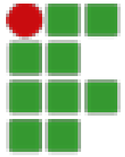


INSTITUTO FEDERAL
Sul-rio-grandense
Campus Gravataí

HEAPSORT
&
CYCLESORT

Adriano Ocampo, Artur Silva e João Victor Dreissig

ORDENAÇÃO: OTIMIZAÇÃO FUNDAMENTAL EM ALGORITMOS

- **Definição:** Transformação de um conjunto de dados desordenado no mesmo conjunto reestruturado para satisfazer uma relação de ordem específica.
- **Aplicação:** Predominantemente em estruturas de dados lineares (vetores, listas).
- **Importância:** É um problema fundamental, pois dados ordenados permitem: Buscas (ex: Busca Binária) e fusões drasticamente mais eficientes.
- **Crucial para:** Otimização de Sistemas de Informação, Bancos de Dados e Algoritmos de Processamento de Dados.

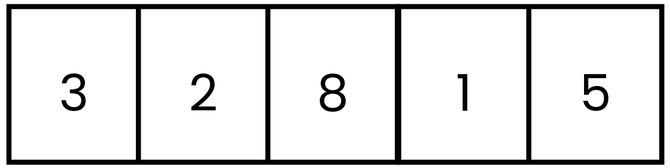
HEAPSORT

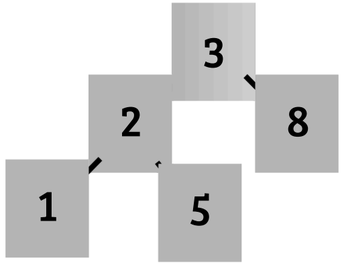
Algoritmo de Ordenação Eficiente

- **O que é:** Ordenação por comparação que usa uma heap binária;
- **Vantagem Principal:** Complexidade de tempo garantida de O(n log n) (melhor e pior caso) – Desempenho muito consistente;
- **Uso de Memória:** É in-place, ideal para recursos limitados;
- **Desvantagem:** É instável, ou seja, não preserva a ordem relativa de elementos com chaves iguais.

