Os caixotes de chuchu da fazendeira Mitski

Prof. Diogo Soares'

1 Introdução

Na longa estrada da BR 174, Mitski é uma fazendeira pacata que vive da venda de seus chuchus em caixotes, que ela cuida ativamente na sua pequena fazenda à beira da estrada. Neste contexto, Mitski segue sua saga para vender ao máximo seus caixotes sem ter muita perda de chuchus (pela demora de venderem e estragarem) e vendendo com o máximo de qualidade. Afinal, servir bem para servir sempre 😁.

Figure 1: Caixotes de chuchu da fazendeira.

No entanto, Mitski anda descontente com seus ajudantes, pois seus caixotes têm apresentado chuchus estragados em algumas unidades, o que ajuda os clientes a não sentirem mais vontade de retornar para comprar mais chuchus. Os ajudantes informaram que isso ocorre pelo fato de que as remessas estão enormes, não havendo mais como manter um controle de qualidade sem a ajuda de ferramentas computacionais.

A ajudante Olívia Rogéria, braço direito de Mitski, comentou que há um(a) grande especialista no desenvolvimento de ferramentas computacionais, e que poderia resolver esta situação. Aliás, um(a) não, mas dois/duas especialistas. E claro que ela está falando de você(s) :)! Mitski amou a ideia, e lhe enviou os detalhes do problema a ser resolvido. Vamos lá, não decepcione nossa amável fazendeira.

2 Detalhes do Problema

O objetivo neste trabalho é exercitar e praticar os conceitos relacionados à construção de algoritmos e estruturas de dados. O problema principal segue a premissa: *Mitski deseja saber e controlar os caixotes de chuchu que compõem o seu estoque para venda.* Para tal, ela necessita de um sistema computacional que as organize em 3 estruturas, listadas a seguir:

- **Preparação para a venda**: antes da venda, os caixotes são posicionados de acordo com sua **aptidão** para a venda. Só que tem uma peculiaridade, Mitski é superticiosa e organiza os caixotes que chegam na ordem inversa. Ou seja, o **primeiro caixote** que ele informará para o sistema é o menos **apto** para a venda, assim, **deve ser o último, de fato, a ser vendido**. Da mesma maneira, o último caixote informado no sistema será o primeiro a ser vendido.
- **Venda**: Mitski necessita de um controle para os caixotes que estão na fila para a venda. Dado o **comando** de venda, a caixa selecionada **deve ser a mais apta para a venda que está aguardando**, baseado na posição armazenada na estrutura de preparação para a venda.
- **Caixa avariada**: Como em todo processo de venda em ambiente de feira, algumas caixas podem ter alguma avaria, isto é, alguns dos chuchus em uma caixa podem estar batidas ou estragadas. Uma vez que seja reportado que uma caixa possui uma avaria, ela deve ser removida da estrutura de preparação para a venda, e ser incluída na estrutura de caixas avariadas. Uma vez avariada, os outros ajudantes de Mitski irão remover os chuchus avariados da caixa e adicionar novos chuchus, mitigando o processo de perda de chuchus. É importante frisar que, dadas duas caixas, **caixa1** e **caixa2**, se a **caixa1** foi avariada antes da **caixa2**, então a **caixa1 será consertada antes** da **caixa2**. Ou seja: as caixas avariadas primeiro, são consertadas primeiro! Uma vez que a equipe de manutenção dos caixotes avisa que uma caixa foi corrigida, a caixa passa a aguardar a ordem de Mitski para entrar novamente na estrutura de venda, tendo preferência sobre todas as outras que já estão aguardando venda.

3 Entrada e Saída

Neste trabalho, a entrada será a padrão do sistema (**stdin** ou pela linha de comando). A saída também será a padrão do sistema (**stdout**). A primeira linha da entrada é composta por um valor inteiro **N** ($0 < \mathbf{N} \leq 5000$), que indica o total de caixas geradas na fazendinha da Mitski. A seguir, haverão **N** linhas compostas por números inteiros, que representam o identificador de cada caixa. Após, a entrada é composta por várias linhas, cada qual contendo um inteiro **I**, que indica a ação a ser realizada no sistema computacional, cujos significados são descritos a seguir:

