Trabalho Prático 2 - Sistema de Escalonamento Hospitalar

[redacted]

19 de janeiro de 2025

Conteúdo

1 Introdução

A gestão eficiente do fluxo de pacientes em hospitais é fundamental para reduzir o tempo de espera e otimizar recursos. O escalonamento hospitalar, especialmente em ambientes com alta demanda e recursos limitados, é um problema desafiador e de grande relevância prática.

Este trabalho desenvolve um sistema de simulação de eventos discretos para modelar o escalonamento hospitalar no Hospital Zambs (HZ). O objetivo é minimizar o tempo de permanência dos pacientes e equilibrar o uso de recursos, utilizando modelagem de filas e gerenciamento eficiente de eventos. A análise apresentada avalia o impacto das configurações de recursos nos tempos de espera, destacando gargalos e propondo melhorias operacionais.

2 Método

O projeto foi implementado em *C++*, utilizando programação orientada a objetos e abstrações clássicas de estruturas de dados para modelar o funcionamento do sistema hospitalar. As principais estruturas implementadas incluem TADs para pacientes, procedimentos, filas e um escalonador, que utilizam internamente estruturas clássicas para garantir eficiência e simplicidade.

Dentre essas estruturas clássicas, destacam-se:

- **MinHeap**: Implementado no TAD *Escalonador*, é utilizado para gerenciar os eventos do sistema em ordem cronológica, garantindo que o próximo evento seja processado com eficiência ($O(\log n)$).
- **Filas**: A estrutura *Fila* utiliza listas encadeadas para gerenciar pacientes de forma dinâmica. Cada fila é associada a um procedimento, com suporte a níveis de prioridade para otimizar a alocação de recursos.
- **Array Dinâmico**: Empregado na estrutura *Procedimento*, gerencia as unidades disponíveis, registrando ocupação, tempos de ociosidade e alocação de pacientes. Essa estrutura permite acessar diretamente os recursos de cada procedimento.

O escalonador centraliza a lógica do sistema, sendo responsável por:

1. Processar eventos do *MinHeap*.
2. Coordenar o fluxo de pacientes entre filas e procedimentos, respeitando as prioridades de cada fila.
3. Atualizar os tempos de espera nas filas e os tempos de ociosidade nas unidades.

A interação entre essas estruturas garante a eficiência do sistema. O *MinHeap* ordena eventos de maneira cronológica, enquanto as filas priorizam o atendimento de pacientes. Unidades ociosas são reutilizadas de forma dinâmica, otimizando a execução do programa.

3 Análise de Complexidade

O algoritmo de simulação envolve tanto a análise de tempo quanto de espaço, considerando as principais estruturas utilizadas. A seguir, detalhamos essas complexidades.

3.1 Análise de Tempo

3.1.1 Etapas do Processo

1. A remoção do próximo evento do heap reorganiza a estrutura em $O(\log n)$.
2. A liberação de pacientes de procedimentos e a alocação em filas são realizadas em $O(1)$ por operação.
3. A realocação de pacientes para procedimentos disponíveis também ocorre em $O(1)$.
4. A etapa final de exibição dos relatórios ordena os elementos usando *heapsort*, com custo $O(n \log n)$.

3.1.2 Complexidade Total

O laço principal processa m eventos, cada um com custo dominante $O(\log n)$, resultando em $O(m \log n)$. Com a adição da etapa de ordenação de relatórios ($O(n \log n)$), a complexidade geral é:

$$O(m \log n + n \log n).$$

Se o número de eventos (m) for muito maior que o número de relatórios (n), o custo final pode ser aproximado por:

$$O(m \log n).$$

3.2 Resumo do Funcionamento do Heap

O heap organiza os eventos cronologicamente em forma de uma árvore binária, garantindo que o menor tempo esteja no topo. Operações de inserção e remoção reorganizam a estrutura em $O(\log n)$, enquanto o acesso ao menor elemento é imediato ($O(1)$). Essa eficiência é fundamental para o escalonador.

