Trabalho 7 – Fila de prioridade usando heap para simulação de atendimento

Data: 21/10/2013 até meia-noite
Dúvidas até: 09/10/2013
Faq disponível em: http://www2.icmc.usp.br/~mello/trabalho07.html

A estrutura de dados Heap e seus algoritmos são úteis para realizar ordenação de dados, como visto no algoritmo Heapsort, e também para implementar filas de prioridade eficientes. Uma das aplicações de filas de prioridade é a alocação de recursos, como no gerenciamento de memória e tempo de processamento realizado pelo sistema operacional. No entanto é possível aplicar o mesmo conceito a problemas não computacionais.

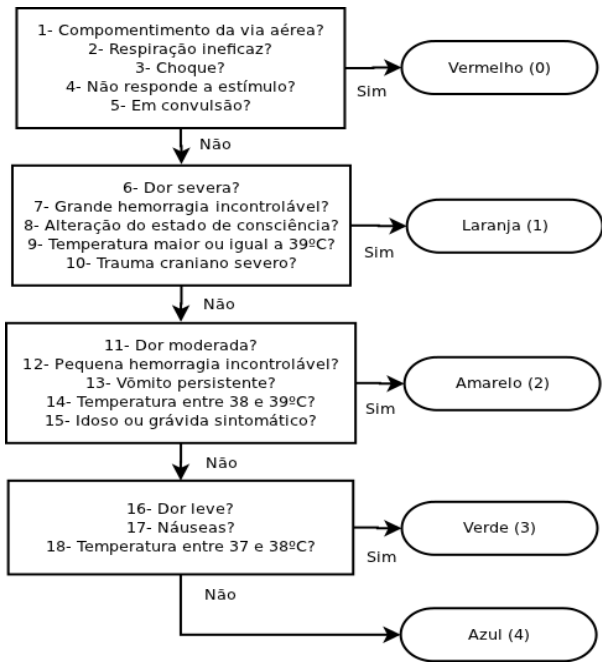
Leia a descrição do trabalho **com atenção e várias vezes**, anotando os pontos principais e as possíveis formas de resolver o problema. Comece a trabalhar o quanto antes para que as dúvidas sejam sanadas a tempo de você conseguir finalizar o trabalho no prazo.

Introdução

Numa unidade médica de urgência e emergência hospitalar, cada paciente a ser atendido recebe uma classificação de risco para definir a prioridade do atendimento. A prioridade é definida pelas cores vermelho (0), laranja (1), amarelo (2), verde (3), e azul (4), cada qual com um determinado tempo de espera tolerável, com os seguintes significados:

- Vermelho: quadro clínico implica em risco de morte, e que o caso deve ser rapidamente encaminhado para a sala de emergência. Atendimento imediato.
- Laranja e amarelo: o paciente não tem risco iminente de morte, mas o atendimento é prioritário, pois o tempo de espera pode aumentar a gravidade do caso. Atendimento com tempo de espera tolerável de – laranja: até 10 minutos, amarelo até 20 minutos.
- Verde: não há risco de morte e o paciente deverá ser atendido após os casos vermelhos, laranjas e amarelos. Atendimento com tempo de espera tolerável de até 60 minutos.
- Azul: quadros crônicos, sem sofrimento agudo. Neste caso, o paciente será encaminhado ao centro de saúde. Atendimento poderá ser realizado em até 180 minutos.

O protocolo para encontrar a prioridade é definido pelo esquema abaixo:



Tarefa

Implementar um programa que controle: i) a fila de pacientes a serem atendidos e ii) os pacientes em atendimento por médicos. Como entrada são oferecidos dados simulados do setor de urgência e emergência de um hospital, que precisa alocar as pessoas que chegam aos médicos de plantão.

Para cada pessoa que chega, uma prioridade será definida, e essa pessoa é alocada a um médico. Cada médico demora um certo tempo (dentro de um intervalo esperado) para atender um paciente, e por meio desse tempo o programa irá simular o tempo de atendimento e espera de pacientes.