- **0**: indica que Mitski deseja enviar a caixa mais apta para a venda. Ao ser enviada, a mensagem "caixa K enviada para a venda" deve ser impressa na saída, na qual K é o identificador da caixa;
- **X, tal que X é um identificar de caixa**: indica que a caixa de identificador X, que estava a venda, foi avariada. Ao ser avariada, a mensagem "caixa K avariada" deve ser impressa na saída, na qual K é o identificador da caixa;
- **-1**: indica que a equipe de manutenção informou que uma caixa foi avariada. Lembre-se que a caixa corrigida é sempre com a maior prioridade, aquela que chegou primeiro avariada. Ao ser consertada, a mensagem "caixa K consertada" deve ser impressa na saída. Lembre-se que a caixa corrigida deve ser levada novamente para a estrutura de preparação para a venda, tendo preferência sobre as que já estão lá;
- **-2**: indica que Mitski deseja obter uma impressão de todas as caixas aguardando para serem vendidas, da mais apta para a menos apta. Cada caixote vendido deve ser impresso em uma linha. A impressão deve ocorrer da caixa mais para aquela menos apta;
- **-3**: indica que Mitski deseja obter uma impressão do identificador de todas as caixas avariadas e que estão aguardando para serem corrigidas. Cada caixote avariado deve ser impresso em uma linha. A impressão deve ocorrer da caixa com mais prioridade para ser corrigida até a com menos prioridade.

O final da entrada é indicado pelo EOF (CTRL + D no terminal em geral).

4 Informações Adicionais

Para este trabalho, assuma que todo o sistema é coeso e que nenhuma entrada absurda é passada como entrada, e os seguintes itens são verdades absolutas:

- Todas as caixas possuem identificadores distintos;
- **Não haverá** um identificador cuja valor seja o mesmo de uma instrução reservada do sistema (0, -1, -2, -3);
- **Não haverá** um comando de caixa avariada com identificador X, tal que X não esteja apta para a venda;
- **Não haverá** comandos de caixas para entrarem em venda, caso não haja caixas aguardando para serem vendidas;
- **Não haverá** comandos para corrigir caixas, caso não existam caixas avariadas;
- Mitski não pedirá impressão de caixas aguardando para serem vendidas caso não haja caixas aguardando para serem vendidas;
- Mitski não pedirá impressão de caixas avariadas caso não haja caixas avariadas;

5 Exemplo de Execução e Saída

Exemplo de Entrada

```
3
5874
3259
8412
0
-2
0
3256
8412
-3
-1
-2
0
```

Exemplo de Saída

```
caixa 8412 enviada para a venda
3256
5874
caixa 3256 enviada para a venda
caixa 3256 avariada
caixa 8412 avariada
3256
8412
caixa 3256 consertada
```

```
3256
5874
caixa 3256 enviada para a venda
```

6 Exemplo passo-a-passo

Para ilustrar o funcionamento do sistema, vamos explanar o exemplo de entrada e saída, apresentados na seção anterior. Nesse exemplo, Mitski indica que três caixotes estão inicialmente aguardando para serem vendidos: 5874, 3256 e 8412. Lembre-se: ela informa da caixa menos apta para a mais apta. A inserção das caixas nessa estrutura (**deve ser um *array* dinâmico ou estático!**) deve ter um comportamento similar ao mostrado na Figura 2.

Figure 2: Caixas sendo inseridas na estrutura de aptas para a venda.

A próxima instrução requerida por Mitski foi a de que a caixa mais apta está pronta para ser vendida e que pode sair do estoque (*instrução 0*). Dessa maneira, a estrutura das caixas prontas para a venda deve receber a caixa 8412. As estruturas envolvidas devem se comportar da maneira como indica a Figura 3. A seguinte mensagem deve ser impressa: **caixa 8412 enviada para a venda**.