3.3 Análise de Espaço

3.3.1 Estruturas de Dados

- **MinHeap:** O heap armazena m eventos, cada um contendo o tempo, tipo de evento e um ponteiro para o paciente. O espaço ocupado pelo heap é proporcional ao número de eventos:

$$O(m).$$

- **Filas:** Cada fila de prioridade (3 por procedimento) mantém ponteiros para os pacientes. Considerando k filas com capacidade c cada, o espaço é:

$$O(k \cdot c).$$

- **Procedimentos:** Cada procedimento armazena u unidades, com um ponteiro para o paciente alocado e variáveis auxiliares. O espaço para p procedimentos é:

$$O(p \cdot u).$$

3.3.2 Complexidade Total

Somando as principais estruturas:

$$O(m) + O(k \cdot c) + O(p \cdot u).$$

Em um cenário típico, $m \gg k \cdot c$ e $m \gg p \cdot u$, portanto, o uso de espaço é dominado pelo heap, resultando em:

$$O(m).$$

3.4 Conclusão

A análise confirma que o algoritmo é eficiente tanto em tempo ($O(m \log n)$) quanto em espaço ($O(m)$), sendo escalável para grandes volumes de dados. O uso de heap e filas garante o processamento ordenado dos eventos e a alocação eficiente dos pacientes nos procedimentos.

4 Estratégias de Robustez

Para garantir a confiabilidade e a resiliência do programa desenvolvido, diversas estratégias foram adotadas na implementação:

4.1 Validação de Entradas

O programa verifica o formato e os dados do arquivo *CSV*, assegurando que cada linha contém os campos esperados. Entradas inválidas, como tempos negativos ou dados incompletos, são identificadas e tratadas de forma apropriada, interrompendo a execução e exibindo mensagens de erro claras.

4.2 Tratamento de Erros

Blocos `try-catch` são usados para lidar com falhas como alocação dinâmica, manipulação de arquivos e operações em estruturas de dados. Caso ocorra uma exceção, o programa registra a falha em um arquivo de log e finaliza de maneira controlada, preservando a consistência do estado interno.

4.3 Programação Defensiva

A implementação verifica os limites das filas, procedimentos e unidades disponíveis antes de processar eventos. Operações como alocar pacientes em unidades ou adicionar eventos ao escalonador garantem que recursos sejam acessados apenas quando disponíveis, evitando erros como `segmentation faults`.

4.4 Checagem de Memory Leaks

O gerenciamento de memória dinâmica foi cuidadosamente projetado, com alocação e desalocação explícitas em todas as estruturas usadas (como *heaps*, filas e arrays dinâmicos). Ferramentas como **Valgrind** foram empregadas para identificar possíveis vazamentos de memória (*memory leaks*), garantindo que todos os recursos alocados sejam liberados corretamente ao final da execução.

4.5 Monitoramento e Rastreamento

O programa registra informações importantes em arquivos de log, permitindo monitorar o comportamento do sistema e identificar problemas. Mensagens de depuração acompanham as principais operações, como a inserção e remoção de eventos no escalonador, facilitando a análise do fluxo de execução.

Essas estratégias tornam o programa robusto e resiliente, permitindo a detecção, tratamento e recuperação de falhas, além de garantir a consistência dos dados e o uso eficiente dos recursos.

5 Análise Experimental de Tempo de Execução

Para avaliar o desempenho do algoritmo, foi realizada uma análise do tempo de execução em função do número de pacientes, variando de 10 até o limite máximo no arquivo de entrada. Os tempos foram comparados com a curva teórica de complexidade $n \log(n)$, esperada devido ao uso de estruturas de heap.

5.1 Resultados

O gráfico da Figura 2 apresenta o tempo experimental (em azul) comparado à curva $n \log(n)$ (em vermelho). Ambos apresentam comportamento similar, confirmando a eficiência do algoritmo.

Pequenas discrepâncias em altos valores de n podem ser atribuídas a fatores externos, como alocação dinâmica de memória e leitura de arquivos.

