Trabalho Prático 2 - Sistema de Escalonamento Logístico

1 Descrição do problema

Você foi contratado para automatizar o sistema logístico dos Armazéns Hanoi, uma empresa antiga originalmente fundada no Vietnam, parte do mesmo grupo que inventou a Torre de Hanoi.
O sistema logístico recebe pedidos de envio de pacotes, que devem ser transportados e armazenados entre os armazéns. O sistema deve rotear e acompanhar os pacotes durante esse processo. Um pacote chega inicialmente a qualquer armazém e é destinado a algum outro armazém da rede. Ao chegar a um armazém, o pacote recebe um número único e é calculada a sua rota em termos de armazéns. As ligações entre os armazéns podem ser vistas como um grafo, não direcionado, conhecido previamente e a rota pode ser vista como uma lista encadeada.
Os armazéns são organizados de acordo com o próximo destino de cada pacote que está lá armazenado. Assim, há uma seção no armazém para cada outro armazém ao qual ele esteja conectado. O funcionamento de cada seção é a peculiaridade dos Armazéns Hanoi: por questões de segurança, cada seção é uma sala na qual os pacotes são manipulados numa lógica LIFO Last-In First-Out. Ou seja, apenas o último pacote colocado na sala pode ser retirado imediatamente. Caso contrário, todos os pacotes que o sucederam no tempo tem que ser retirados antes que ele o seja.
Uma vez por dia pacotes são transportados de um armazém para o outro, de acordo com a sua rota, com prioridade para os pacotes que estão em trânsito há mais tempo. Há um limite do número de pacotes que podem ser transportados por dia entre dois armazéns.
O sistema logístico deve ser capaz de identificar gargalos, ou seja, pacotes que ficaram retidos em um armazém mais tempo do que o mínimo necessário. Basicamente, há dois parâmetros de tempo a serem considerados:

1. Tempo de transporte entre dois armazéns
2. Tempo de manipulação de pacote na respectiva sala do armazém.

Além destes parâmetros, também deve ser definida a topologia de conexão entre os armazéns e a capacidade do meio de transporte entre dois armazéns.
Você pode assumir que o tempo de transporte entre dois armazéns e a capacidade dos meios de transporte sejam idênticos.
A sua entrada consiste em um arquivo com as seguintes informações, uma por linha:

- Data hora da postagem
- Nome Remetente
- Nome Destinatário
- Tipo
- Armazém de origem
- Armazém de destino

E a saída deve conter, para cada pacote, os seguintes tempos:

- Tempo esperado de estadia
- Tempo armazenado
- Tempo em trânsito

2 Simulação de eventos discretos

Segundo a Wikipedia [1], a simulação de eventos discretos (SED) modela a operação de um sistema como uma sequência de eventos discretos no tempo. Cada evento ocorre em um determinado instante de tempo e marca uma mudança de estado no sistema. Entre eventos consecutivos, considera-se que o sistema não sofre mudança alguma, assim, a simulação pode saltar diretamente do instante de ocorrência de um evento para o próximo.
Esta forma de execução se contrasta com a simulação contínua, na qual a simulação acompanha continuamente a dinâmica do sistema ao longo do tempo, sem saltos discretos de um evento ao outro. Assim, na simulação contínua o tempo é quebrado em pequenos intervalos e o estado do sistema é avaliado de acordo com o que ocorre dentro de cada intervalo. Em contraponto, a simulação de eventos discretos não precisa simular cada fatia de tempo e, ao saltar de um evento ao outro, ela é usualmente executada com mais rapidez que sua correspondente simulação contínua.
Uma técnica conhecida para execução de simulações de eventos discretos é o "Método das três fases". Nesta abordagem, a primeira fase sempre avança o relógio para o próximo evento a ocorrer, respeitando a ordem cronológica de eventos (chamados de eventos do tipo A). A segunda fase é a execução de todos os eventos que incondicionalmente ocorrem no instante atual (chamados de eventos do tipo B). A terceira fase é a execução de todos os eventos que condicionalmente ocorrem no tempo atual (chamados eventos do tipo C). O método das três fases é um refinamento da abordagem baseada em eventos, na qual os eventos simultâneos são ordenados de modo a tornar mais eficiente o uso dos recursos computacionais. O método das três fases é utilizado por diversos pacotes comerciais de simulação, mas do ponto de vista do usuário, tais especificidades técnicas do método de execução são geralmente ocultas.

