Trabalho Final Estrutura de Dados I

Trabalho Final: Algoritmos de Ordenação

Nome: Cristiano da Silva Pinheiro
Data: 01/07/2024

1) Escolher um algoritmo dentre os seguintes: shellsort, heapsort e quicksort
R: O algoritmo escolhido foi o shellsort.

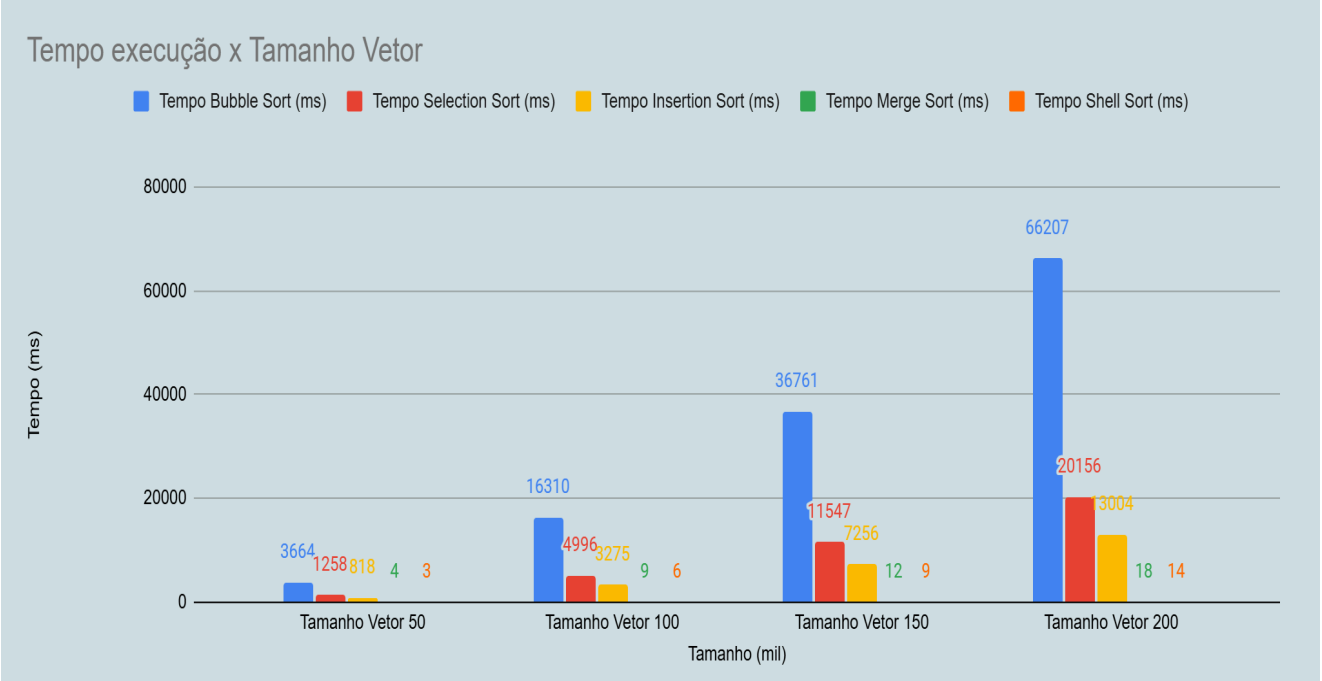
2) Implementar, na linguagem C, o algoritmo escolhido.

```c
// Função responsável por ordenar vetor através do método shell sort
// Vetor a ser ordenado [vetor]
// Tamanho do vetor [tamanho]
void shell_sort(int vetor[], int tamanho) {
    int intervalo, i, j, temp;
    // Inicializa o intervalo como metade do tamanho do array e divide por dois até que intervalo == 1
    for (intervalo = tamanho / 2; intervalo > 0; intervalo /= 2) {
        // Percorre o array a partir do intervalo, ou seja, divide o vetor em sub vetores os quais terão os valores das posições nos intervalos de n
        for (i = intervalo; i < tamanho; i++) {
            // Move os elementos maiores que o valor atual para a frente
            temp = vetor[i];
            for (j = i; j >= intervalo && vetor[j - intervalo] > temp; j -= intervalo) {
                vetor[j] = vetor[j - intervalo];
            }
             // Insere o valor atual na posição correta
            vetor[j] = temp;
        }
    }
}
```

4) Verificar o tempo de execução do algoritmo implementado com vetores de 50 mil, 100 mil, 150mil e 200 mil números inteiros.