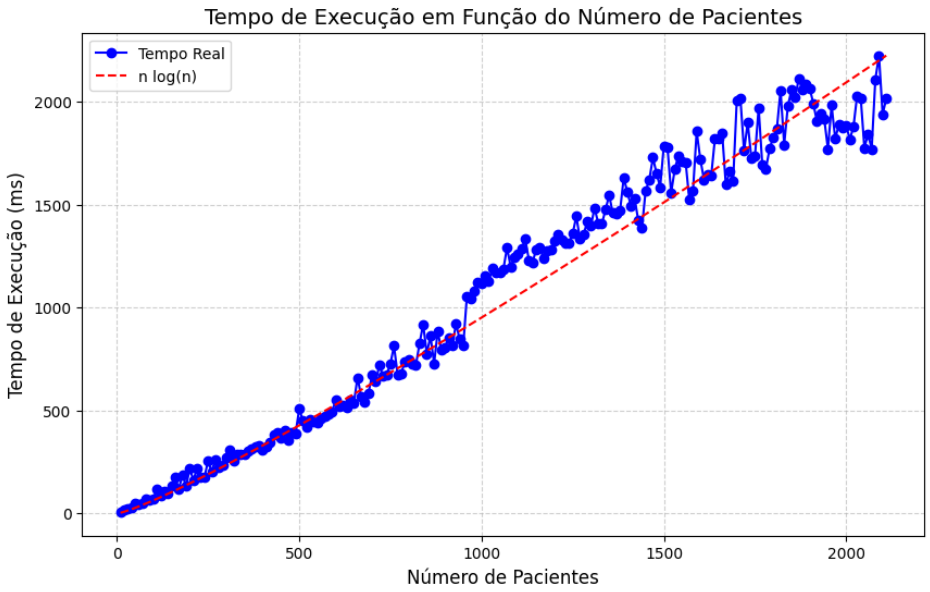

Figura 1: Tempo de execução comparado à curva $n \log(n)$.

5.2 Conclusão

Os resultados confirmam que o algoritmo segue a complexidade $O(n \log(n))$, sendo eficiente para o problema proposto e escalável para um grande número de pacientes. A complexidade de espaço não foi experimentada devido à especificação do TP.

6 Análise Experimental de Tempo de Fila

Os testes foram realizados utilizando os 100 pacientes descritos no Exemplo 2 do VPL de teste. A análise foi conduzida com base nos gráficos gerados, os quais mostram os tempos de espera acumulados em cada fila do sistema.

6.1 Tempos de Espera e Unidades por Procedimento

Os tempos de demora médios de cada procedimento e o número de unidades alocadas são apresentados na Tabela 1. Esses valores foram utilizados como parâmetros na simulação para a análise das filas de espera.

Tabela 1: Tempos médios de demora e número de unidades por procedimento.

Procedimento	Tempo Médio (h)	Número de Unidades
Triagem	0.2	2
Atendimento	0.5	10
Medidas	0.1	20
Testes	0.05	3
Imagem	0.5	50
Instrumentos	0.05	22

6.2 Análise dos Gargalos nas Filas

A primeira observação relevante é que o maior gargalo identificado foi na fila de **triagem**, conforme mostrado na Figura 2. Essa fila apresenta um tempo acumulado significativo devido à sua posição inicial no fluxo de atendimento e ao pequeno número de unidades alocadas (apenas 2 unidades). A baixa quantidade de unidades impacta diretamente a capacidade de processamento da triagem, fazendo com que pacientes se acumulem e prolonguem os tempos de espera para as etapas subsequentes. Este gargalo reflete a importância de ajustar adequadamente o número de unidades para procedimentos iniciais de alta demanda.

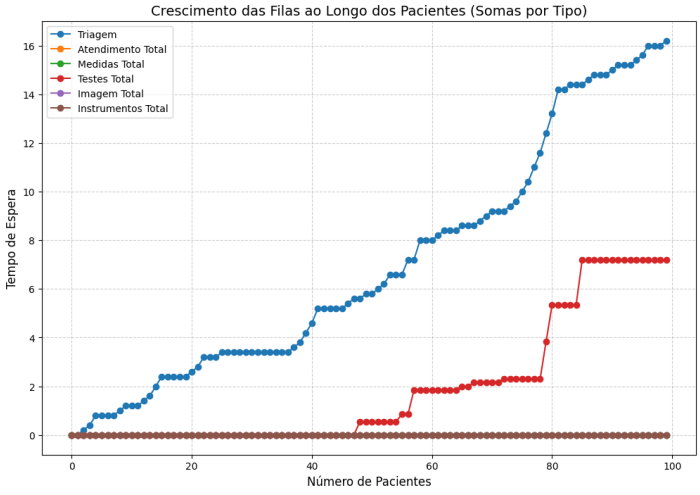

Figura 2: Tempos de fila para 100 pacientes somando prioridades.