A partir do tempo de espera tolerável de cada cor (descrito na introdução) é possível saber se há uma demanda maior do que a capacidade do hospital. Os tempos que os pacientes demoram para ser atendidos são, segundo suas cores:

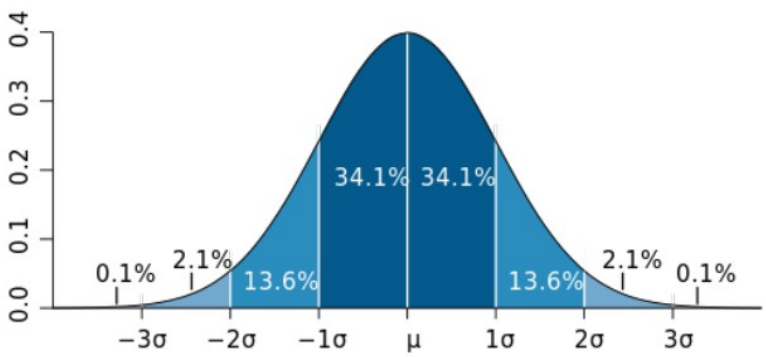
- 0 - Vermelho: 50 minutos em média com um desvio padrão de 10 minutos
- 1 - Laranja: 20 minutos em média com um desvio padrão de 5 minutos
- 2 - Amarelo: 15 minutos em média com um desvio padrão de 5 minutos
- 3 - Verde: 8 minutos em média com um desvio padrão de 2 minutos
- 4 - Azul: 5 minutos em média com um desvio padrão de 5 minutos

Esses tempos obedecem a distribuição de probabilidades Normal (http://pt.wikipedia.org/wiki/Distribuição_normal). Essa distribuição considera uma **média** e um **desvio padrão** ao redor dessa média para produzir valores aleatórios. Nessa distribuição há maior chance de um valor sorteado ser próximo da média do que distante da média. Isso é diferente da distribuição uniforme, que gera números aleatórios em um intervalo com igual probabilidade para todos os valores.

A média pode ser vista como o valor *central* da distribuição e o desvio padrão como a *dispersão* com relação a essa média. Por exemplo, se geramos um conjunto de *N* valores segundo uma distribuição Normal de média 10 e desvio padrão igual a 2, teremos 68,2% dos valores produzidos entre [média – desvio padrão, media + desvio padrão], i.e., [10-2, 10+2] = [8, 12].

Para outro caso, considere uma média igual a 2 com desvio padrão igual a 4. Neste caso, 68,2% dos valores produzidos estariam entre [2-4, 2+4] = [-2, 6]. Sendo assim, há casos que os tempos de atendimentos produzidos pela função de geração de números aleatórios serão negativos. Nesses casos, substitua o valor produzido por um custo fixo ***C***, dado como entrada.

Apesar de 68,2% dos valores produzidos pela distribuição Normal estarem entre [média - desvio, médio + desvio], há outros valores menos prováveis localizados mais abaixo e outros mais acima desse intervalo conforme a leitura http://pt.wikipedia.org/wiki/Distribuição_normal e conforme observado na figura abaixo:



Dica: O livro The Art of Computer Programming, Volume 2, seção 3.4.1 tem o algoritmo para a geração de números aleatórios segundo a distribuição Normal, utilizando o *método polar* que foi originalmente desenvolvido por Box, Muller e Marsaglia em 1958. Esse método gera números com base na distribuição de média 0 e desvio padrão 1. Para outras distribuições (com média e desvio padrão diferentes), ajustar multiplicando pelo desvio padrão e somando a média.

Funcionamento do programa

1) Entrada de dados

Iniciamente será definido pelo teclado o número *M* de médicos em plantão, seguido pelos registros de cada médico, no formato:

```
<M>
<Registro1> <Registro2> ... <RegistroM>
<custo_fixo>
```

Considere que o registro é um número inteiro entre 1 e 999999. O primeiro registro de médico deve ser usado como semente para geração de números aleatórios, ou seja, você deverá passar o valor do primeiro registro médico para a função **srand(), uma única vez, antes do início da chegada de pacientes.**

O valor <custo_fixo> será o valor a substituir caso o número sorteado esteja fora do intervalo desejado.

A seguir, cada grupo de pacientes a entrar na fila de atendimento será incluido por meio de um arquivo cujo nome será fornecido como entrada, por exemplo:

```
arq_pacientes_001.txt
```

O arquivo está em formato texto e possui a seguinte configuração para cada bloco de pacientes:

```
<T>
<N>
<Sobrenome_1> <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14> <15>
<16> <17> <18>
...
<Sobrenome_N> <1> <2> <3> <4> <5> <6> <7> <8> <9> <10> <11> <12> <13> <14> <15>
<16> <17> <18>
```

Onde <N> é o número de pacientes a chegar para atendimento num dado instante. O valor <T> é um número real positivo, representando o tempo passado desde a chegada do último grupo de pacientes.

