Relatório EP3 - MAC0121

João Gabriel Basi - N° USP: 9793801

1. O programa

O programa recebe um vetor desordenado de n posições e o ordena usando 3-reversões. Quando é possível ordená-lo, o programa imprime uma sequencia de movimentos que fazem isso, quando não é possível, o programa imprime "Nao e possivel".

2. As funções

O programa utiliza duas bibliotecas feitas por mim com algumas funções auxiliares:

- *mystack.h*: Define uma pilha e algumas funções sobre ela;
- *myvector.h*: Tem funções que ajudam no manuseio de vetores e matrizes.

A função principal do programa é a *triSort*, que coordena como o vetor será ordenado. Para ordenar vetores com tamanhos ímpares ela utiliza as funções:

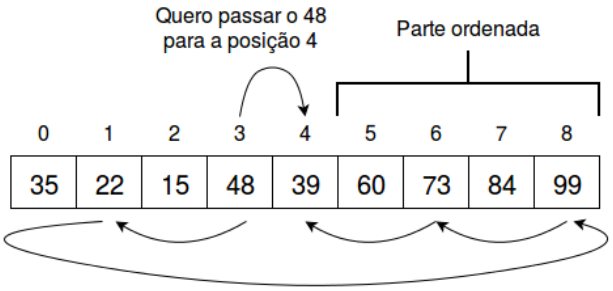
- *pMaior*: Recebe um vetor e seu tamanho e retorna a posição do maior elemento do vetor;
- *triReversao*: Recebe um vetor, seu tamanho e uma posição e realiza uma 3-reversão na posição fornecida.

Já para vetores de tamanhos pares, ela utiliza as funções:

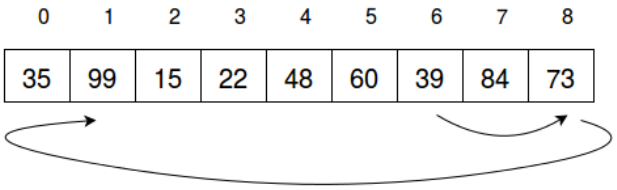
- *separa*: Recebe um vetor, seu tamanho e dois vetores com metade de seu tamanho e separa os elementos do vetor maior pela paridade de seus índices nos dois vetores menores;
- *junta*: Faz o contrário da função *separa*, ou seja, volta os elementos dos vetores menores para o vetor maior intercalando-os;
- *mergeSort*: Função de ordenação recursiva que recebe um vetor, o intervalo a ser ordenado, uma pilha e um coeficiente de paridade. Ordena o vetor normalmente utilizando um Merge Sort, calcula a posição relativa dos elementos movimentados no vetor original utilizando o coeficiente de paridade e as guarda na pilha.

3. Métodos e otimizações

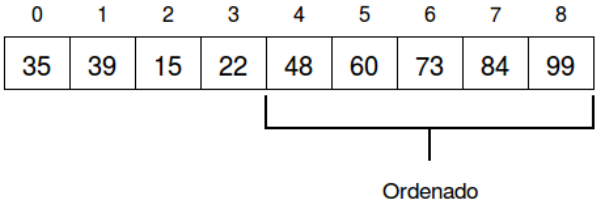
Para vetores com um número ímpar de posições, a cada passo eu identifico o maior número da parte desordenada do vetor utilizando a *pMaior*, se a paridade do índice dele for igual à paridade do índice da posição final, ele faz 3-reversões no elemento para frente até ele chegar na posição final, se for diferente, ele faz 3-reversões para trás até o número dar a volta no vetor e chegar na posição de destino e volta para arrumar os números que já tinham sido ordenados. Exemplo na imagem.

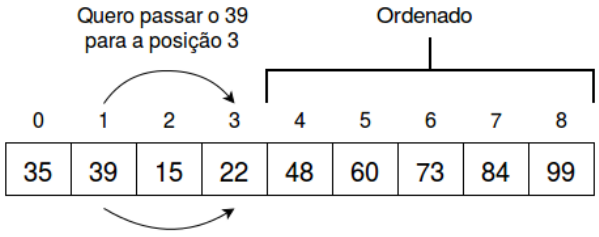


Como 3 != 4 mod 2, então o 48 tem que mudar de paridade. O caminho mais curto é voltando no vetor

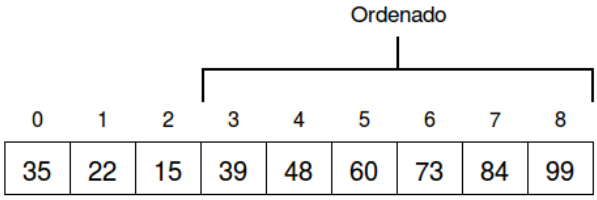


Agora voltamos os números que já estavam ordenados para seus lugares





Como 1 = 3 mod 2, então o 39 só precisa ir direto para a posição 3



Como sempre é possível ordenar vetores ímpares, já que um número pode ir para qualquer posição do vetor, o programa imprime os movimentos feitos à medida que eles vão acontecendo.
Para vetores com um número par de posições, como não importa qual rotação sea feita, a paridade da posição de um elemento não muda, eu separei os números com índices pares e ímpares em dois vetores menores e os ordenei com um algoritmo de ordenação tradicional, já que nesses vetores uma 2-rotação é equivalente à uma 3-rotação no vetor original. Como é possível que o vetor seja impossível de ordenar, já que um número não pode ir para qualquer posição do vetor, eu guardo os movimentos feitos em uma pilha. Depois de ordenar-los eu junto eles de volta no vetor maior, se o vetor final estiver ordenado, o programa imprime os movimentos feitos, se não, ele imprime "Nao e possivel".

5. Prós e contras

Prós:

- Ordena vetores pares em ordem O(nlogn) no pior caso;
-

Contras:

- Tem complexidade O(nlogn) para vetores pares ordenados também, por utilizar o merge sort;
- O algoritmo de ordenação para ímpares é muito simples, por isso ele pode não achar os melhores movimentos que ordenam o vetor.