \leq x_i, y_i \leq 10^5$, todos separados por espaço. O id do cliente deverá ser definido como um número sequencial a partir de 0 definido pela ordem de leitura de cada registro. A mesma lógica se aplica para os Ids das lojas.

Note que há três cenários possíveis: há mais necessidade de clientes que estoque, há um número igual de necessidade de clientes e estoque em loja ou há menos necessidade de clientes que estoque em loja. A seguir está listado um exemplo de entrada e sua respectiva saída.

6 Saída

A saída deve ser impressa na tela (*stdout*) e seguir o seguinte padrão: Deverá ser impresso m saídas com a primeira linha com o identificador da loja e a segunda linha contendo uma lista de *ids* das pessoas que foram alocadas para aquela loja.

A impressão desses grupos deve ser ordenado pelo identificador da loja. Lembrando que a identificação ocorre pela posição da pessoa ou loja no arquivo (0 para a primeira loja/pessoa). Fique atento para imprimir exatamente como foi descrito pois a correção do algoritmo será feita de forma automática.

7 Exemplo Prático de Entrada e Saída

Arquivo de entrada

```
10 9                                    // <tamano do grid>
2                                       // <Número de lojas>
3 7 6                                   // <Capacidade> <localização> (Loja 0)
3 4 2
7                                       // <Número de clientes>
20 MG DINHEIRO 7 4                      // <Idade> <estado> <tipo de pagamento>
30 SP DINHEIRO 4 8                      // <localização> (Pessoa 0)
70 RN DINHEIRO 8 4
30 RS DEBITO 3 5
18 SP DINHEIRO 3 0
26 RS DEBITO 5 6
94 RN CREDITO 5 0
```

Exemplo prático da saída

```
0                                       // Loja 0
2 1 5                                   // Lista das pessoas agendadas na loja 0
1
4 3 0
```

8 Especificação das entregas

Você deve submeter um arquivo compacto (zip ou tar.gz) no formato **MATRICULA_NOME** via Moodle contendo:

- todos os arquivos de código-fonte implementados;
- um arquivo *makefile*[1] **que crie um executável com nome tp01**;
  - **ATENÇÃO:** O makefile é para garantir que o código será compilado da forma como vocês implementaram, evitando erros na compilação. É **essencial** que ao digitar "make" na linha de comando dentro da pasta onde reside o arquivo makefile, o mesmo compile o programa e gere um executável chamado **tp01**.
- sua documentação (arquivo pdf).

Sua documentação deverá ser sucinta e conter não mais do que 5 páginas com o seguinte conteúdo obrigatório:

- Modelagem computacional do problema;

[1] https://pt.wikibooks.org/wiki/Programar_em_C/Makefiles

- estruturas de dados e algoritmos utilizados para resolver o problema (pseudo-código da solução impelementada), bem como justificativa para tal escolha. Não transcreva trechos da código-fonte;
- análise de complexidade de tempo assintótica da solução proposta, devidamente justificada.

9 Implementação

9.1 Linguagem, Ambiente e Parâmetros

O seu programa deverá ser implementado na linguagem **C** ou **C++** e deverá fazer uso apenas de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de bibliotecas que não a padrão não serão aceitos.

O aluno pode implementar seu programa em qualquer ambiente (Windows, Linux, MacOS, etc...), no entanto, deve garantir que seu código compile e rode nas máquinas do DCC (tigre.dcc.ufmg.br ou jaguar.dcc.ufmg.br), pois será neste ambiente que o TP será corrigido. Note que essas máquinas são acessíveis a todos os alunos do DCC com seu login e senha, podendo inclusive ser realizado acesso remoto via ssh. O aluno pode buscar informações no site do CRC (Centro de Recursos Computacionais) do DCC (https://www.crc.dcc.ufmg.br/).

