UNIVERSIDADE FEDERAL DO CEARÁ

Nome: José Robertty e Maria Tassiane
Disciplina: Estrutura de Dados Avançada
Professor: Fábio Dias

Este presente trabalho foi pensado com o intuito da visualização dos tempos de execução e da visualização da complexidade dos algoritmos HeapBinário e o HeapTerciario.

Especificações da Máquina:

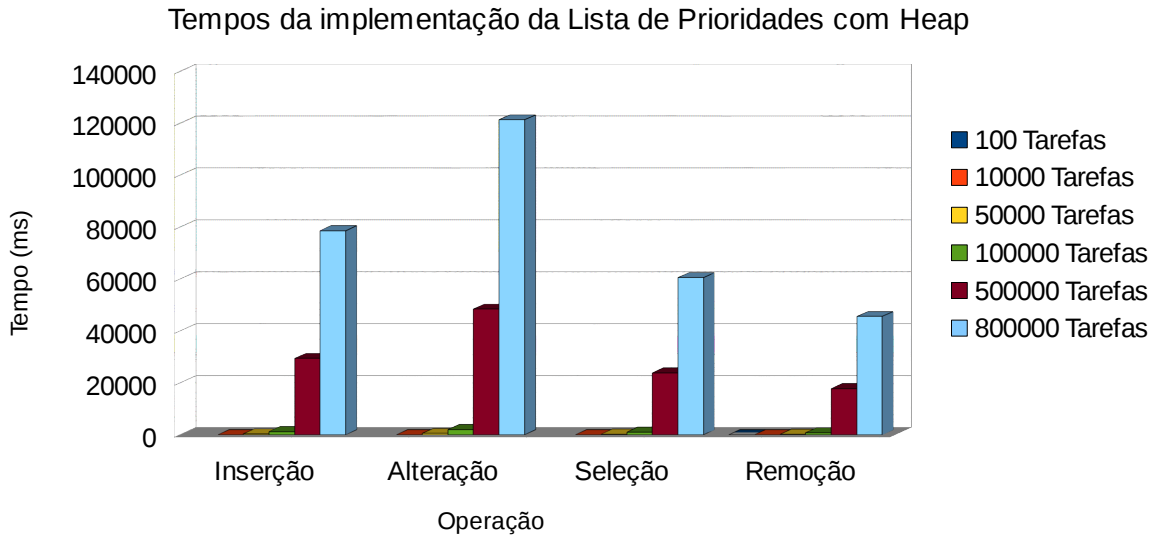
- Sistema operacional: Ubuntu 15.10
- Processador: Intel core i5-4210U 1.70GHz
- Memória: 7,7 GiB
- Tipo de sistema: 64-bit

Ao testarmos os algoritmos, temos os seguintes resultados para o HeapBinário implementado para Lista de Prioridades, temos os tempos (em milissegundos) obtidos:

Operação/ Tarefas	**100**	**10000**	**50000**	**100000**	**500000**	**800000**	**TOTAL**
I	0	33	324	1262	29478	78678	109775
A	0	33	491	2014	48434	121384	172356
S	0	20	250	1003	23867	60577	85717
R	1	18	196	748	17830	45572	64365
TOTAL	1	104	1261	5027	119609	306211	432213

Tabela 1 – Tempos da implementação da Lista de Prioridades com HeapBinário (I = Inserção, A = Alteração, S = Selecionar, R = Remover)

Gráfico 1



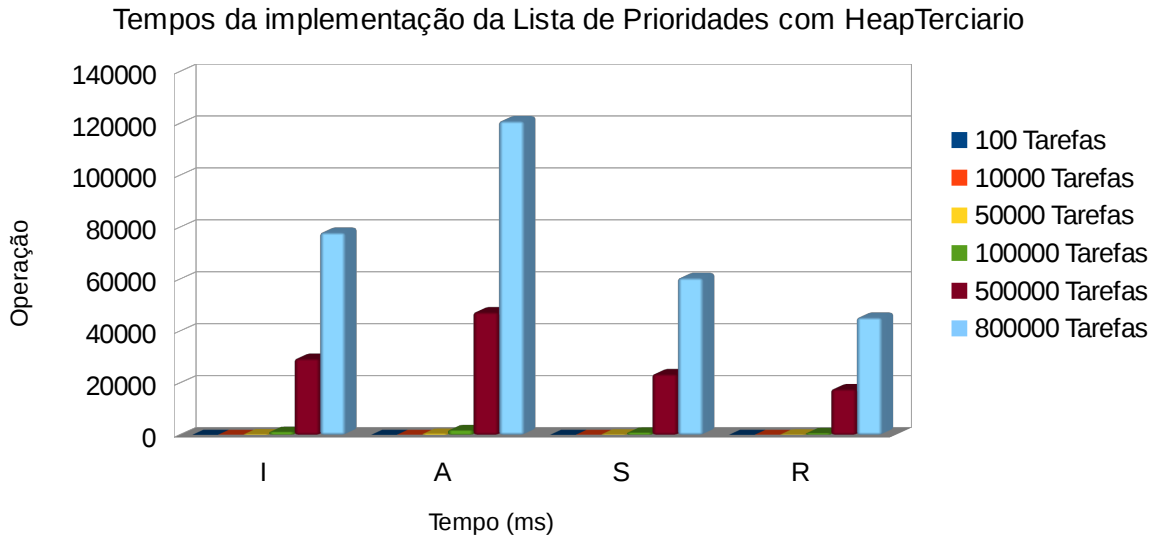
Na tabela 1 e Gráfico 1, vemos que a estrutura de dados HeapBinária gasta muito tempo nas tarefas onde se utiliza mais alteração e gasta pouco tempo nas tarefas onde se utiliza mais remoção e as tarefas que se utiliza mais seleção.