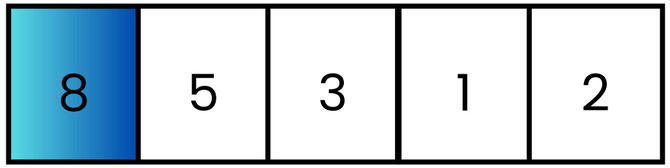
HEAPSORT

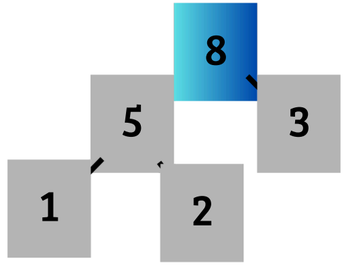
vetor



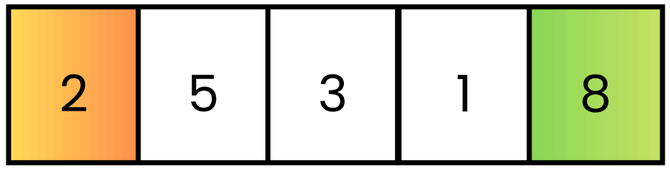


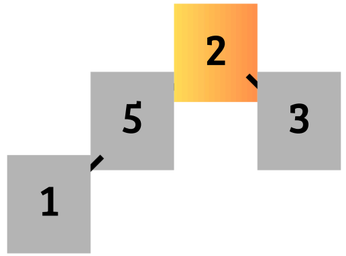
heapify



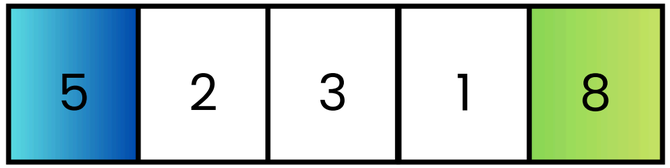


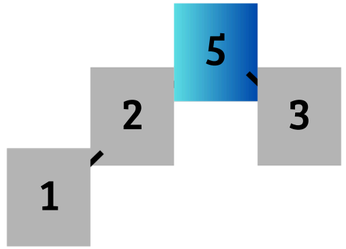
remove



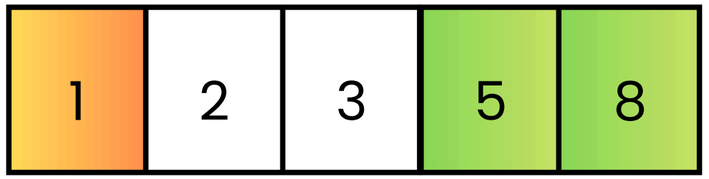


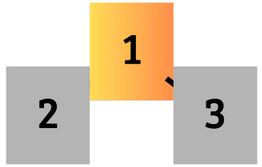
heapify



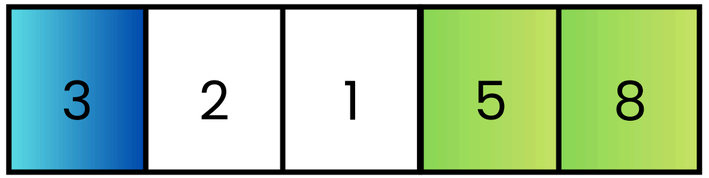


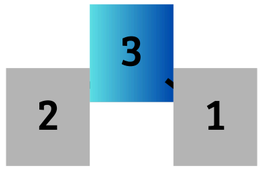
heapify



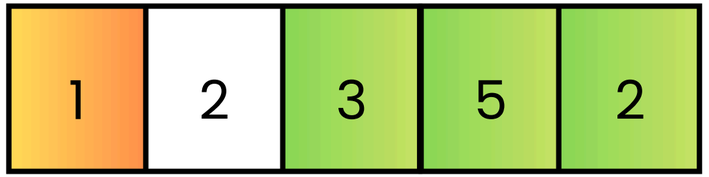


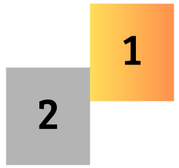
remove



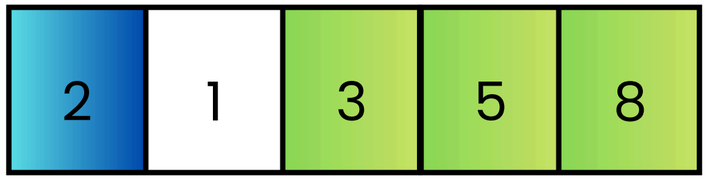


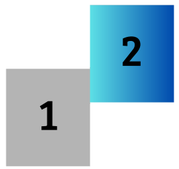
heapify





remove





heapify

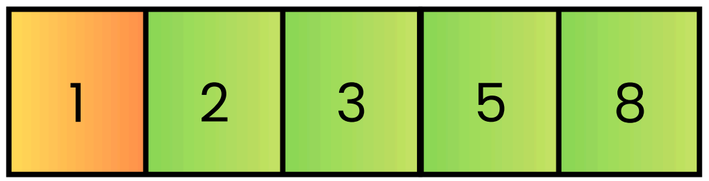


TABELA DE COMPARAÇÃO

Algoritmo	Complexidade (Pior Caso)	Uso de Memória (Extra)	Estabilidade
Heapsort	O(n log n)	O(1)(In-place)	**Não** (Instável)
Quick Sort	$O(n^2)$	O(log n) (Recursão)	**Não** (Instável)
Merge Sort	O(n log n)	O(n)	**Sim** (Estável)

