Trabalho Prático 2 - Sistema de Escalonamento Hospitalar

1 Descrição do problema

Você foi contratado para implementar e otimizar um sistema de filas para o Hospital Zambs (HZ). Esse sistema deve acompanhar o paciente desde a sua admissão ao HZ até a sua alta hospitalar. O atendimento de emergência do HZ é organizado em três estágios. No primeiro estágio, triagem, é determinado o grau de urgência da condição clínica do paciente:

Verde: Paciente não corre risco de vida

Amarelo: Paciente tem uma condição grave

Vermelho: Paciente tem que ser atendido imediatamente, pois corre risco de vida iminente.

A partir da triagem o paciente é admitido ao serviço de emergência e atendido pelos profissionais de saúde pertinentes. Após o primeiro atendimento, o paciente pode ser internado ou não. Se ele não for internado, ele recebe alta ou é transferido para outra instituição. Se internado, ele prossegue com os tratamentos indicados. De forma genérica, temos 4 tipos de tratamentos:

1. Medidas hospitalares (p.ex., medir pressão e temperatura corpórea).
2. Testes de laboratório
3. Exames de imagem
4. Uso de instrumentos e aplicação de medicamentos

Para avaliar sua proposta, você recebe como entrada um arquivo com dados de pacientes atendidos, um por linha, no seguinte formato:

```
0009600024 0 2017 3 21 6 0 7 15 5 38
```

A Tabela 1 apresenta o nome e a descrição de cada atributo do paciente.

Atributo	Descrição
Identificador de paciente	Identificador único de paciente
Alta após atendimento	Alta após atendimento (0 - Não teve alta, 1-Teve alta)
Ano	Ano de admissão
Mês	Mês de admissão
Dia	Dia de admissão
Hora	Hora de admissão
Grau de urgência	0-Verde, 1-Amarelo e 2-Vermelho
#medidas hospitalares	Número de medidas hospitalares realizadas
#testes de laboratório	Número de testes de laboratória realizados
#exames de imagem	Número de exames de imagem realizados
#instrumentos/medicamentos	Número de instrumentos/medicamentos aplicados

Tabela 1: Descrição dos atributos dos pacientes

Este trabalho prático consiste em simular o funcionamento do HZ, com foco no tempo que cada paciente fica no hospital, entre a admissão na triagem e a alta, que denominaremos T_p, na utilização dos vários recursos e sua relação custo benefício.

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.

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.

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

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

```
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

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 Hospitalar

Nesta seção vamos descrever o Sistema de Escalonamento Hospitalar a ser implementado no trabalho prático.

3.1 Formato do arquivo de entrada

O arquivo de entrada para a sua simulação possui dois tipos de informação: (1) especificação da arquitetura a ser simulada, em termos de tempos de serviço e número de unidades que prestam o serviço; e (ii) lista de pacientes e respectivas características de atendimento. A Figura 2 apresenta a especificação do arquivo de entrada. Todos os campos são numéricos.

3.2 Tipos abstratos de dados

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

3.2.1 Paciente

O paciente, cujas informações de entrada são lidas de um arquivo, representa a trajetória de atendimento e serviços prestados pelo hospital. Além dessas informações de entrada, é importante armazenar todos os momentos quando o paciente foi de alguma forma atendido,

