Algoritmos e Estruturas de Dados II

Trabalho Prático I

Aluno: Hugo Demattos Nogueira

Matrícula: 2012076534

Turma TW

Introdução

Várias pessoas almoçam frequente na cantina do Icex. Como o número de pessoas é grande, e às vezes o atendimento não é tão eficiente, filas se formam, atrasando o almoço de alunos e funcionários. Neste Trabalho Prático faremos uma simulação do caminho percorrido pelos clientes para se servirem na Cantina.

Consideraremos que os clientes primeiramente entram na fila para comprar a ficha, e que quando um cliente está no caixa comprando a ficha ele já saiu da fila. Depois, o cliente deve entrar na fila para pegar bandeja, prato e talheres, que serão considerados como uma coisa só. Também consideraremos que quando um cliente está pegando a bandeja ele já saiu da fila. Com a badeja em mãos, os clientes devem servir os 4 alimentos, arroz, feijão, guarnição e salada, completando assim, seu atendimento.

Para tentar solucionar o problema, ou seja, diminuir o tempo de atendimento médio e aumentar o número total de pessoas servidas dentro do intervalo de tempo proposto proporemos duplicar a capacidade da cantina, fazendo com que duas pessoas comprem fichas, peguem as bandejas e se sirvam ao mesmo tempo.

Desenvolvimento

Consideraremos três casos. O primeiro será o caso inicial sugerido, com uma fila de fichas, um caixa, uma fila e uma pilha de bandejas e uma pessoa pegando a bandeja e se servindo a cada iteração. Duas pessoas entrarão na fila de fichas a cada minuto (iteração).

No caso 2, duplicaremos a capacidade do restaurante, com uma fila de ficha, dois caixas, uma fila de bandeja, duas pilhas de bandejas com 30 bandejas cada e duas pessoas servem o mesmo alimento ao mesmo tempo. Também duas pessoas entrarão na fila de fichas a cada minuto.

No caso 3 tentaremos fazer uma entrada mais realista, com uma pessoa entrando na fila de fichas por minuto na primeira hora de funcionamento, duas por minuto na segunda hora e uma por minuto nas últimas duas horas. Haverá uma fila de ficha, um caixa, uma fila de bandeja, uma pilha de bandejas com 20 bandejas que demora mais a recarregar e uma pessoa servindo o alimento de cada vez.

Consideraremos cada iteração como 1 minuto de funcionamento da cantina.

Na implementação do algoritmo para modelar o atendimento na Cantina do Icex usamos basicamente duas estruturas de dados, pilha e fila.

Foi implementado um tipo fila por meio de apontadores, para que não houvesse limitação quanto ao seu tamanho, podendo suportar um número muito grande de usuários da Cantina. A implementação das células foi a seguinte:

```
typedef struct FCelula {
   int Pessoa;
   FApontador FProx;
   } FTCelula;
```

O campo “int Pessoa;” serve para guardar o instante de tempo em que cada pessoa entrou na fila, que será subtraído do instante de tempo em que a pessoa acaba de se servir, descobrindo o tempo gasto por esta pessoa.

Para o tipo fila, foram definidas as funções:

void FFVazia(TFila *Fila); - faz fila vazia;

int VaziaFila(TFila Fila); - que retorna 1 caso a fila esteja vazia e 0 caso não esteja;

void Enfileira(int x, TFila *Fila); - enfileira o item x, que será o tempo que a pessoa entrou na fila;

void Desenfileira(int *x, TFila *Fila); - remove o item que está na frente da fila.

Para controlar a pilha de bandejas, foi definido um tipo pilha, também implementado por meio de apontadores. Para o tipo pilha, foram criadas 5 funções:

void FPVazia(TPilha *Pilha); - faz pilha vaziaç

int VaziaPilha(TPilha Pilha); - retorna 1 caso a pilha esteja vazia e 0 caso não esteja;

void Empilha(TBandeja x, TPilha *Pilha); - empilha um item do tipo bandeja no topo;

void Desempilha(TBandeja *x, TPilha *Pilha); - retira um item do tipo bandeja do topo;

int TamanhoPilha(TPilha Pilha); - retorna o número de bandejas na pilha.

Os tipos fila e pilha e suas respectivas funções foram declarados em arquivos “.h”, enquanto suas funções foram implementadas em arquivos “.c” para deixar o código mais compacto.

Implementação

Primeiramente consideraremos o caso inicial descrito acima, com apenas uma fila de fichas, um caixa, uma fila de bandejas, uma pilha de bandejas que já começava cheia e cada pessoa servindo cada alimento por vez.