RESULTADO DOS TESTES

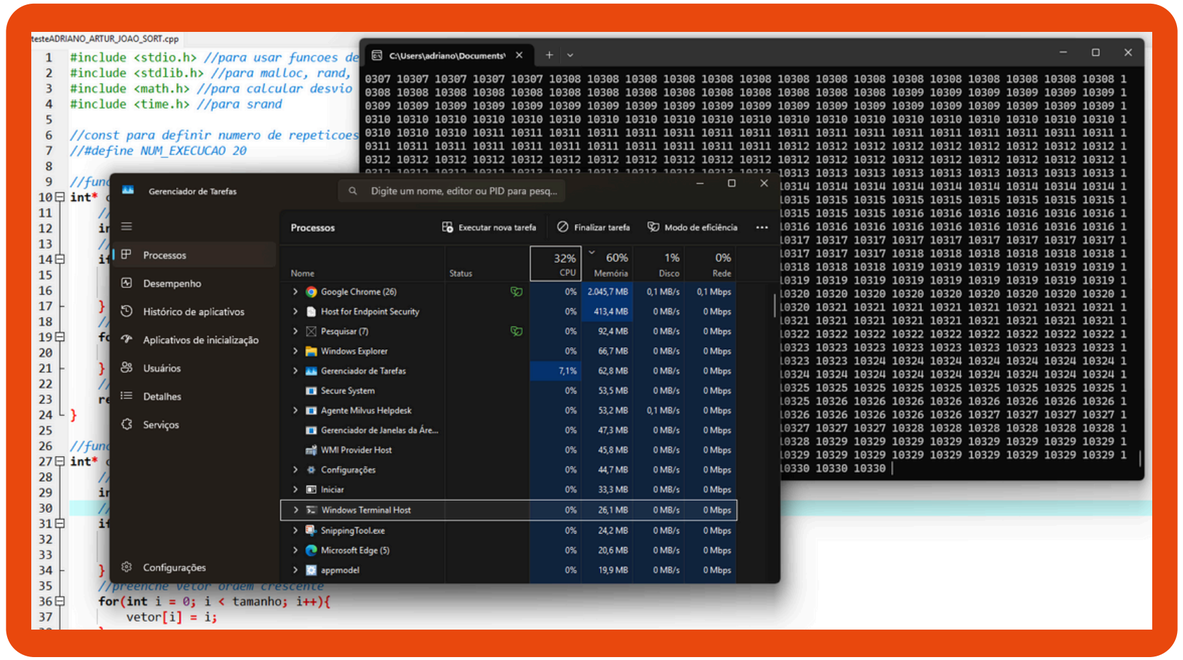
HeapSort		
Vetor de 1.000		
20 execuções		
Vetor aleatório	**Vetor crescente**	**Vetor decrescente**
média: **0.000000seg.**	média: **0.000200seg.**	média: **0.000000seg.**
desvio padrão: **0.000000 seg.**	desvio padrão: **0.000410 seg.**	desvio padrão: **0.000000 seg.**