```
<tempotriagem> <numerotriagem>
<tempoatendimento> <numeroatendimento>
<tempomh> <numeromh>
<tempotl> <numerotl>
<tempoei> <numeroei>
<tempoim> <numeroim>
<numeropacientes>
<id_1> <alta_1> <ano_1> <mes_1> <dia_1> <hora_1> <grau_1> <mh_1> <tl_1> <ei_1> <im_1>
...
<id_n> <alta_n> <ano_n> <mes_n> <dia_n> <hora_n> <grau_n> <mh_n> <tl_n> <ei_n> <im_n>
```

Figura 2: Especificação do arquivo de entrada

assim como os intervalos de tempo que ele ficou em cada uma das filas. A partir dessas informações é que será possível realizar as estatísticas de atendimento do hospital e avaliar cenários de alocação de recursos e tempos para os procedimentos.
Para fins da simulação, um paciente pode estar em 14 estados:

1. Não chegou ainda ao hospital
2. Na fila de triagem
3. Sendo triado
4. Na fila de atendimento
5. Sendo atendido
6. Na fila de medidas hospitalares
7. Realizando medidas hospitalares
8. Na fila de testes de laboratório
9. Realizando testes de laboratório
10. Na fila de exames de imagem
11. Realizando exames de imagem
12. Na fila para instrumentos/medicamentos
13. Sendo aplicados instrumentos/medicamentos
14. Alta hospitalar

Note que há uma simplificação importante na simulação, o paciente assume os estados na ordem apresentada e não há situação em que ele retorne a um estado anterior. O estado "Não chegou ainda ao hospital´´ é o estado inicial de todos os pacientes e ele se encerra na data-hora de admissão atribuída a cada paciente no arquivo de entrada. Um outro caso a ser tratado é de quando o paciente tem alta logo após o atendimento, indo de "Sendo atendido´´ para "Alta hospitalar´´.
O TAD paciente, além de manter o prontuário do paciente (quais e quantos procedimentos deve realizar), deve armazenar o estado atual, e as estatísticas de atendimento do paciente, em particular o tempo ocioso e o tempo sendo atendido. Essas informações serão fundamentais para o cálculo das estatísticas gerais do sistema.

3.2.2 Procedimento

Procedimentos são todos os serviços à disposição de um paciente. Para cada serviço é definido um tempo médio de execução e o número de unidades que executam o serviço. Por exemplo, se é definido que `<numerotriagem>` é 5, isso significa que há 5 atendentes para a triagem, o que permite atender até 5 pacientes simultaneamente. Para fins da simulação, um serviço, na prática cada uma das suas unidades, pode estar ocupado ou ocioso.
O TAD procedimento deve registrar a utilização de cada unidade de cada procedimento e/ou seu tempo ocioso. Também é importante manter a data hora até a qual cada unidade está ocupada.

3.2.3 Fila

O TAD fila tem por objetivo controlar os pacientes aguardando para cada procedimento e, se for o caso, em cada prioridade. Idealmente você deve implementar um único TAD que será instanciado múltiplas vezes, um para cada fila a ser simulada.
As operações a serem suportadas pelo TAD são as tradicionais:

1. Inicializa
2. Enfileira
3. Desenfileira
4. FilaVazia
5. Finaliza

Em termos de estatísticas, é importante registrar o tempo que um dado paciente ficou na fila e o tamanho da fila em cada intervalo de tempo, com vistas a calcular a contenção pelo procedimento. As estatísticas devem ser geradas quando da finalização da fila.

3.2.4 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 devem ser geradas quando finalizar.

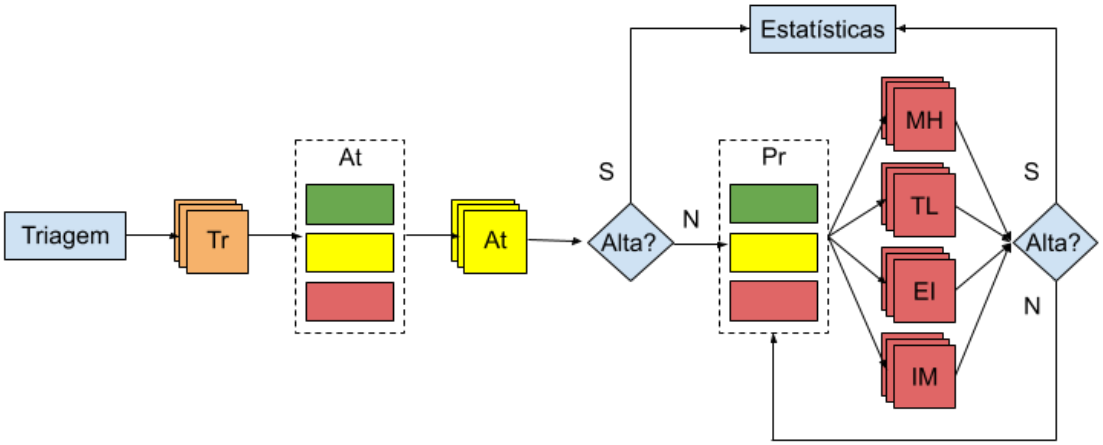

Figura 3: Sistema de simulação HZ

3.3 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 de procedimentos, e várias filas, que mantem os pacientes que estão esperando atendimento.
A Figura 3 apresenta o sistema a ser simulado. Importante mencionar que os pacientes são divididos em 3 graus de urgência (p.ex., vermelho, amarelo e verde), cada um com uma fila de espera diferente.
A Figura 4 apresenta uma adaptação do pseudo-código genérico de simulação de eventos discretos para o caso do Hospital HZ.

3.4 Simulação de atendimentos

A simulação de um atendimento, como mencionado, começa com a inserção do evento de chegada do paciente ao hospital. Para tal é inserido um evento no escalonador para cada paciente no arquivo de entrada, com execução no tempo de chegada do paciente.
A simulação começa pela retirada do primeiro evento de chegada, associado ao primeiro paciente que chegou ao hospital. O escalonador então insere o paciente na fila de triagem e verifica se alguma fila tem algum paciente a ser escalonado e, havendo paciente e unidade do serviço desejado disponível, esse paciente deve ser escalonado na unidade. Esse escalonamento deve seguir os seguintes passos:

1. registra o tempo que a unidade a ser utilizada ficou ociosa;
2. registra o tempo que o paciente ficou esperando na fila;
3. calcula a duração do serviço a ser executado;
4. define o tempo do evento, que é o tempo atual somado ao tempo do serviço a ser executado, ou seja, o tempo de término do serviço; e
5. insere o evento no escalonador.

Após verificar todas as filas e respectivas unidades, o processo inicia novamente pela obtenção do próximo evento. Duas observações importantes. A primeira é que pode