<Sobrenome> é o sobrenome do paciente (assuma uma única palavra, sem separação por espaço) . E os números de <1> a <18> São respostas S (Sim) ou N (Não) a cada uma das 18 perguntas do protocolo de triagem especificado anteriormente.

O arquivo texto poderá ter um ou mais grupos de pacientes conforme exemplificado anteriormente.

2) Processamento e saída de dados

O sistema funciona com "chamadas" para atendimento. Cada chamada para atendimento é realizada no instante em que um grupo de pacientes dá entrada no pronto-socorro. Assim, os pacientes ficam sempre esperando até que um novo grupo chegue e então ocorre a liberação dos médicos e nova alocação de pacientes.

Assim, para cada grupo de pacientes lido do arquivo, o programa deverá calcular as prioridades, sortear o número aleatório com média e desvio padrão que deverá ser o tempo que os pacientes irão demorar para ser atendidos, e inserir cada paciente na fila de prioridade utilizando uma estrutura *heap*.

Após dar entrada em todos os pacientes na fila, os pacientes da fila serão atendidos em ordem de prioridade (da menor para a maior), com critério de desempate pela ordem de chegada. Assim, para cada médico disponível:

- Alocar um paciente na fila usando como ordem o registro profissional (alocar primeiro a médicos com número de registro menor).
- Caso *não* haja médicos disponíveis para atender a todos, o programa deverá percorrer a fila de espera, calcular o tempo de espera estimado dos pacientes na espera (naquele instante), em minutos, e gerar esse valor como saída. Esse tempo de espera estimado será o menor tempo esperado (ou seja o tempo médio para cada tipo de paciente) entre os pacientes sendo atendidos somado aos tempos de atendimento previstos (de acordo com a prioridade) para os pacientes que estão na frente na fila. A ordem de impressão é a ordem da fila, começando daquele que deverá ser atendido primeiro.
- O formato da saída do tempo de **espera** previsto em minutos é:
  E <Sobrenome> <Tempo>\n

Quando um grupo chegar e houver pacientes em atendimento, o programa deverá realizar uma simulação dos paciente já atendidos. Fazer a diferença do tempo restante para cada paciente ser atendido com o tempo <T> passado. Veja que alguns pacientes poderão ter tempo restante negativo.

Retirar do atendimento os pacientes já atendidos (que estão **saindo** do hospital), percorrendo a lista de médicos e gerando como saída o registro do médico, o sobrenome do paciente, e o tempo de espera, no formato:

S <Num_Registro> <Sobrenome> <Tempo_max_tolerancia> <Tempo_Espera_Real>/n

Para imprimir a saída, os pacientes devem ser exibidos por ordem da lista de médicos e liberando aqueles pacientes cujo tempo já tenha sido esgotado (ou seja, o tempo restante seja menor ou igual a zero).

O <Tempo_max_tolerancia> é aquele tempo definido pela cor respectiva <Tempo_Espera_Real> é o tempo que o paciente ficou esperando efetivamente para ser atendido.

A fila de espera e a lista de pacientes em atendimento só deverá ser processada a cada chegada de um grupo de pacientes, ou quando terminar o arquivo, ou seja, não há mais pacientes a chegar.

Em resumo, utilizar os seguintes passos:
a) processar a chegada do grupo de pacientes, inserindo cada um na fila de prioridades, já com sua prioridade calculada e tempo de atendimento sorteado
b) dado o tempo passado T (lido no início do grupo de pacientes), subtrair T do 'tempo restante' de todos os pacientes em atendimento (se houver), simulando o passar do tempo.
c) após subtrair, percorrer a lista de médicos, na ordem do registro, liberando e imprimindo a saída na tela de todos aqueles cujo 'tempo restante' seja menor ou igual a zero.

d) se a fila não for vazia, alocar novos pacientes aos médicos disponíveis
e) retornar ao passo (a) até que não haja mais pacientes a serem processados

Ao final, quando não houver mais pacientes a chegar, o programa deverá simular o atendimento até que a fila seja vazia. A partir desse ponto não é necessário exibir os tempos de espera previsto (saída do tipo 'E'), apenas os tempos de espera reais após atendimento (saídas do tipo 'S') .