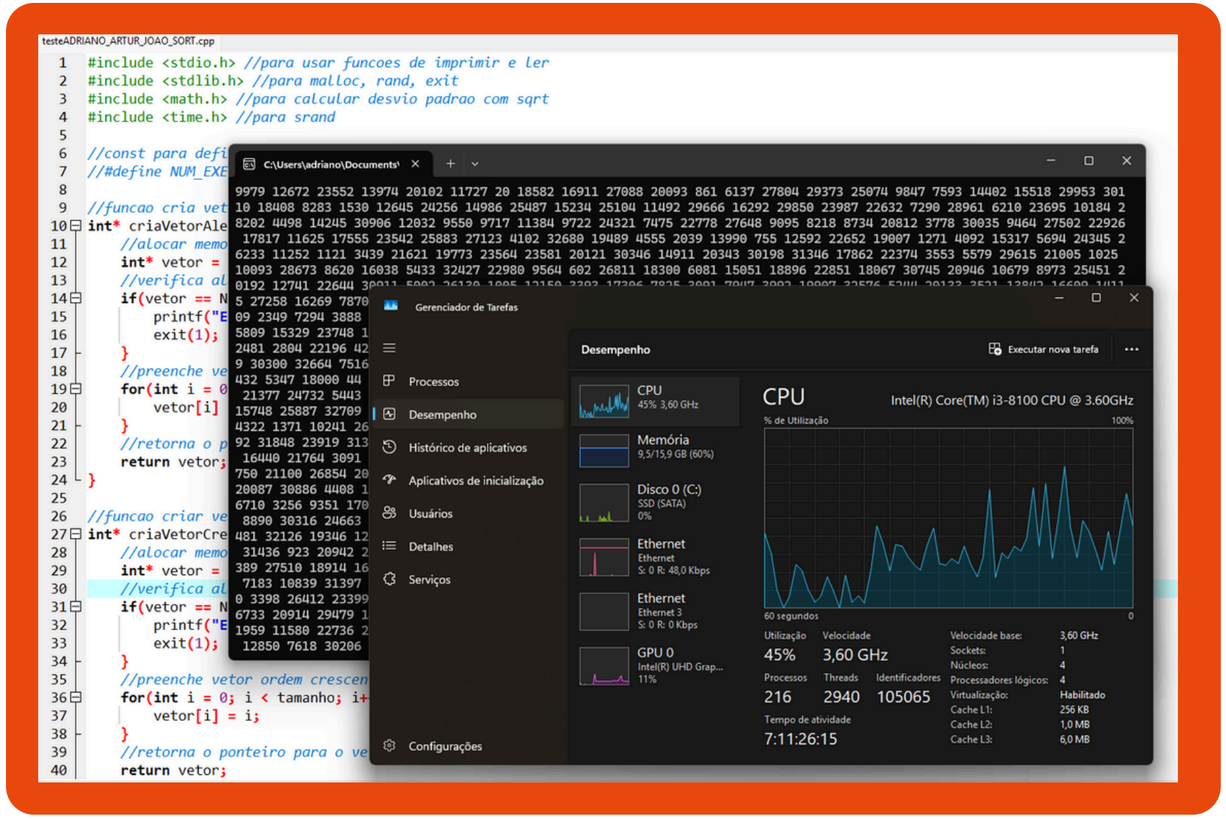
HeapSort		
Vetor de 100.000		
20 execuções		
Vetor aleatório	**Vetor crescente**	**Vetor decrescente**
média: **0.026050seg.**	média: **0.020250 seg.**	média: **0.019200 seg.**
desvio padrão: **0.007451 seg.**	desvio padrão: **0.005098 seg.**	desvio padrão: **0.006396 s**

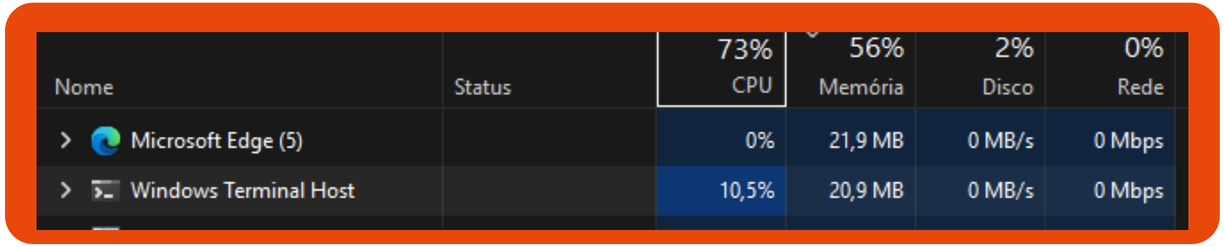
HeapSort		
Vetor de 1.000.000		
20 execuções		
Vetor aleatório	**Vetor crescente**	**Vetor decrescente**
média: **0.346500seg.**	média: **0.225050seg.**	média: **0.226550seg.**
desvio padrão: **0.018648 seg.**	desvio padrão: **0.007437 seg.**	desvio padrão: **0.008217 seg.**

AMBIENTE DE TESTE

CPU	Intel(R) Core(TM) i3-8100 CPU @ 3.60GHz
Memória	16,0 GB
Disco 0 (C:)	KINGSTON SA400S37480G







CYCLESORT

Otimizado para Mínimas Escritas

- **O que é:** Ordenação baseada em comparação que funciona ao identificar e rotacionar ciclos de elementos fora de posição;

- **Vantagem Principal:** É o algoritmo teoricamente ótimo em termos de número de escritas na memória (swaps), garantindo que cada elemento seja escrito em sua posição final no máximo uma vez;

- **Uso de Memória:** É in-place O(1), ideal para sistemas com recursos limitados;

- **Desvantagem Crucial:** Sua complexidade de tempo no pior caso é $O(n^2)$ – o desempenho é lento e inconsistente para grandes conjuntos de dados.