Criamos uma fila para as fichas e uma para as bandejas. Criamos também uma pilha para as bandejas e uma variável do tipo bandeja (TBandeja) simplesmente para ajudar a empilhar e desempilhar as bandejas. Logo após criar esta pilha já empilhamos 30 bandejas, de acordo com o caso inicial proposto. Criamos as variáveis do tipo “int” “ficha” e “pega bandeja”, para receber o instante de tempo em que a pessoa que está comprando a ficha ou pegando a bandeja entrou na fila.

Para simular o “tempo passando”, criamos uma estrutura de repetição usando “for”. Foram feitas 240 iterações, 1 iteração representando 1 minuto, totalizando as 4 horas de funcionamento da Cantina.

Inicialmente a ideia era fazer com que o primeiro da fila iria para o caixa, depois para a fila da bandeja, e assim por diante. Porém, se implementado desse modo, haveria algumas dificuldades para impedir que um cliente que saísse da fila passasse por todos os estágios da Cantina em uma iteração só.

Então, como solução para este problema, o código foi implementado de “cabeça para baixo”. O primeiro bloco de comandos dentro do anel de repetição verifica se tem alguém saindo da fila de bandejas e se há alguma bandeja disponível. Se essas duas condições forem verdadeiras, uma bandeja é retirada da pilha, a variável inteira “n” que conta o número total de pessoas servidas é incrementada, e, principalmente, o tempo total gasto por esta pessoa desde que ela entrou na fila até servir os alimentos é calculado. Para fazer esta conta, subtraímos o instante de tempo “atual”, representado por “i”, da variável “pegaBand”, que contém o instante de tempo em que essa pessoa entrou na fila, usando para isso uma variável auxiliar. Ao tempo gasto por essa pessoa somamos 4, que é quanto ela demoraria para servir os 4 alimentos. Então adicionamos este tempo a uma variável que guarda a soma dos tempos de todas as pessoas atendidas, para calcular a média de tempo de atendimento por pessoa no final do programa.

```
if(pegaBand!=0 && (!VaziaPilha(PilhaBandeja))){
        aux=pegaBand;
        pegaBand=0;
```

```
            Desempilha(&Band, &PilhaBandeja);

            n++;

            tempoDessaPessoa=i-aux+4; //o "+4" eh o tempo que a pessoa demora para servir os alimentos

            somaTempos=somaTempos+tempoDessaPessoa;

        }

    //sai da fila de bandeja

        if(pegaBand==0 && (!VaziaFila(FilaBandeja))){

            Desenfileira(&pegaBand, &FilaBandeja);

        }
```

Continuando a lógica do código “ao contrário”, vem um conjunto de comandos que testa se alguém já comprou a ficha (se ficha!=0), e coloca essa pessoa a fila da bandeja, sendo que a variável ficha contém o momento em que a pessoa entrou na fila do caixa.

```
    if(ficha!=0){

            if(TamanhoFila(FilaBandeja)<=1){

               ficha++;

            }

            Enfileira(ficha, &FilaBandeja);

            ficha=0;
```

Depois, vem um conjunto de comandos que desenfileira a primeira pessoa da fila da ficha e a direciona para o caixa, se o caixa estiver vazio.

```
    if(ficha==0 && (!VaziaFila(FilaFicha))){

            Desenfileira(&ficha, &FilaFicha);

        }
```

Depois, vem um conjunto de comandos que faz as pessoas entrarem na fila do caixa, “salvando” o momento em que elas entram na fila. São enfileiradas duas pessoas por minuto (iteração), como proposto inicialmente.

Segue-se então um bloco de comandos que testa se já se passaram 12 instantes de tempo desde que as bandejas foram recarregadas pela última vez e se a pilha não excederá 30 bandejas depois da recarga. Se essas condições forem verdadeiras, 10 bandejas são adicionadas à pilha.

Finalmente a estrutura de repetição é fechada, e é calculado o tempo médio de atendimento por pessoa, dividindo-se o a soma do tempo gasto por todas as pessoas atendidas pelo número de pessoas atendidas.

Foram propostos mais dois casos, sendo que estes dois casos adicionais têm uma implementação muito semelhante a do primeiro caso mostrado acima.

O caso 2 é implementado quase da mesma maneira que o caso 1, porém faz cada coisa duas vezes por iteração, seguindo a mesma lógica acima. Para duas pessoas comprarem a ficha ao mesmo tempo, foram criadas as variáveis inteiras ficha1 e ficha2, e a cada interação itens da fila são desenfileirados para essas variáveis, e depois enfileirados na fila da bandeja. Os itens da bandeja são desenfileirados para as variáveis pegaBand1 (pega bandeja) e pegaBand2, e depois são usados para computar o tempo médio gasto pelos usuários, já que essas variáveis contém o instante de tempo que cada cliente entrou na fila do caixa.