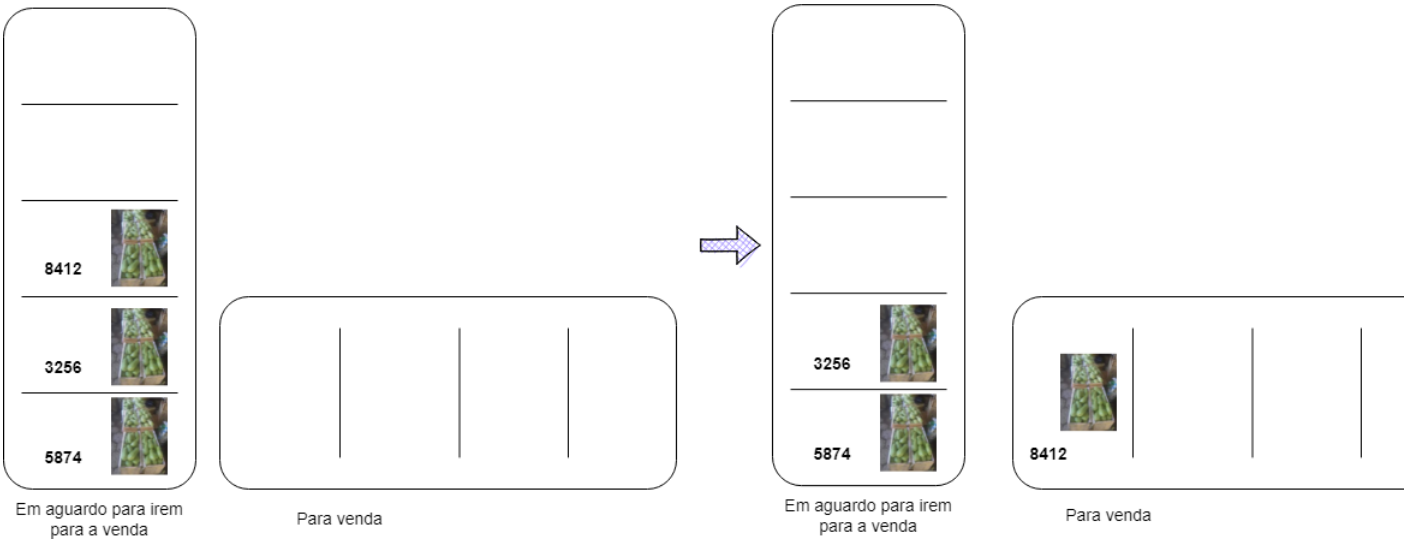

Figure 3: Caixa 8412 indo para a lista de caixas a venda.

A próxima instrução solicitada é de impressão das caixas aguardando para irem para a venda (*instrução 2*). Como a impressão deve ocorrer da de maior aptidão para menor, a impressão será:

```
3256
5874
```

Na sequência, uma nova instrução de caixa pronta para a venda é enviada. Assim, a caixa mais apta, de identificador 3256, entra a venda. A Figura 4 ilustra o comportamento das estruturas envolvidas. Deve ser impresso na saída padrão a seguinte mensagem:

```
caixa 3256 enviada para a venda
```

