UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Bacharelado em Ciência da Computação

Algoritmos e Estruturas de Dados - GBC034

Professora: Maria Camila Nardini Barioni

- João Vitor Gonçalves Oliveira – 11921bcc024

Extra) Considerando as alterações feitas na função max_heapify, transformando esta em uma função não recursiva obtemos os seguintes resultados:

Entrada com vetor de 10.000 elementos(não recursivo):

Organização do vetor	Tempo ShellSort	Tempo HeapSort
Aleatório	0.000080	0.000240
Decrescente	0.000027	0.000101
Crescente	0.000032	0.000096

Entrada com vetor de 50.000 elementos(não recursivo):

Organização do vetor	Tempo ShellSort	Tempo HeapSort
Aleatório	0.005486	0.007964
Decrescente	0.001051	0.005090
Crescente	0.000931	0.005037

Entrada com vetor de 100.000 elementos(não recursivo):

Organização do vetor	Tempo ShellSort	Tempo HeapSort
Aleatório	0.010507	0.016378
Decrescente	0.003363	0.010768
Crescente	0.001205	0.010131

Abaixo os tempos do HeapSort em que a função max_heapify era recursiva

Entrada com vetor de 10.000 elementos

Organização do vetor	Tempo HeapSort
Aleatório	0.000128
Decrescente	0.000071
Crescente	0.000084

Entrada com 50.000 elementos:

Organização do vetor	Tempo HeapSort
Aleatório	0.008498
Decrescente	0.005441
Crescente	0.006300

Entrada com 100.000 elementos:

Organização do vetor	Tempo HeapSort
Aleatório	0.017690
Decrescente	0.013379
Crescente	0.008918

Considerando os tempos obtidos, obtivemos uma melhora pouco significativa nos casos em que os elementos de entrada eram sequenciais, tanto decrescente como crescente, e um tempo maior em vetores aleatórios. Porém ainda com a melhora em vetores sequenciais seu custo ainda não é melhor que o algoritmo de shellsort.