6.3 Impacto do Número de Unidades nas Prioridades

Outra observação importante é que as filas de prioridades mais baixas não foram significativamente prejudicadas. O tempo de espera nas filas está fortemente relacionado ao

número de unidades do procedimento. Mesmo para filas de baixa prioridade, a quantidade de unidades sobrepõe o impacto da prioridade no tempo de espera. Isso ocorre porque mais unidades permitem atender simultaneamente a diversos pacientes, minimizando atrasos e equilibrando o sistema como um todo, conforme mostrado na Figura 3. Assim, o impacto da prioridade se torna secundário em relação à disponibilidade de recursos.

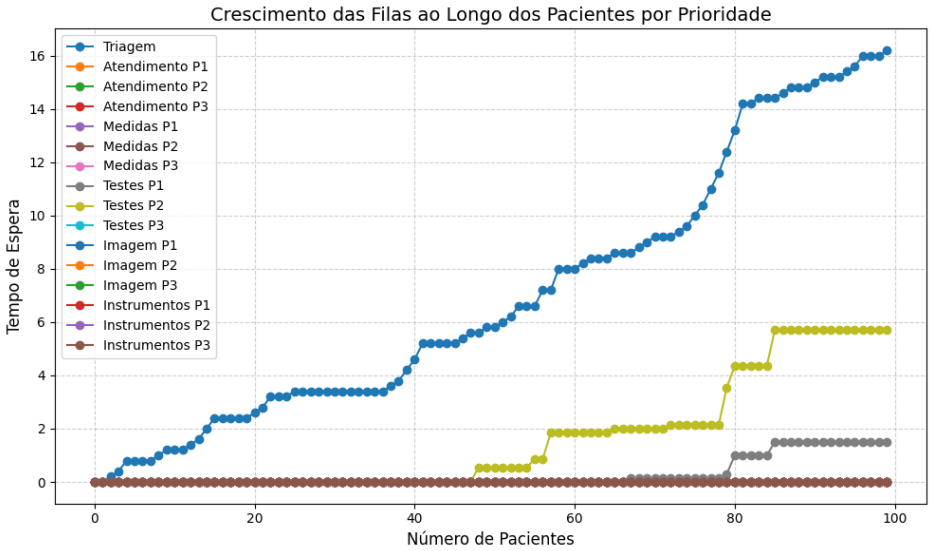

Figura 3: Tempos de fila para 100 pacientes por prioridade.

6.4 Análise do Crescimento do Tempo em Relação ao Número de Unidades

A quantidade de unidades de testes também é relativamente pequena (3 unidades). Dado o impacto significativo do número de unidades no tempo de espera total, foi realizada uma análise específica para o crescimento do tempo de fila em função da quantidade de unidades no procedimento de triagem, esse procedimento foi escolhido por ser o primeiro e seu tempo não ser influenciado por outros procedimentos. Essa análise utilizou uma base de dados com 1000 pacientes, conforme mostrado na Figura 4.

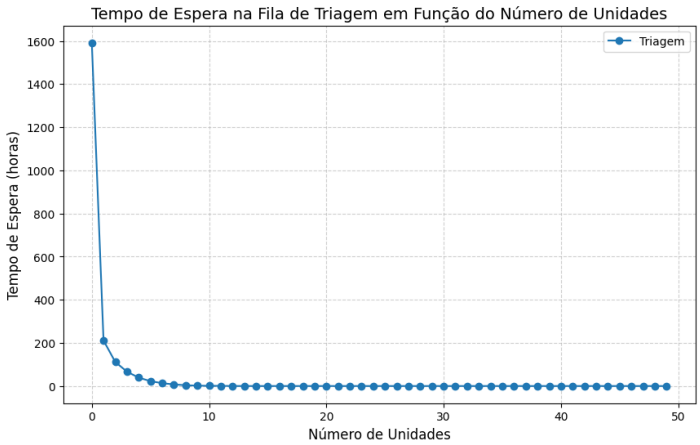

Figura 4: Tempo total na fila de triagem conforme o crescimento de unidades.