CYCLESORT

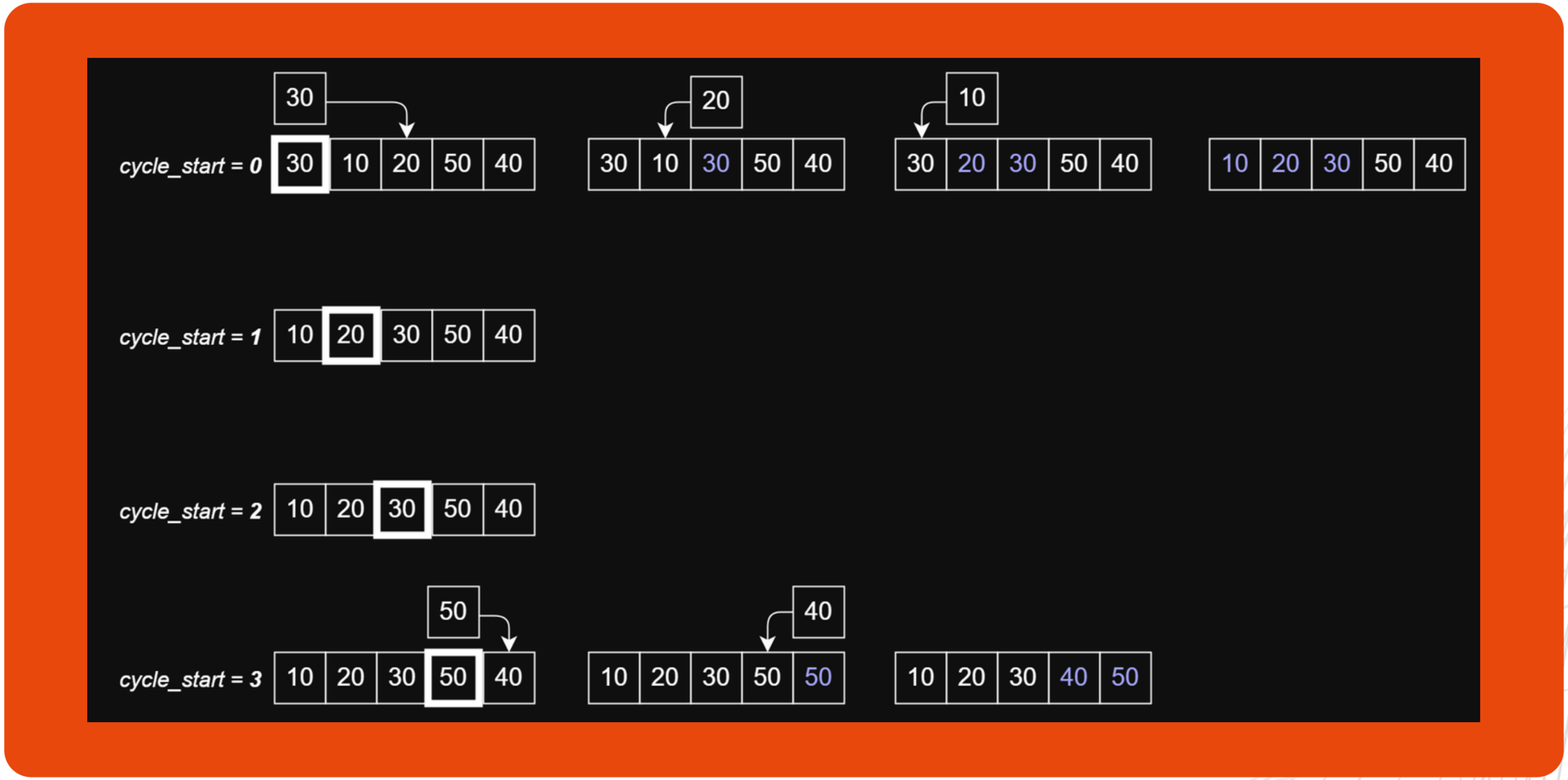

cycle_start = 0
30 10 20 50 40
30 10 30 50 40
30 20 30 50 40
10 20 30 50 40
cycle_start = 1
10 20 30 50 40
cycle_start = 2
10 20 30 50 40
cycle_start = 3
10 20 30 50 40
10 20 30 50 50
10 20 30 40 50

TABELA DE COMPARAÇÃO

Algoritmo	Pior Caso (Tempo)	Uso de Memória (Extra)	Estabilidade	Swaps (Escritas)
Cycle Sort	**$O(n^2)$**	**O(1) (In-place)**	Não (Instável)	**Ótimo (Mínimo)**
Heapsort	O(n log n)	**O(1) (In-place)**	Não (Instável)	Alto

RESULTADO DOS TESTES

CycleSort		
Vetor de 1.000		
20 execuções		
Vetor aleatório	**Vetor crescente**	**Vetor decrescente**
média: **0.004700seg.**	média: **0.001550seg**.	média: **0.001450seg.**
desvio padrão: **0.007385 seg.**	desvio padrão: **0.004774 seg.**	desvio padrão: **0.004489 seg.**

CycleSort		
Vetor de 100.000		
20 execuções		
Vetor aleatório	**Vetor crescente**	**Vetor decrescente**
média: **47.443900 seg**.	média: **11.900800 seg.**	média: **18.077900 seg.**
desvio padrão: **1.010258 seg.**	desvio padrão: **0.885235 seg.**	desvio padrão: **0.246267 seg.**

CycleSort		
Vetor de 1.000.000		
nenhuma execução foi concluída		
Vetor aleatório	**Vetor crescente**	**Vetor decrescente**