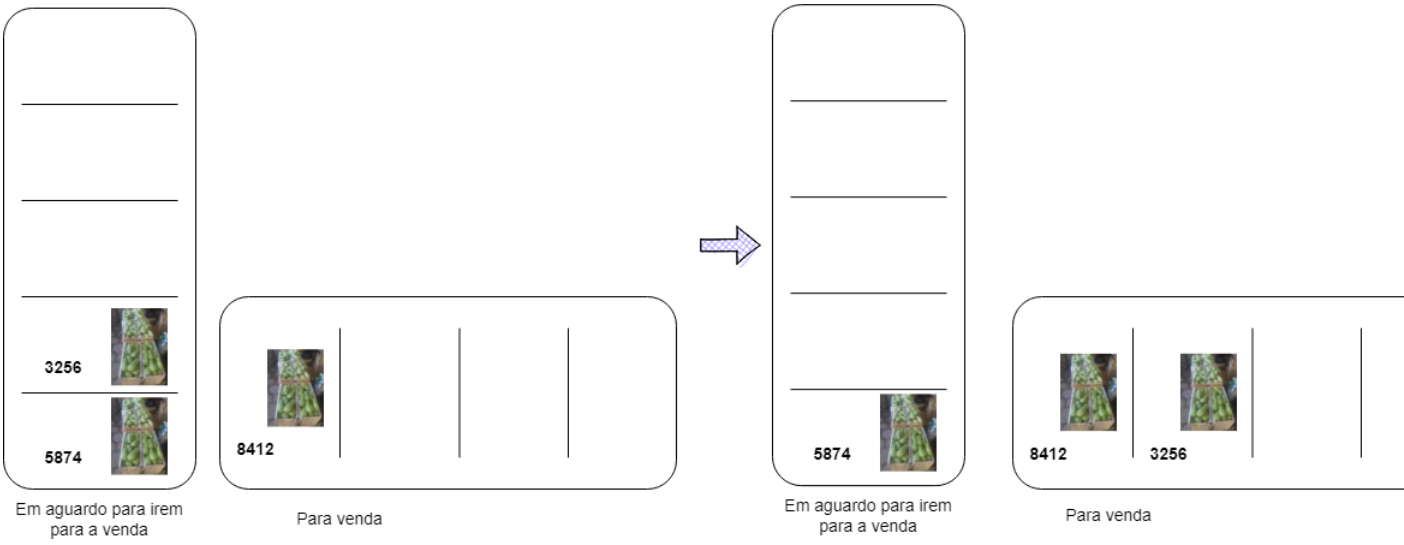

Figure 4: Caixa 8412 indo para a lista de caixas a venda.

Após, é dada uma instrução com identificador 3256, indicando que essa caixa foi avariada (lembre se que ela não foi adicionada na lista de aptas para a venda, pois ela já estava lá!). Assim essa caixa deve ser removida da estrutura de aptas para a venda e ser inserida na estrutura de caixas avariadas. A Figura 5 ilustra esse comportamento. A mensagem a seguir deve ser impressa:

```
caixa 3256 avariada
```

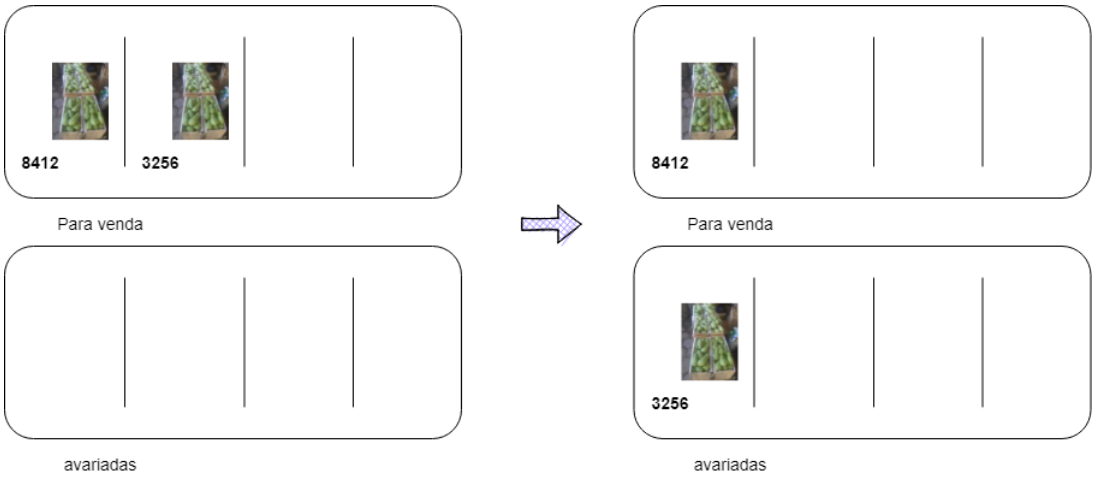

Figure 5: Caixa 3256 avariada.

Na sequência, é informado que a caixa 8412 foi avariada. Deve-se então realizar manipulações nas estruturas semelhante a instrução anterior, conforme mostra a Figura 6. A seguinte mensagem deve ser impressa na saída:

```
caixa 8412 avariada
```