2.1 Exemplo

Um exercício comum para se entender como são construídos modelos de simulação de eventos discretos é a modelagem de um sistema de fila de atendimento, tal como a fila formada por clientes que chegam em um agência bancária e esperam por atendimento no caixa. Neste caso, as entidades do sistema são os clientes que buscam atendimento e os eventos são: a chegada de um novo cliente, o início do atendimento no caixa e o fim do atendimento (equivalente à saída do cliente do sistema). Os estados do sistema, que são passíveis de alteração pelos eventos anteriores, são: o número de clientes na fila de atendimento (um número inteiro entre 0 e n) e o estado do caixa (livre ou ocupado). As variáveis aleatórias que devem ser identificadas para modelar a componente estocástica do sistema são: o tempo entre chegadas sucessivas de clientes e o tempo de serviço no caixa de atendimento.

[1] https://pt.wikipedia.org/wiki/Simulação_de_eventos_discretos

2.2 Componentes

Além da lógica do que acontece quando ocorrem eventos no sistema, a simulação de eventos discretos ainda inclui os componentes descritos a seguir.

2.2.1 Estado

Cada estado do sistema é representado por um conjunto de variáveis que capturam as suas propriedades mais significativas. O comportamento dos estados ao longo do tempo, $S(t)$ pode ser matematicamente representado por uma função degrau cujos valores mudam em resposta à execução dos eventos discretos do próprio sistema.

2.2.2 Relógio

A simulação deve manter controle do tempo atual de simulação, independentemente da unidade de medida de tempo utilizada pelo sistema sendo modelado. Na simulação de eventos discretos, em contraponto à denominada simulação em tempo real, o relógio "salta´´ entre os instantes de ocorrência dos eventos, ou seja, de outra forma, o relógio avança para o instante de início do próximo evento enquanto a simulação é executada.

2.2.3 Lista de eventos

Usualmente, a simulação mantém ao menos uma lista de eventos pendentes, que representa os eventos que ainda devem ser executados. Um evento é descrito pelo seu instante de ocorrência no tempo e um tipo, indicando o código que será usado para simular o evento. É comum que o código do evento seja parametrizado e, neste caso, a descrição do evento também conterá parâmetros para o seu código de execução.
Quando os eventos são instantâneos, as atividades que se estendem ao longo do tempo são modeladas como sequências de eventos. Alguns ambientes de simulação permitem que o instante de execução de um evento possa ser especificado por um intervalo identificado pelo instante de início e fim do evento.
O conjunto de eventos aguardando por execução são usualmente organizados como uma fila com prioridade, ordenada cronologicamente. Isto é, independentemente da ordem com que os eventos são adicionados ao conjunto de eventos, eles são removidos em ordem estritamente cronológica. Diversos algoritmos genéricos de busca e ordenação de filas com prioridade se provaram eficazes para a simulação de eventos discretos, tais como o *splay tree*, *skip lists*, *calendar queues*, e *ladder queues*.
Tipicamente, os eventos são agendados para execução de modo dinâmico, ao longo da própria da simulação. Por exemplo, no exemplo do banco proposto anteriormente, o evento "chegada do cliente´´ no instante t poderia, caso a fila de clientes esteja vazia e o caixa livre neste instante, incluir a criação dos eventos subsequentes "saída do cliente´´ no instante $t + s$, onde s é um tempo gerado a partir da distribuição do tempo de serviço no caixa.

2.2.4 Estatísticas

Uma simulação normalmente estima as principais estatísticas de interesse do comportamento do sistema. No caso do banco, por exemplo, uma estatística de interesse seria o tempo médio de espera por atendimento dos clientes. Em um modelo de simulação, as métricas (como a do tempo médio de espera por atendimento) são geralmente obtidas por meio de médias de replicações do modelo. Cada replicação representa uma diferente

```
Inicializar Condição de Término para FALSO
Inicializar as variáveis de estado do sistema
Inicializar o Relógio (usualmente zero)
Agendar um evento inicial
Enquanto (a condição de término é FALSA)
    Definir o relógio para o instante do próximo evento
    Executar o próximo evento
    Se for o caso, agendar novo(s) evento(s)
    Remover o evento executado da lista de eventos
    Atualizar as estatísticas
Fim
Gerar relatórios de estatísticas
```

Figura 1: Pseudo-código de simulador discreto de eventos típico

execução completa do modelo. Para se avaliar a qualidade do valor obtido, são construídos Intervalos de confiança para as estatísticas estimadas.

2.2.5 Condições de finalização de execução

Teoricamente, uma simulação de eventos discretos poderia ser executada para sempre. Assim, o projetista do modelo de simulação deve decidir quando a simulação deve terminar. Tipicamente, as condições de finalização são "no instante t´´ ou "após o processamento de um número n de eventos´´ ou, mais geralmente, "quando a medida estatística X atingir o valor x´´.