O caso 3 é implementado quase exatamente da mesma maneira que o caso 1, porém com o comando de enfileirar na fila do caixa sendo usado apenas uma vez a cada iteração nas sessenta primeiras iterações (primeira hora de funcionamento), duas vezes a cada iteração nas sessenta iterações seguintes (segunda hora de funcionamento) e uma vez a cada iteração no resto das iterações (últimas duas horas de funcionamento).

Análise de complexidade

Nesta análise, consideraremos somente as operações de enfileirar, desenfileirar, empilhar e desempilhar, que são as com maior custo. Operações como atribuições e incremento de variáveis serão desprezadas.

Análise do Caso 1:

A cada iteração, 2 pessoas entram na fila da ficha, 1 sai desta mesma fila, 1 pessoa entra na fila da bandeja e 1 sai desta mesma fila. Considerando as operações relacionadas à pilha, aproximadamente 1 bandeja é desempilhada a cada iteração. As bandejas são empilhadas 10 vezes a cada 12 iterações. Assumindo que os custos de enfileirar, desenfileirar, empilhar e desempilhar são aproximadamente iguais, temos:

$$f(n) = n(2\ enfileiramentos + 1\ desenfileiramento + 1\ enfileiramento + 1\ desenfileiramento + 1\ empilhamento + \frac{10}{12} empilhamentos = \ 6{,}83n \approx 7n$$

Logo, a complexidade do algoritmo é de O(n).

Análise do caso 2:

A análise do caso 2 é parecida com a do caso 1, porém com o dobro de operações, exceto a operação de entrar na fila do caixa, que permanece igual a do caso 1:

$$f(n) = n(2\ enfileiramentos + 2\ desenfileiramento + 2\ enfileiramentos + 2\ desenfileiramento + 2\ empilhamento + \frac{20}{12} empilhamentos = \ 11{,}67n \approx 12n$$

A complexidade também é O(n).

Análise do caso 3:

Também é parecida com a do caso 1, porém com menos operações de entrada na fila do caixa:

Número de entradas na fila do caixa: 1 pessoa por iteração em ¼ das iterações, 2 por iteração em ¼ das iterações e 1 por iteração nos últimos 2/4.

$$Número\ médio\ de\ entradas: \frac{1 * \frac{1}{4} + 2 * \frac{1}{4} + 1 * 1/2}{\frac{1}{4} + \frac{1}{4} + 1/2} = 1{,}25$$

$f(n) = n(1{,}25\ enfileiramentos + 1\ desenfileiramento + 1\ enfileiramento + 1\ desenfileiramento + 1\ empilhamento + \frac{10}{12} empilhamentos = 6{,}08n \approx 6n$

Os três casos apresentam complexidade O(n), o que já era esperado, já que existe somente um anel de repetição.

Resultados

Resultados do programa nos três casos considerados:

	Caso 1	Caso 2	Caso 3
Média de tempo por pessoa em minutos	65	92	41
Número de pessoas atendidas	220	440	210

Como fica claro pela tabela acima, no caso 2, as pessoas esperam mais na média, porém o número de pessoas atendidas é muito maior. O tempo de espera no caso 2 é maior porque a Cantina consegue atender pessoas que estão mais atrás na fila, ou seja, que entraram há mais tempo. Nos outros dois casos a Cantina para de funcionar (acabam as iterações) antes que essas pessoas que estão muito atrás na fila sejam atendidas, assim, analisando somente o tempo médio, parece que os casos 1 e 3 são menos eficientes que o caso 2.

Os casos 1 e 3 são semelhantes, sendo que o caso 1 consegue atender um número um pouco maior de pessoas. O tempo médio do caso 3 é menor, pois este simula um dia em que poucas pessoas entram na fila. Os usuários atendidos neste caso aguardam menos tempo para o atendimento, por isso o tempo médio é o menor.

Conclusão

O caso 3 é o que atende menos usuários entre todos considerados, e provavelmente é o jeito mais barato de operar a cantina, já que menos bandejas são recarregas na média. Logo, esse seria o caso ideal para dias com baixa demanda.

O caso 2 parece pior que os outros porque tem o maior tempo médio, mas como ele consegue atender o maior número de clientes, seria o caso ideal para dias muito movimentados. O tempo médio de espera deste caso é o maior porque ele consegue atender pessoas que estão mais atrás na fila, sendo que nos outros casos considerados estas pessoas simplesmente não foram atendidas.

O caso 1 é bastante semelhante ao caso 3, porém com uma capacidade de atendimento um pouco maior, e deverá ser adotado em dias em que o movimento não é nem tão grande nem tão pequeno.