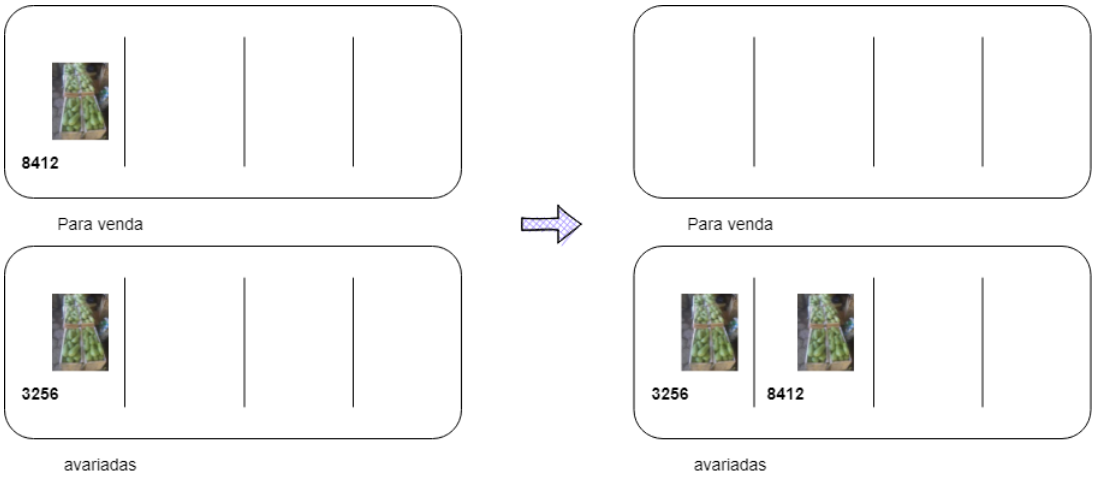

Figure 6: Caixa 8412 avariada.

Após, é dada a instrução de impressão das caixas avariadas (*instrução -3*). Lembre-se que a impressão deve ocorrer da caixa que será consertada primeiro até a que será consertada por último.

```
3256
8412
```

A próxima instrução é relacionada ao conserto de uma caixa (*instrução -1*). Assim, 3256 foi consertada e pode voltar para a lista de aptas para a venda. Lembre-se que a caixa consertada tem mais aptidão do que todas as outras já adicionadas na lista de aptas para a venda. A Figura 7 ilustra esse comportamento. Na saída deve ser impresso:

```
caixa 3256 consertada
```

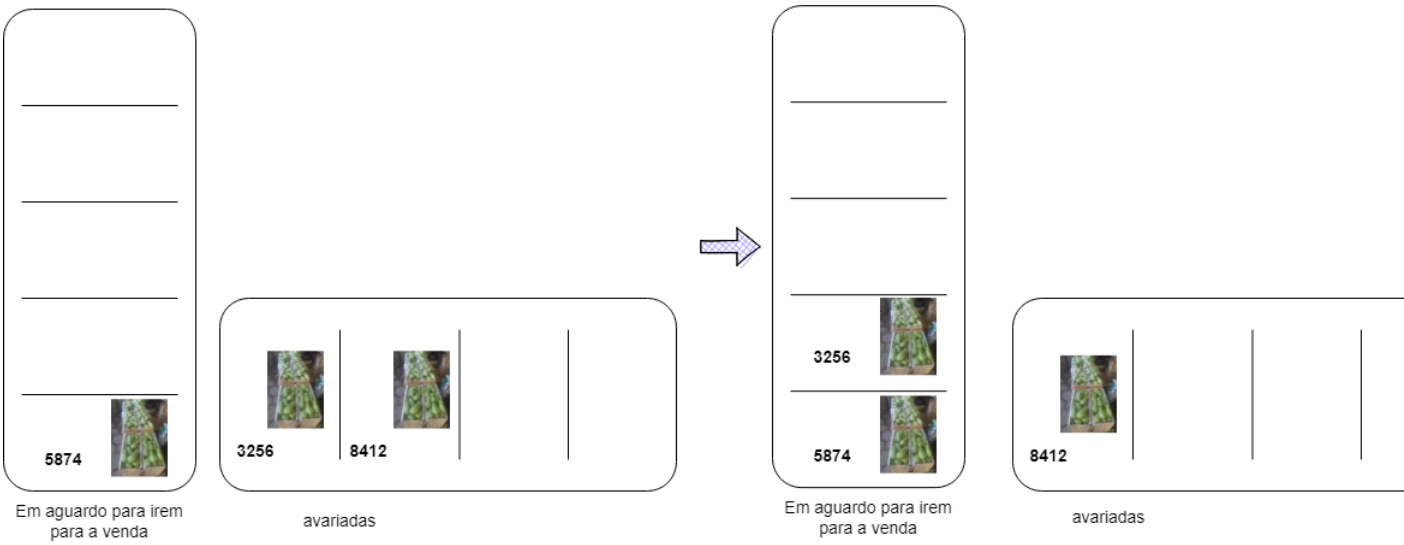

Figure 7: Caixa 3256 consertada.