2.2.6 Execução

Um simulador de eventos discretos possui a estrutura típica ilustrada na Figura 1.

3 Sistema de Escalonamento Logístico dos Armazéns Hanoi

Nesta seção vamos descrever o Sistema de Escalonamento Logístico dos Armazéns Hanoi a ser implementado no trabalho prático.

3.1 Tipos abstratos de dados

Para a simulação proposta, você precisa projetar e implementar 4 tipos abstratos de dados, descritos a seguir.

3.1.1 Pacote

O pacote, cujas informações de entrada são lidas de um arquivo, representa a trajetória do pacote nos vários armazéns. Além dessas informações de entrada, é importante armazenar todos os momentos quando o pacote foi de alguma forma alterado, assim como os intervalos de tempo que ele ficou parado. A partir dessas informações é que será possível realizar as estatísticas de funcionamento dos Armazéns Hanoi.
Para fins da simulação, um pacote pode estar em 14 estados:

1. Não foi postado

2. Chegada escalonada a um armazém (postagem ou transporte)
3. Chegou a um armazém mas não foi armazenado
4. Armazenado em um armazém
5. Alocado para transporte
6. Entregue

O TAD pacote, além de manter os dados do pacote e sua rota, deve armazenar o estado atual, e as estatísticas de envio e armazenamento, em particular o tempo armazenado e o tempo sendo transportado. Essas informações serão fundamentais para o cálculo das estatísticas gerais do sistema.

3.1.2 Armazém

O TAD armazém mantém as informações de armazenamento. Para cada seção, é mantida uma pilha que se destina a manter os pacotes armazenados. Assim, é importante implementar as operações armazena e recupera pacote. Importante pensar na contabilização do tempo necessário para recuperar um pacote.

3.1.3 Transporte

O TAD transporte mantém as informações dos pacotes sendo transportados entre armazéns. Importante ressaltar que, como o transporte tem tempo constante, ele na prática é implementado através do escalonamento da chegada dos pacotes.

3.1.4 Escalonador

O escalonador é um elemento central da simulação de eventos discretos. Ele é implementado como uma fila de prioridade que recupera o próximo evento (ou seja o evento de menor data-hora que está na fila). Sugere-se implementar a fila de prioridade utilizando um *minheap*.
As operações a serem implementadas incluem:

1. Inicializa
2. InsereEvento
3. RetiraProximoEvento
4. Finaliza

Para fins de escalonamento, você pode converter as várias data-hora para o número de horas, incluindo frações, a partir de uma data de referência. As estatísticas de escalonamento devem ser geradas quando finalizar.

3.2 Arquitetura do sistema

Nesta seção vamos descrever a arquitetura do sistema a ser implementado. O sistema de simulação possui um escalonador, que controla a execução das várias tarefas de logística. A Figura 2 apresenta uma adaptação do pseudo-código genérico de simulação de eventos discretos para o caso do Armazéns Hanoi.

```
Inicializa Condição de Término para FALSO
Inicializa as variáveis de estado do sistema
Inicializa o Relógio (usualmente zero)
Para cada ligação entre dois armazéns
    Escalona evento de transporte
Para cada pacote a ser transportado
    Calcula e armazena a rota do pacote
    Escalona a chegada de pacotes nos armazéns de postagem
Enquanto houver eventos ou seções não vazias
    Remove o próximo evento do escalonador
    Avança o relógio para o instante do próximo evento
    Se evento é transporte
        Se há pacotes na seção associada
            Remove os pacotes mais antigos até a capacidade do transporte
            Escalona a chegada dos pacotes removidos no próximo armazém
        Escalona o próximo evento de transporte
    Se evento é chegada de pacotes
        Se pacote chegou ao destino final
            Registra entrega de pacote
        Senão chegou ao destino final
            Armazena o pacote na respectiva seção
    Atualizar as estatísticas
Fim
Gerar relatórios de estatísticas
```

Figura 2: Pseudo-código do Simulador dos Armazéns de Hanoi

3.3 Simulação de envios de pacotes

A simulação do envio de um pacote, como mencionado, começa com a inserção do evento de chegada do pacote no armazém de origem. Para tal é inserido um evento no escalonador para cada pacote no arquivo de entrada, com execução no tempo de chegada do pacote.
Um primeiro tipo de evento é a chegada de pacotes a um armazém. O escalonador insere o pacote na seção associada ao próximo armazém da sua rota. Essa inserção é instantânea e deve ser registrado o tempo de inserção, permitindo o cálculo do tempo de armazenamento quando o pacote for colocado no transporte.
Um evento que é tratado em paralelo é o transporte de pacotes. Inicialmente a rota de cada pacote é calculada a partir do armazém de origem e do armazém de destino, e não muda ao longo do seu envio. Importante ressaltar que, para grafos não direcionados e não ponderados, como é o caso, o menor caminho entre dois vértices pode ser calculado simplesmente aplicando uma busca em profundidade a partir do vértice de origem até o vértice destino.