```
Inicializa Condição de Término para FALSO
Inicializa as variáveis de estado do sistema
Inicializa o Relógio (usualmente zero)
Escalona a chegada de pacientes
Enquanto houver eventos ou filas não vazias
    Remove o próximo evento do escalonador
    Avança o relógio para o instante do próximo evento
    Verifica o próximo estado do paciente
    Se houver mais serviços para paciente
        Enfileira o paciente na fila adequada
    Senão
        Finaliza o atendimento (Alta Hospitalar)
    Para cada fila de espera que tenha pacientes
        Se há unidade disponível para execução
            Escalonar o evento de execução do serviço
    Atualizar as estatísticas
Fim
Gerar relatórios de estatísticas
```

Figura 4: Pseudo-código do Simulador do Hospital HZ

Procedimento	Tempo
Triagem	12 min (0.2 horas)
Atendimento	30 min (0.5 horas)
Medidas	6 min (0.1 horas)
Testes	3 min (0.05 horas)
Imagem	30 min (0.5 horas)
Instrumentos/Medicamentos	3 min (0.05 horas)

Tabela 2: Latência típica para procedimentos

haver uma prioridade entre as filas, e as filas de maior prioridade devem ser verificadas primeiro. A segunda é que, se todas as unidades que servem uma fila estiverem ocupadas, naturalmente o paciente irá esperar. Uma vez que o evento tenha sido executado (ou seja, tenha sido atingido o momento de término dele), ele será o próximo evento do escalonador. A Tabela 2 apresenta custos típicos para os vários procedimentos.

3.5 Exemplo

Considerando o paciente exemplo:

```
0009600024 0 2017 3 21 6 0 7 15 5 38
```

e os tempos apresentados na Tabela 2, teríamos o cálculo do seu tempo mínimo de permanência apresentada na Tabela 3. Esse é o tempo mínimo porque ainda é necessário contabilizar o tempo em fila(s) de espera.
Vamos agora ilustrar a simulação através de um exemplo completo. Seja o arquivo de entrada exemplo apresentado na Figura 5. Importante ressaltar que as primeiras 6 linhas

Procedimento	Tempo	Quant.	Total
Triagem	12 min (0.2 horas)	1	0.20
Atendimento	30 min (0.5 horas)	1	0.50
Medidas	6 min (0.1 horas)	7	0.70
Testes	3 min (0.05 horas)	15	0.75
Imagem	30 min (0.5 horas)	5	2.50
Instrumentos/Medicamentos	3 min (0.05 horas)	38	1.90
Tempo de Permanência Mínimo			6.55

Tabela 3: Tempo de permanência mínimo para paciente exemplo