O arquivo da entrada deve ser passado ao seu programa como entrada padrão, através da linha de comando (e.g., *$ ./tp01 < casoTeste01.txt*) e gerar o resultado também na saída padrão (não gerar saída em arquivo).

ATENÇÃO: Não é necessário que o aluno implemente em ambiente Linux. Recomenda-se que o aluno teste seu código nas máquinas previamente especificadas, as quais serão utilizadas para correção do TP, a fim de conferir a funcionalidade, makefile e demais características do código.

9.2 Testes automatizados

A sua implementação passará por um processo de correção automatizado, utilizando além dos casos de testes já disponibilizados, outros exclusivos criados para o processo de correção. O formato da saída de seu programa deve seguir a especificação apresentada nas seções anteriores. Saídas diferentes serão consideradas erro para o programa. O aluno deve certificar-se que seu programa execute corretamente para qualquer entrada válida do problema.

ATENÇÃO: O tempo máximo esperado para execução do programa, dado o tamanho máximo do problema definido em seções anteriores, é de 5 segundos.

9.3 Qualidade do código

Preze pela qualidade do código-fonte, mantendo-o organizado e comentado de modo a facilitar seu entendimento para correção. Caso alguma questão não esteja clara na documentação e no código fonte, a nota do trabalho pode ser penalizada.

10 Critérios para pontuação

A nota final do TP (NF) será composta por dois fatores: fator parcial de implementação (fpi) e fator parcial da documentação (npd). Os critérios adotados para pontuação dos fatores é explicado a seguir.

10.1 Fator parcial de implementação

Serão avaliados quatro aspectos da implementação da solução do problema, conforme a Tabela 1.

Aspecto	**Sigla**	**Valores possíveis**
Compilação no ambiente de correção	co	0 ou 1
Respostas corretas nos casos de teste de correção	ec	0 a 100%
Tempo de execução abaixo do limite	te	0 ou 1
Qualidade do código	qc	0 a 100 %

Tabela 1: Aspectos de avaliação da implementação da solução do problema

O fator parcial de implementação será calculado pela seguinte fórmula:

$$fpi = co \times (ec - 0,15 \times (1 - qc) - 0,15 \times (1 - te))$$

Caso o valor calculado do fator seja menor que zero, ele será considerado igual a zero.

10.2 Fator parcial da documentação

Serão avaliados quatro aspectos da documentação entregue pelo aluno, conforme a Tabela 2.

Aspecto	**Sigla**	**Valores possíveis**
Apresentação (formato, clareza, objetividade)	ap	0 a 100%
Modelagem computacional	mc	0 a 100%
Descrição da solução	ds	0 a 100%
Análise de complexidade de tempo assintótica	at	0 a 100 %

Tabela 2: Aspectos de avaliação da documentação

O fator parcial de documentação será calculado pela seguinte fórmula:

$$fpd = 0,4 \times mc + 0,4 \times ds + 0,2 \times at - 0,25 \times (1 - ap)$$

Caso o valor calculado do fator seja menor que zero, ele será considerado igual a zero.

10.3 Nota final do TP

A nota final do trabalho prático será obtida pela equação a seguir:

$$NF = 10 \times (0,6 \times fpi + 0,4 \times fpd)$$

É importante ressaltar que é obrigatória a entrega do código fonte da solução e documentação. Na ausência de um desses elementos, a nota do trabalho prático será considerada igual a zero, pois não haverá possibilidade de avaliar adequadamente o trabalho realizado.

Assim como em todos os trabalhos dessa disciplina é estritamente proibida a cópia parcial ou integral de código-fontes, seja da Internet ou de colegas. Se for identifcado o plágio, o aluno terá a nota zerada e o professor será informado para que as medidas cabíveis sejam tomadas.

ATENÇÃO: Os alunos que submeterem os TPs com atraso, terão a nota final penalizada em termos percentuais de acordo com a seguinte regra: $2^{d-1}/0,16$ (onde d é a quantidade de dias úteis de atraso na entrega do TP)