4 Como será feita a entrega

4.1 Submissão

A entrega do TP2 compreende duas submissões:

VPL TP2: Submissão do código a ser submetido até **16/06, 23:59** (o sistema vai ficar aberto madrugada adentro, mais para evitar problemas transientes de infraestrutura). **Submissões em atraso terão desconto.** Detalhes sobre a submissão do código são apresentados na Seção 4.3.

Relatório TP2: Arquivo PDF contendo a documentação do TP, assim como a avaliação experimental, conforme instruções, a ser submetido até **16/06, 23:59** (o sistema vai ficar aberto madrugada adentro, mais para evitar problemas transientes de infraestrutura). **Submissões em atraso terão desconto.** Detalhes sobre a submissão de relatório são apresentados na Seção 4.2.

4.2 Documentação

A documentação do trabalho deve ser entregue em formato **PDF** e também **DEVE** seguir o modelo de relatório que será postado no `minha.ufmg`. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

1. **Capa:** Título, nome, e matrícula.
2. **Introdução:** Contém a apresentação do contexto, problema, e qual solução será empregada.
3. **Método:** Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
4. **Análise de Complexidade:** Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
5. **Estratégias de Robustez:** Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.

6. **Análise Experimental:** Apresenta os experimentos realizados em termos de desempenho computacional[2], assim como as análises dos resultados.

7. **Conclusões:** A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.

8. **Bibliografia:** Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.

9. Número máximo de páginas incluindo a capa: 10

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

A documentação deverá ser entregue como uma atividade separada designada para tal no minha.ufmg. A entrega deve ser um arquivo `.pdf`, nomeado `nome_sobrenome_matricula.pdf`, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais.

4.3 Código

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é terminantemente vetado. Você DEVE utilizar a estrutura de projeto abaixo junto ao Makefile:

```
– TP
    |– src
    |– bin
    |– obj
    |– include
    Makefile
```

A pasta **TP** é a raiz do projeto; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (headers) do projeto, com extensão *.h, por fim as pastas **bin** e **obj** devem estar vazias. O Makefile deve estar na raiz do projeto. A execução do Makefile deve gerar os códigos objeto *.o no diretório **obj** e o executável do TP no diretório **bin**. O arquivo executável **DEVE** se chamar **tp3.out** e deve estar localizado na pasta **bin**. O código será compilado com o comando:

```
make all
```

O seu código será avaliado através de uma **VPL** que será disponibilizada no moodle. Você também terá à disposição uma VPL de testes para verificar se a formatação da sua saída está de acordo com a requisitada. A VPL de testes não vale pontos e não conta como trabalho entregue. Um pdf com instruções de como enviar seu trabalho para que ele seja compilado corretamente estará disponível no Moodle.

[2]Para este trabalho não é necessário analisar a localidade de referência.

5 Avaliação

- Corretude na execução dos casos de teste - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Conteúdo segundo modelo proposto na seção **Documentação**, com as seções detalhadas corretamente - (20% da nota total)
- Definição e implementação das estruturas de dados e funções - (10% da nota total)
- Apresentação da análise de complexidade das implementações - (10% da nota total)
- Análise experimental - (25% da nota total)
- Aderência completa às instruções de entrega - (5% da nota total)

Se o programa submetido **não compilar**[3], seu trabalho não será avaliado e sua nota será **0**. Trabalhos entregues com atraso sofrerão **penalização de** 2^{d-1} pontos, com d = dias úteis de atraso.

6 Considerações finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia atentamente o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é crime. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na seção de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos.**

7 FAQ (*Frequently asked Questions*)

1. Posso utilizar qualquer versão do C++? NÃO, o corretor da VPL utiliza C++11.
2. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM, porém lembre-se que a correção é feita sob o sistema Linux, então certifique-se que seu trabalho está funcional em Linux.
3. Posso utilizar alguma estrutura de dados do C++ do tipo Queue, Stack, Vector, List, etc? NÃO.
4. Posso utilizar smart pointers? NÃO.

[3]Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

5. Posso utilizar o tipo String? SIM.
6. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO.
7. Posso utilizar alguma biblioteca para tratar exceções? SIM.
8. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
9. As análises e apresentação dos resultados são importantes na documentação? SIM.
10. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO.
11. Posso fazer o trabalho em dupla ou em grupo? NÃO.
12. Posso trocar informações com os colegas sobre os fundamentos teóricos do trabalho? SIM.
13. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.