INSTITUTO FEDERAL
Paraná
Campus Pinhais

Bacharelado em Ciência da Computação
Estruturas de Dados Avançadas — 25 de março de 2026
Trabalho 1 — v1.0

1 Introdução

O objetivo deste trabalho é a solução de um problema que utilize de maneira integrada as ideias estudadas até o momento: *hash*, *heaps* e números aleatórios.

Em todas as situações apresentadas abaixo, *hashes* são usadas para acesso rápido a elementos de interesse, *heaps* são usados para ordenar ou priorizar os elementos, e números aleatórios podem ser usados para gerar casos de teste variados e reprodutíveis.

1.1 Temas disponíveis

1. Sistema de cache

- *Hash*: usada para acessar os itens da cache
- *Heap*: usado para decidir qual item remover da cache (mais recente, mais antigo, ...)
- Números aleatórios: usados para gerar sequência de requisições
- Testar e estabelecer métricas de desempenho (acerto/erro) e tempo médio de acesso

2. Sistema de recomendação

- *Hash*: usada para mapear usuários e itens avaliados
- *Heap*: usado para manter os itens com melhores avaliações
- Números aleatórios: usados para gerar avaliações de diferentes itens e usuários
- Implementar um mecanismo de *trending topics*

3. Sistema de inventário

- *Hash*: usada para armazenar e encontrar o inventário do jogador
- *Heap*: usado para organizar itens por raridade
- Números aleatórios: usados para gerar itens com raridades diferentes
- Implementar um mecanismo de *drop* balanceado conforme o aparecimento de itens raros no mundo

4. Contagem de frequência de palavras

- *Hash*: usada para contar a frequência de palavras em um texto
- *Heap*: usado para obter as palavras mais frequentes

- Números aleatórios: usados para gerar textos
- Comparar desempenho com diferentes tamanhos de entrada, e também textos reais com textos aleatórios

5. Escalonamento de tarefas

- *Hash*: usada para definir qual processador executará a tarefa
- *Heap*: usado para manter uma fila de prioridades
- Números aleatórios: usados para gerar tarefas com tempos de prioridades aleatórias
- Comparar desempenho com diferentes algoritmos (FIFO, prioridades, diferentes *quantum*, ...)

Outros temas e ideias são bem-vindos, desde que sejam coerentes com a disciplina e devem ser discutidos com o professor da disciplina.

2 Requisitos mínimos

Você deve implementar manualmente as estruturas de dados e a geração de números aleatórios. Não serão aceitos trabalhos que utilizem implementações prontas ou bibliotecas.

- A sua implementação de tabelas *hash* deve suportar pelo menos duas funções de espalhamento diferentes. Você deve justificar sua escolha com base no seu cenário e na sua pesquisa.
- A sua implementação de números aleatórios também deve suportar dois métodos de geração diferentes, justificado com base no seu cenário e na literatura.

Suas estruturas de dados e programas devem ser parametrizáveis, permitindo a entrada de argumentos como tamanho da entrada, distribuição, ... no momento da execução do programa.

Seu trabalho deve conter testes que simulem entradas com tamanhos e complexidades diferentes, em diversos níveis. Seus testes também devem possibilitar a comparação e análises das métricas sugeridas para seu cenário. A análise deve ser crítica, com explicações detalhadas dos fenômenos identificados nos testes.

Crie testes reprodutíveis: crie *scripts* que repetem testes com as mesmas *seeds* de geração de números pseudo-aleatórios, de forma que o mesmo teste pode ser executado diversas vezes.

Seus programas e testes devem calcular e permitir a análise do número de operações, tempo de execução e uso de memória para cada teste executado.

Seu trabalho será avaliado considerando a implementação correta das estruturas de dados e algoritmos, a parametrização dos algoritmos, o uso de diferentes estratégias dentro de um mesmo algoritmo/estrutura de dados, qualidade de código, análise crítica de desempenho e das métricas do cenário, qualidade dos casos de teste gerados. Além disso, será considerado para formação do conceito a qualidade do relatório escrito e o domínio do trabalho no momento da defesa e em outras interações com o professor.

3 Entrega

Você deverá entregar pelo SUAP, até 13/abril, um arquivo *.zip* contendo uma pasta, que por sua vez contém:

- Os arquivos de código e de testes do trabalho;
- Um arquivo de texto chamado `README.md` contendo o nome completo dos integrantes do grupo, uma breve descrição de cada arquivo no trabalho, instruções para execução dos testes e dos programas, e quaisquer outras informações sobre o progresso do trabalho que julgarem relevantes;
- Um arquivo chamado `relatorio.pdf` que contém o relatório da equipe em formato `pdf`;
- Absolutamente mais nada! Remova todos os executáveis e arquivos intermediários que estejam na pasta do trabalho antes de enviá-lo.

O arquivo compactado descrito acima deve ser nomeado com as iniciais do nome de cada integrante separados por um hífen. Por exemplo: se o Fulano da Silva Sousa fez o trabalho com o João de Souza, o arquivo deve ser nomeado `fss-js.zip`. A pasta, dentro do arquivo compactado, deve ter o mesmo nome, a não ser pela extensão.

O trabalho pode ser feito individualmente ou em duplas.

Histórico das Revisões:

- 25/mar/2026 - v1.0: primeira versão.