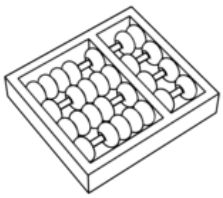

Instituto de Computação
Universidade Estadual de campinas

MC202 - Estruturas de Dados

Laboratório 4

Filas

Data de publicação: Quarta feira, 27 de abril de 2016
Prazo máximo de submissão: Sexta feira, 6 de maio de 2016 às 23h59m

Professor: Neucimar J. Leite <neucimar@ic.unciamp.br>

Monitores:
- Juan Hernández (PED) <juan.albarracin@students.ic.unicamp.br>
- Leonardo Yvens (PAD) <leoyvens@gmail.com>

Grupo do curso: https://groups.google.com/d/forum/mc202bc_2016s1
Sítio eletrônico da submissão do código: https://susy.ic.unicamp.br:9999/mc202bc

Enunciado

Simule uma cadeia produtiva simples de *n* elementos, com *k* operários, para determinar qual dentre eles termina o trabalho primeiro.

Descrição do problema

Considere uma fila de *n* insumos a serem processados manualmente por *k* operários. Cada insumo demanda uma quantidade diferente de tempo para se tornar um produto definitivo. Existe um tempo limite *m* em que o insumo pode ficar fora da fila, portanto, se ele demandar um tempo maior que *m* para ficar pronto, após *m* intervalos de tempo com o operário, deve voltar à fila para ser processado posteriormente. Assim que um operário terminar o produto ou retornar o insumo à fila, pega o insumo seguinte da mesma. Se na hora de considerar o produto seguinte a fila estiver vazia, considera-se que o operário terminou seu trabalho e pode folgar o resto do dia. Você deve indicar, dados os tempos de processamento dos *n* insumos, o valor *m* e o número de operários (k), qual ou quais operários (se houver empate) serão os primeiros a terminar seu trabalho.

Como fazer a simulação?

Três estruturas seriam necessárias:

1. A fila com os insumos. Neste caso, só basta armazenar a quantidade de tempo que cada insumo precisa.
2. Vetor dos tempos restantes de cada operário para terminar o produto.
3. Vetor dos tempos que cada produto leva fora da fila.

Os itens 2 e 3 são apenas sugestões. Fique à vontade para substituí-los por outras estruturas, se achar necessário. Todos os valores são inteiros. Cada iteração do programa pode ser considerada como um (1) instante de tempo. Assim, todos os valores do vetor do item 2 são decrementados de 1, enquanto os valores do vetor do item 3 são incrementados de 1.

Há dois eventos importantes:

- Um operário terminou o produto (um dos valores do vetor do item 2 atingiu 0). Neste caso, ele deve pegar o elemento no início da fila sem adicionar nada nela.
- O tempo fora da lista de um insumo atingiu *m* e ainda não foi terminado (um dos valores do vetor do item 3 atingiu *m*). Neste caso, o operário deve retornar o insumo, com o seu novo tempo restante para ser concluído, ao final da fila e pegar o elemento no início da mesma.

Como tratar eventos importantes que acontecem simultaneamente? Seguem as regras:

A. Para serem identificados, os operários devem ter um índice associado que vai de 0 a *k-1*.
B. Em cada iteração do programa:
    a. Primeiramente, todos os operários que precisam retornar insumos à fila fazem isto, antes de qualquer outro operário pegar um elemento da mesma.
    b. Após finalizado o item anterior, os operários que precisarem poderão começar a retirar elementos da fila.
    c. Após as etapas acima, procede-se à atualização dos valores dos vetores referentes ao percurso de tempo.
C. A ordem em que os operários retiram ou retornam elementos da fila é definida pelo índice associado. Exemplo: o operário *2* e *5* precisam retirar um elemento da fila, então o *2* retira primeiro, seguido pelo operário *5*.
D. As ações de retirar ou retornar um elemento à fila não consomem tempo.

É importante seguir estas regras pois caso contrário o resultado final pode variar.

Dica: observando o exemplo detalhado abaixo, você observará que nem todas as iterações têm eventos importantes. Neste caso, você poderá fazer com que a sua simulação pule as

iterações sem eventos importantes, obtendo os mesmos resultados. Estas melhorias no programa não influem na nota final da atividade.

Exemplo detalhado

Na tabela abaixo é mostrado o estado da fila e dos operários, em cada um dos três passos fundamentais à cada iteração. Todo operário é representado por um número que indica o tempo restante do insumo para se tornar um produto acabado, e um subscrito que indica o tempo restante para ele retornar à fila. Ao se pegar um elemento da fila, o subscrito associado é igual a *k*.

n = 5, k = 3, m = 4

Iteração	Estado da fila	Estado dos operários	Comentários
0	2 6 5 3 7	0 0 0	Estado inicial
1	Adicionar elementos à fila **(Nada é feito)** Retirar elementos da fila **3 7** Percurso do tempo **3 7**	Adicionar elementos à fila **(Nada é feito)** Retirar elementos da fila **2_4 6_4 5_4** Percurso do tempo **1_3 5_3 4_3**	Os operários retiram os *k* primeiros elementos da fila.
2	Adicionar elementos à fila **3 7** Retirar elementos da fila **3 7** Percurso do tempo **3 7**	Adicionar elementos à fila **1_3 5_3 4_3** Retirar elementos da fila **1_3 5_3 4_3** Percurso do tempo **0_2 4_2 3_2**	O tempo avança. A fila não é acessada em momento algum.
3	Adicionar elementos à fila **3 7** Retirar elementos da fila **7** Percurso do tempo **7**	Adicionar elementos à fila **0 4_2 3_2** Retirar elementos da fila **3_4 4_2 3_2** Percurso do tempo **2_3 3_1 2_1**	O primeiro operário termina um produto. Não adiciona nada à fila, e retira o insumo seguinte.
4	Adicionar elementos à fila **7**	Adicionar elementos à fila **2_3 3_1 2_1**	O tempo avança. A fila não é acessada em momento algum..