Como não chegam mais pacientes, faremos agora um tratamento especial, pois agora não há mais passagem de tempo definida pelos grupos de pacientes. Assim, os médicos liberam um a um os pacientes. Isso é feito usando os seguintes passos:
a) verificar na lista de pacientes sendo atendidos qual é aquele cujo tempo restante é mínimo.
b) utilizar esse tempo mínimo obtido no passo (a), subtraindo-o do 'tempo restante' de todos os pacientes em atendimento, simulando o passar do tempo.
c) após subtrair, percorrer a lista de médicos, na ordem do registro, liberando e imprimindo a saída na tela de todos aqueles cujo 'tempo restante' seja menor ou igual a zero.
d) se a fila não for vazia, alocar novos pacientes aos médicos disponíveis
e) se a lista de pacientes sendo atendidos não for vazia, retornar ao passo (a)

Exemplo

Para esse exemplo suponha que o tempo de atendimento sorteado seja **sempre igual à média**.

Entrada:

```
2
101 102
1
arq_pacientes_001.txt
```

Conteúdo do arquivo texto:

```
0.0
3
Valle N N N N N N N N N N N N N N S S N
Silva N S N N N S N S N N N N N N S N N N
Sousa N N N N N N N N N S N N S S N N N
15.0
1
Costa N N N N N N N N S N N N N N N N N
22.5
2
Gomes N N N N N N N N N N N N N N N N S
Lima N N N N N N N N N N N N N N N N N
```

Saída:

```
E Valle 15.0
S 102 Sousa 20 0.0
E Valle 20.0
S 102 Costa 10 0.0
E Gomes 8.0
E Lima 16.0
S 102 Valle 60 37.5
```

```
S 101 Silva 0 0.0
S 102 Gomes 60 8.0
S 101 Lima 180 12.5
```

No exemplo acima, chegam 3 pacientes inicialmente, aqueles com maior prioridade foram atendidos diretamente e 1 ficou na espera, que nesse instante estava estimada em 15 minutos dado que um dos médicos poderia ficar livre nesse tempo (tempo médio) após atender o paciente prioridade 2.

Após 15 minutos um novo paciente chegou (Costa). Nesse instante, o paciente Sousa terminou de ser atendido tendo esperado 0 minutos.

O paciente Valle que já estava esperando teve seu tempo de espera estimado em mais 20 minutos dado que chegou um paciente com maior prioridade.

22,5 minutos depois, dois novos novos pacientes chegam para atendimento e o tempo estimado de espera foi de 8 minutos para um e 16 para o outro.

Ao terminarem as entradas, todos pacientes são atendidos na ordem e retirados, um a um, com seus tempos de espera reais exibidos.

Informações importantes

1. Um dos objetivos da disciplina de ICC2 é o **aprendizado individual** dos conceitos de programação. A principal evidência desse aprendizado está nos trabalhos, que são individuais neste curso. Você deverá desenvolver seu trabalho sem copiar trechos de código de outros alunos, nem codificar em conjunto. Portanto, compartilhem idéias, soluções, modos de resolver o problema, mas não o código.
    - Quando autores e copiadores combinam, estão fraudando o sistema de avaliação.
    - A fraude no sistema de avaliação fere o código de ética da USP.
    - O trabalho em grupo e a cooperação entre colegas é benéfica e útil ao aprendizado. Para ajudar um colega você pode lhe explicar estratégias e idéias Por exemplo, pode explicar que é preciso usar dois loops para processar os dados, ou que é melhor usar uma certa estrutura de dados, etc. O que você não deve fazer é mostrar o seu código. Mostrar/compartilhar o código pode prejudicar o aprendizado do seu colega:
        - depois de o seu colega ter visto o seu código, será muito mais difícil para ele imaginar uma solução original e própria;
        - o seu colega não entenderá realmente o problema: a compreensão passa pela prática da codificação e não pela imitação/cópia.
    - Um colega que tenha visto a sua solução pode eventualmente divulgá-la a outros colegas, deixando você numa situação muito complicada, por tabela.
    - O texto acima foi baseado e adaptado da página http://www.ime.usp.br/~mac2166/plagio/, da qual recomendo a leitura completa.

2. Todos os códigos fontes serão comparados por um (ou mais) sistema(s) de detecção de plágio, e **os trabalhos com alta similaridade detectada terão suas notas zeradas**, tanto aqueles relativos ao código de origem quanto do código copiado. Utilizamos para isso os códigos de todas as turmas, de diferentes anos.