```
0.2 2
0.5 2
0.1 2
0.05 2
0.5 2
0.05 2
10
0009600008 0 2017 3 21 2 1 5 43 2 110
0009600009 0 2017 3 21 2 0 3 1 5 21
0009600010 0 2017 3 21 2 1 3 4 2 8
0009600011 0 2017 3 21 2 2 15 28 4 68
0009600012 0 2017 3 21 2 0 3 2 7 9
0009600013 1 2017 3 21 3 0 2 2 2 8
0009600015 1 2017 3 21 3 0 1 3 1 4
0009600016 0 2017 3 21 3 2 5 14 1 28
0009600017 1 2017 3 21 4 0 1 0 0 0
0009600018 0 2017 3 21 4 1 2 8 3 11
```

Figura 5: Arquivo de entrada exemplo

contem as especificações do HZ, seguidas do número de pacientes e das informações dos pacientes, uma linha por paciente.
A saída esperada para o programa é apresentada na Figura 6. Por exemplo, a primeira linha traz informações do paciente `9600008`, que foi admitido em `Tue Mar 21 02:00:00 2017` e teve alta em `Tue Mar 21 12:33:00 2017`, tendo permanecido no HZ por `10.55` horas, das quais `9.85` horas em atendimento e `0.70` horas em espera.

4 Pontos extras

O sistema implementado pode ser otimizado de várias formas, que serão valoradas como pontos extras:

Fragmentação de procedimentos: O sistema original simula a execução de todos os procedimentos de um dado tipo em bloco. Isso pode aumentar a contenção e afetar o tempo de atendimento de outros pacientes. Uma otimização é fragmentar os procedimentos, por exemplo em blocos que estejam limitados a um tamanho pré-definido, ou mesmo determinado dinamicamente.

```
9600008 Tue Mar 21 02:00:00 2017 Tue Mar 21 12:33:00 2017 10.55  9.85  0.70
9600009 Tue Mar 21 02:00:00 2017 Tue Mar 21 06:36:00 2017  4.60  4.60  0.00
9600010 Tue Mar 21 02:00:00 2017 Tue Mar 21 05:06:00 2017  3.10  2.60  0.50
9600011 Tue Mar 21 02:00:00 2017 Tue Mar 21 12:27:00 2017 10.45  9.00  1.45
9600012 Tue Mar 21 02:00:00 2017 Tue Mar 21 08:39:00 2017  6.65  5.05  1.60
9600013 Tue Mar 21 03:00:00 2017 Tue Mar 21 04:12:00 2017  1.20  0.70  0.50
9600015 Tue Mar 21 03:00:00 2017 Tue Mar 21 03:42:00 2017  0.70  0.70  0.00
9600016 Tue Mar 21 03:00:00 2017 Tue Mar 21 07:27:00 2017  4.45  3.80  0.65
9600017 Tue Mar 21 04:00:00 2017 Tue Mar 21 04:42:00 2017  0.70  0.70  0.00
9600018 Tue Mar 21 04:00:00 2017 Tue Mar 21 13:00:00 2017  9.00  3.35  5.65
```

Figura 6: Arquivo de saída exemplo

Ordem de execução de procedimentos: O sistema original executa os procedimentos sempre na mesma ordem, o que pode ser ineficiente e causar contenção. Uma estratégia é mudar a ordem dos procedimentos dependendo da demanda pelos serviços. Por exemplo, quando for escalonar os procedimentos de um paciente, insere na fila que tenha a menor expectativa de tempo de espera.

Outras propostas de otimização são possíveis e serão valoradas. É necessário justificar, quantificar e entender os eventuais ganhos de uma proposta de otimização para receber os pontos extras.

5 Como será feita a entrega

5.1 Submissão

A entrega do TP1 compreende duas submissões:

VPL TP1: Submissão do código a ser submetido até **13/01, 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 5.3.

Relatório TP1: Arquivo PDF contendo a documentação do TP, assim como a avaliação experimental, conforme instruções, a ser submetido até **13/01, 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 5.2.

5.2 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.

5.3 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:

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

```
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.

6 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.

7 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.**

[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.

8 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.
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.