6.5 Conclusão

Com base nas análises realizadas, pode-se concluir que:

- O número de unidades é o fator de maior impacto no tempo de espera total, superando a influência da prioridade.
- O sistema de prioridades não gera situações em que pacientes de baixa prioridade fiquem permanentemente sem atendimento ('starving").
- Procedimentos com poucas unidades tendem a criar gargalos que reduzem o congestionamento nos procedimentos subsequentes.

Mais análises podem ser realizadas para explorar outros aspectos do sistema, mas os resultados apresentados são os mais relevantes no contexto desta discussão.

7 Proposta de Otimização: Balanceamento Dinâmico de Filas

A otimização proposta baseia-se na implementação de um balanceamento dinâmico de filas, com o objetivo de reduzir os tempos de espera dos pacientes e mitigar os gargalos causados pela sobrecarga em procedimentos específicos. Essa abordagem permite que o sistema responda adaptativamente às condições de demanda, priorizando filas menos congestionadas.

7.1 Estratégia de Implementação

A estratégia consiste em monitorar continuamente o tempo médio de espera de cada fila e priorizar, no momento de escalonamento, a fila com o menor tempo estimado de espera. O processo seria implementado em três etapas principais:

1. **Cálculo do Tempo Estimado de Espera**: Cada fila atualiza seu tempo médio de espera com base na soma dos tempos de espera dos pacientes já atendidos e no número atual de pacientes em fila. Esse cálculo é realizado de forma dinâmica, refletindo o estado atual do sistema.
2. **Alocação Baseada em Prioridade**: Ao inserir um paciente em um procedimento, o sistema avalia todas as filas disponíveis para esse procedimento e direciona o paciente para a fila com menor tempo médio de espera. Critérios secundários, como o tempo de chegada do paciente ou o grau de urgência, podem ser utilizados em caso de empate.
3. **Reavaliação Contínua**: A cada inserção ou remoção de pacientes, os tempos médios de espera das filas são recalculados. Isso garante que as decisões de alocação reflitam sempre o estado mais recente do sistema.

7.2 Justificativa da Otimização

Essa abordagem é particularmente eficaz em sistemas com alto fluxo de pacientes e grande variabilidade na duração dos procedimentos. Ao priorizar filas com menor tempo estimado de espera, o sistema evita que filas de procedimentos mais demandados fiquem congestionadas, melhorando a experiência dos pacientes e otimizando a utilização dos recursos disponíveis.

7.3 Benefícios Esperados

A implementação do balanceamento dinâmico de filas oferece os seguintes benefícios:

- Redução significativa no tempo médio de espera dos pacientes.
- Melhor distribuição da carga de trabalho entre as filas, reduzindo gargalos em procedimentos altamente demandados.
- Aumento da eficiência global do sistema, garantindo que as unidades estejam sempre ocupadas de forma produtiva.

Essa otimização torna o sistema mais adaptável às condições de alta demanda, garantindo uma operação mais eficiente e equitativa.

8 Conclusão

Este trabalho implementou um sistema de simulação para escalonamento hospitalar, permitindo avaliar o impacto de diferentes configurações de recursos no tempo de espera dos pacientes. A simulação demonstrou a importância de estratégias eficientes de alocação de recursos para minimizar gargalos e otimizar o atendimento.

Os principais aprendizados incluem a aplicação prática de simulação de eventos discretos, o uso eficiente de estruturas de dados como *heaps* e filas, e a análise detalhada do impacto das decisões de escalonamento no desempenho geral do sistema. Além disso, a implementação evidenciou como ajustes dinâmicos e estratégicos podem melhorar a eficiência e a equidade no uso dos recursos hospitalares.

9 Bibliografia

Referências

[1] Wikipedia contributors. *Simulação de eventos discretos*. Disponível em: `https://pt.wikipedia.org/wiki/Simulao_de_eventos_discretos`.

[2] Especificação do Trabalho Prático 2 - Sistema de Escalonamento Hospitalar, DCC/ICEx/UFMG.

[3] Anisio Lacerda, Wagner Meira Jr., Washington Cunha. *Estruturas de Dados*, Slides da disciplina DCC205 - Estruturas de Dados, Departamento de Ciência da Computação, ICEx/UFMG.