Ao testarmos os algoritmos, temos os seguintes resultados para o HeapTerciario implementado para Lista de Prioridades, temos os tempos (em milissegundos) obtidos:

Operação/ Tarefas	**100**	**10000**	**50000**	**100000**	**500000**	**800000**	**TOTAL**
I	1	39	302	1268	29422	77986	109018
A	1	33	504	1899	47332	120952	170721
S	1	20	251	967	23464	60509	85212
R	1	26	199	752	17695	45247	63919
TOTAL	4	118	1256	4886	117913	304694	428870

Tabela 2 – Tempos da implementação da Lista de Prioridades com HeapTerciario (I = Inserção, A = Alteração, S = Selecionar, R = Remover)

Grafico 2



Na tabela 2 e Gráfico 2, vemos que a estrutura de dados HeapTerciario gasta muito tempo nas tarefas onde se utiliza mais alteração e gasta pouco tempo nas tarefas onde se utiliza mais remoção e as tarefas que se utiliza mais seleção.

Como vemos, nas operações onde o HeapBinário é mais eficiente é exatamente aonde o HeapTerciarios, pois as implementações e complexidades são semelhantes.

Tabela 3

Implementação / Tarefas	**100**	**10000**	**50000**	**100000**	**500000**	**800000**
HeapBinário	1	104	1261	5027	119609	306211
HeapTerciario	4	118	1256	4886	117913	304694

Grafico 3

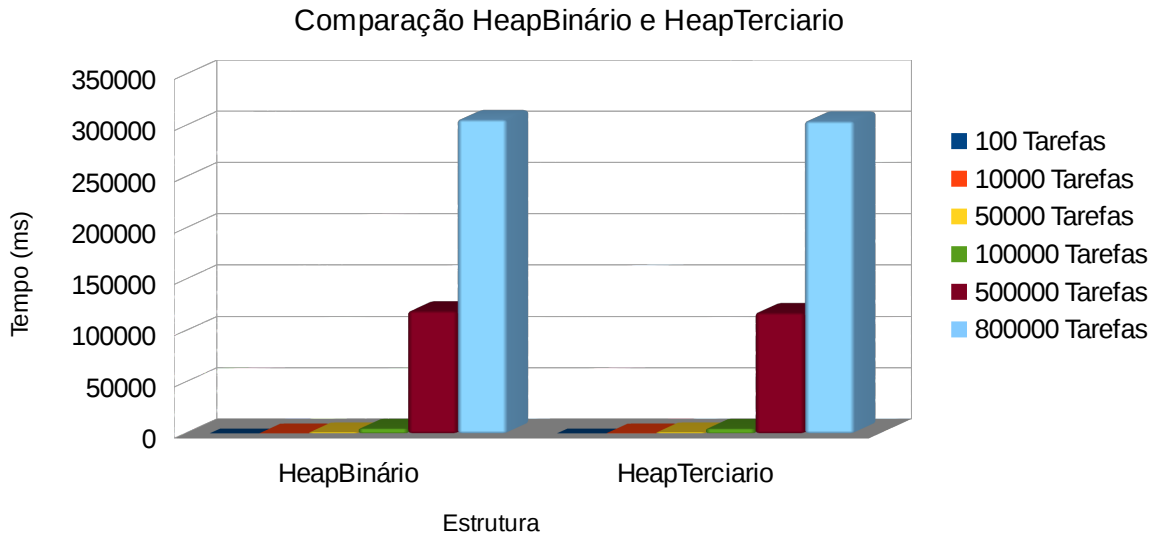


Tabela 3 e Gráfico 3 – Comparação de tempos (em milissegundos) do HeapBinário e o HeapTerciario em relação ao total de tarefas executadas.

É fácil concluir que o HeapBinário com poucas tarefas é mais eficiente, porém com quantidades maiores de tarefas o HeapTerciario fica mais eficiente.