Após, pede-se novamente que sejam impressos as caixas aguardando para irem para a venda (*instrução -2*). Assim, deve ser impresso na saída:

```
3256
5874
```

Por fim, a última instrução dada é da caixa mais apta entrar para a venda (*instrução 0*). Assim, 3256 vai para a venda. A figura 8 ilustra esse comportamento.

```
caixa 3256 enviada para a venda
```

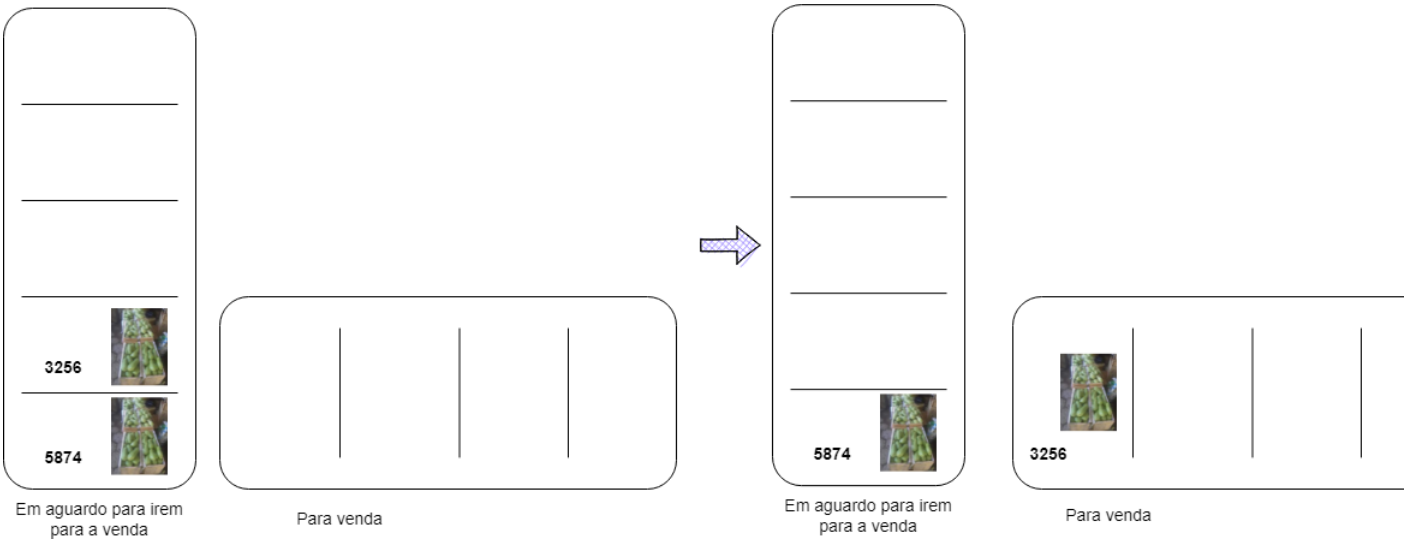

Figure 8: Caixa 3256 enviada para a venda.

7 O que deve ser entregue

Código-fonte. A implementação deve ser feita em C. Utilize as bibliotecas padrões de C que você(s) acharem necessário. **Plágio é crime!**.

Aplique boas práticas de programação e organize seu código em funções, procedimento com o significado de cada parte. A separação e modulação é um dos princípios da engenharia de software: cada função procedimento deve realizar uma função única e bem definida e condizentes com sua semântica.

Relatório. Entregue um pequeno relatório (1/2/3 páginas no máximo) com os detalhes da sua implementação, solução e considerações feitas na realização desse trabalho (considerações com os valores de entrada por exemplo). Também considere as limitações do seu trabalho, possíveis pontos de falha do código, etc.

8 Considerações Finais

PRESTE BASTANTE ATENÇÃO NOS DETALHES DA ESPECIFICAÇÃO DO TRABALHO!

O que será avaliado:

- Boas práticas de programação;
- Implementação corretas dos algoritmos;
- Relatório.

Bom trabalho!