Não rodou o programa	Não rodou o programa	Não rodou o programa

MÉDIA EM SEGUNDOS

1 millhão (1194.412 segundos / 19.9 minutos

CycleSort	CycleSort	CycleSort
Vetor de 1.000	Vetor de 1.000	Vetor de 1.000
20 execuções	20 execuções	20 execuções
Vetor Crescente	**Vetor Decrescente**	**Vetor Aleatório**
Desvio Padrão: 0.00000 s	Desvio Padrão: 0.00327 s	Desvio Padrão: 0.00883 s
Média: 0.00000 s	Média: 0.00075 s	Média: 4.13480 s
CycleSort	CycleSort	CycleSort
Vetor de 100.000	Vetor de 100.000	Vetor de 100.000
20 execuções	20 execuções	20 execuções
Vetor Crescente	**Vetor Decrescente**	**Vetor Aleatório**
Desvio Padrão: 0.00883 s	Rodou apenas até o exemplo 11, com o tempo médio de 7.51155 s	Simplesmente não rodou.
Média: 4.13480 s		
CycleSort	CycleSort	CycleSort
Vetor de 1.000.000	Vetor de 1.000.000	Vetor de 1.000.000
20 execuções	20 execuções	20 execuções
Vetor Crescente	**Vetor Descrescente**	**Vetor Aleatório**
Simplesmente não rodou.	Simplesmente não rodou.	Simplesmente não rodou.

CycleSort		
Vetor de 500.000 e 700.000		
500.000		700.000
Vetor crescente		**Vetor crescente**
média:**260.246090 seg.** (10 execuções, sem desvio padrão)		1execução:**552.790000 seg.**

MUITO
Obrigado

INSTITUTO FEDERAL
Sul-rio-grandense
Campus Gravataí