Método	Tamanho Vetor 50	Tamanho Vetor 100	Tamanho Vetor 150	Tamanho Vetor 200
Tempo Bubble Sort (ms)	3664	16310	36761	66207
Tempo Selection Sort (ms)	1258	4996	11547	20156
Tempo Insertion Sort (ms)	818	3275	7256	13004
Tempo Merge Sort (ms)	4	9	12	18
Tempo Shell Sort (ms)	3	6	9	14

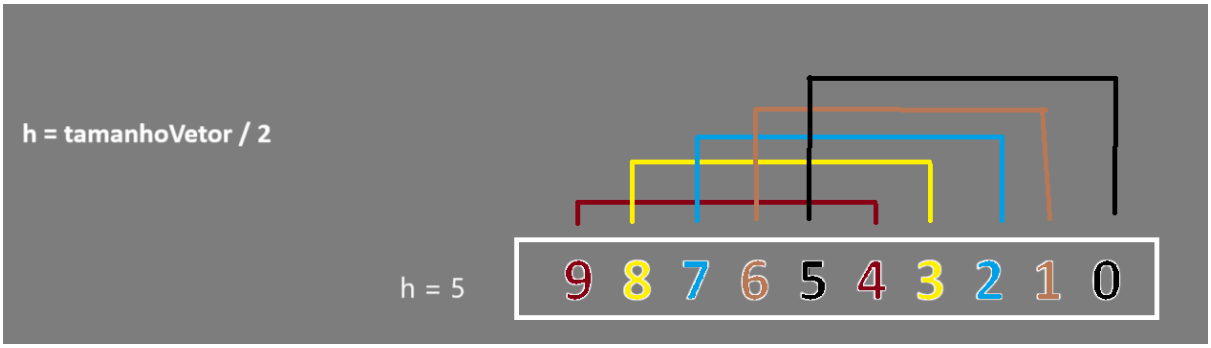
5) Comparar graficamente os tempos de execução do algoritmo implementado com os algoritmos bubblesort, selectionsort, insertionsort e mergesort, para vetores de 50 mil, 100 mil, 150 mil e 200 mil números inteiros.



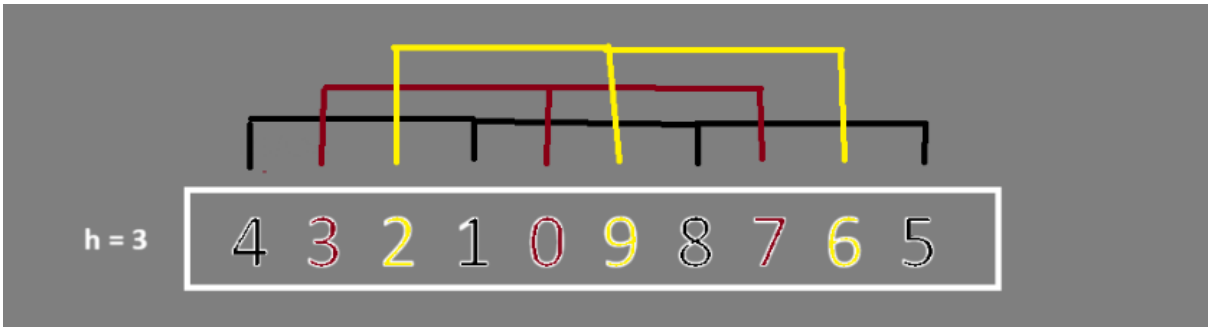
Entre os métodos implementados é possível concluir que o método Bubble é o mais lento em todos os cenários e também o que mais demora conforme o aumento da quantidade de dados a serem ordenados. Outro fato é que os métodos Merge Sort e o Shell Sort são os mais eficientes, com tempos consistentemente baixos.