	Retirar elementos da fila **7** Percurso do tempo **7**	Retirar elementos da fila **2_3 3_1 2_1** Percurso do tempo **1_2 2_0 1_0**	
5	Adicionar elementos à fila **7 2 1** Retirar elementos da fila **1** Percurso do tempo **1**	Adicionar elementos à fila **1_2 0 0** Retirar elementos da fila **1_2 7_4 2_4** Percurso do tempo **0_1 6_3 1_3**	Os insumos do segundo e terceiro operário atingiram o tempo máximo permitido para ficar fora da fila e, portanto, são retornados a ela. Após retornar todos os insumos, dois deles são retirados da fila pelos mesmos operários.
6	**(Vazia)**	**0_3 5_2 0_2**	Apresentando os estados finais da iteração, para você testar o seu entendimento do problema.
7	Adicionar elementos à fila **(Vazia)** Retirar elementos da fila **(Condição de parada atingida)**	Adicionar elementos à fila **0 5_2 0** Retirar elementos da fila **0 5_2 0**	Cabe aqui a você explicar o que aconteceu nesta iteração.

Neste exemplo, as iterações que não têm eventos importantes são as iterações 2 e 4. Na iteração 7, o primeiro e terceiro operário terminaram de processar seus insumos e, ao retirar um insumo da fila, encontraram-na vazia. Isto significa que a resposta é **0** e **2**.

Especificação de entrada e saída

A entrada tem duas linhas. A primeira contem os valores *n*, *k* e *m* separados por espaço. A segunda linha contém *n* inteiros separados por espaço, representando o tempo que cada insumo requer para se tornar um produto final. A ordem dos números na segunda linha é igual a ordem que inicialmente têm os insumos na fila.

A saída deve ser uma linha contendo os índices dos operários que terminaram primeiro, separados por espaço.

Seguem alguns exemplos de entradas com suas respectivas saídas.

Entrada	**Saída**
5 3 4 2 6 5 3 7	0 2

10 3 4 5 10 15 20 25 30 35 40 45 50	1

Estrutura da submissão

O código fonte deve ser submetido no sistema SuSy para ser executado e testado. O sistema receberá três arquivos:

Nome	**Função**
main.c	Programa principal que lê os dados de entrada, faz as chamadas às funções e escreve a saída.
fila.h	Contém unicamente a declaração do TAD e de funções para operações em fila. Não será fornecida uma estrutura de arquivo, portanto você tem a liberdade de fazer a sua implementação para demonstrar seu domínio sobre o assunto envolvendo filas. Espera-se, no mínimo uma função de criação de fila, uma função de destruição da fila, uma função para adicionar um elemento à fila (*enqueue*) e uma função para remover um elemento da fila (*dequeue*).
fila.c	Implementação das funções declaradas em **fila.h**.

Observações

- A estrutura da simulação fornecida pode não ser seguida estritamente, desde que você tenha certeza de que a lógica implementada sempre conseguirá os mesmos resultados da lógica sugerida.

- É mandatório liberar memória dinamicamente alocada.

- Arquivos de teste serão fornecidos para a validação do programa. A avaliação no sistema será feita com esses e outros testes privados.

- O limite de submissões no SuSy é **15**. Recomenda-se executar de forma local seu programa, usando tanto os testes abertos quanto os exemplos aqui fornecidos.
  Submeta seu programa somente após haver testado localmente e de maneira bem sucedida todos os casos.

- A clareza do código e a sua documentação (por meio de comentários) serão avaliadas. O esforço dos monitores para entender o seu código deve ser mínimo. Com a documentação, você demonstra que compreendeu de maneira efetiva os conteúdos do curso referentes ao tema filas.

- Dúvidas podem ser esclarecidas nas aulas de laboratório ou no grupo do curso indicado no cabeçalho deste documento.

Critério de avaliação

$$nota = 7\frac{n_c}{n} + QD + AC$$

$$0 \leq QD \leq 1$$
$$0 \leq AC \leq 2$$

onde,

n	:	Número total de testes no SuSy
n_c	:	Número de testes corretos
QD	:	Qualidade do código
AC	:	Indicador de quanto o estudante tem demostrado dominar o tema de pilhas

A qualidade do código (*QD*) dependerá tanto da legibilidade quanto da documentação. Deve ser fácil para os monitores entender o que foi feito. Já o *AC* é uma nota que depende de quanto o estudante conseguiu convencer o revisor de que sabe implementar corretamente os TADs e operar com a estrutura em questão.

Considerações finais

- Embora haja várias maneiras de resolver os problemas indicados nos laboratórios, o estudante deve optar pela maneira que melhor exercite os conceitos trabalhados em sala de aula. **Não atender a esta advertência invalidará a parte da nota referente aos testes do SuSy**.

- Casos de plágio acarretam **média final zero na disciplina** para todos os envolvidos, sem exceção. O SuSy pode detectar casos de plágio no código, portanto evite compartilhar seu código com outros colegas, mesmo que seja apenas pequenos trechos.