6) Utilizar o vetor de tamanho 10, abaixo, e mostrar, didaticamente, o passo a passo do processo de ordenação realizado pelo algoritmo implementado. int vetor[10] = {9,8,7,6,5,4,3,2,1,0}

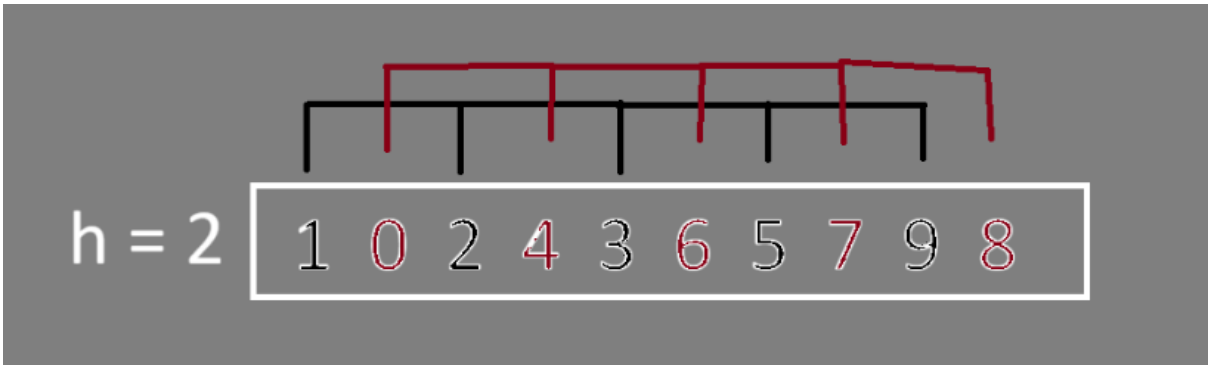
PRIMEIRO CICLO DE ORDENAÇÃO:
Define o “h” como o tamanho do vetor dividido por 2, neste cenários será 5. Após isso, divide o vetor em sub-vetores num intervalo de “h”. Dessa forma, é pegado o primeiro valor do grupo e comparado com o seguinte, assim, se o valor for menor que o atual é alterada a posição de cada um. Após finalizar a sequência de comparações é alterado o valor de “h” para “h” / 2 e é realizado as comparações até que o valor de “h” resulte a 1.



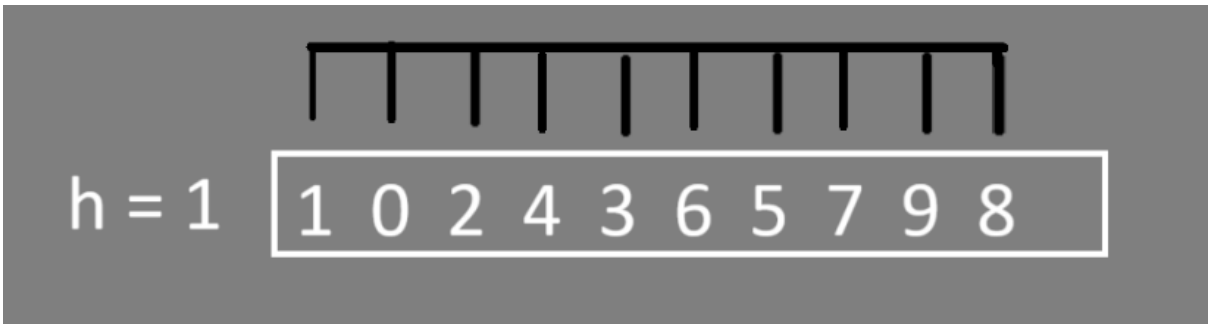
SEGUNDO CICLO:



TERCEIRO CICLO:



QUARTO CICLO:



LISTA ORDENADA: