

Comparação em desempenho dos métodos de ordenação Bubble Sort, Insertion Sort, Selection Sort, Merge Sort e Quick Sort

Após nossos estudos sobre os cinco métodos de ordenações ficam evidente que o método Quick Sort e o Merge Sort são os melhores, conforme é ilustrado nas figuras (1, 2, 3, 4 e 5) abaixo conseguimos identificar que sua quantidade de comparações e quantidade de trocas diminui significativamente.



Figura 1 - Gráfico de desempenho do método Bubble Sort



Figura 2 - Gráfico de desempenho do método Insertion Sort



Figura 3 - Gráfico de desempenho do método Selection Sort



Figura 4 - Gráfico de desempenho do método Merge Sort



Figura 5 - Gráfico de desempenho do método Quick Sort

Atividade:

Baseado no código abaixo, que gera números ordenados, invertidos e aleatórios, construa um programa para provar os resultados das cinco figuras acima. Exemplo do bubbleSort() do código abaixo.

```
/*
PROCESSO AVALIATIVO N2 - ESTRUTURA DE DADOS

EQUIPE
NOMES:

CONSEGUIU CHEGAR NO RESULTADO?
R:

SE NÃO, POR QUE NÃO CONSEGUIU CHEGAR NO RESULTADO?
R.:

*/

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define TAMANHO 10000

// Prototipo de Função
void geraNumero(int *vet, int op);
void bubbleSort(int *vet);
void imprimirVetor(int *vet);

int main (void){
    // Variáveis
    int vet1[TAMANHO];

    // Entrada de dados;
    geraNumero(vet1,3);
    // printf("Desordenado: \n\n");
    // imprimirVetor(vet1);

    // Processamentos dos dados
    bubbleSort(vet1);

    // Saída de dados
    // printf("\n\nOrdenado: \n\n");
    //imprimirVetor(vet1);
    return 0;
}

// Função geração de números
void geraNumero(int *vet, int op) {
    int i, j;

    switch (op) {
        case 1: // Ordenados
            for (i = 0; i < TAMANHO; i++) {
                vet[i] = i + 1;
            }
            break;
```

```

        case 2: //Invertidos
            for (i = 0; i < TAMANHO; i++) {
                vet[i] = TAMANHO-i;
            }
            break;
        case 3: // Aleatórios
            for (i = 0; i < TAMANHO; i++) {
                vet[i] = (int) (rand() % TAMANHO);
            }
            break;
    }
}

// Função ordenação bubble sort
void bubbleSort(int *vet){
    int n, troca, i, aux, qtd_trocas, qtd_comparacoes;

    n = 1;
    troca = 1;
    qtd_trocas = 0;
    qtd_comparacoes = 0;

    // Ponto do algoritmo para iniciar o tempo de execução
    float tempo_inicial = clock();
    while (n <= TAMANHO && troca == 1) {
        troca = 0;
        for (i = 0; i <= TAMANHO-2; i++) {
            // Ponto do algoritmo para contar as comparações
            qtd_comparacoes++;
            if (vet[i] > vet[i + 1]) {
                // Ponto do algoritmo para contar as trocas
                qtd_trocas++;
                troca = 1;
                aux = vet[i];
                vet[i] = vet[i + 1];
                vet[i + 1] = aux;
            }
        }
        n = n + 1;
    }
    // Ponto do algoritmo para calcular o tempo de execução
    float tempo_final = clock() - tempo_inicial;

    // Saída de dados
    printf("\nQuantidade de comparacoes: %i\n",qtd_comparacoes);
    printf("Quantidade de trocas: %i\n",qtd_trocas);
    printf("Tempo de execucao do algoritmo: %.3f",tempo_final/1000);
}

// Função impressão do vetor
void imprimirVetor(int *vet){
    int i;
    for(i=0;i<TAMANHO;i++){
        printf("%i, ",vet[i]);
        if ((i+1)%14 == 0){
            printf("\n");
        }
    }
}

```

```
}
```

Esta atividade pode ser realizada em equipes no máximo de **três** integrantes e deve ser entregue conforme data no Microsoft Teams.

No Microsoft Teams um membro da equipe deve postar o algoritmo, com o cabeçalho do código exemplo preenchido, nomes dos integrantes e se conseguiu chegar no resultado. O código deve estar em C ANSI ou C++.

Avaliação deste trabalho:

- **Parte 1** (2 pontos) - Atender aos seguintes requisitos:
 - Realizar contagem de tempo conforme o exemplo, considere apenas a parte para ordenação;
 - Utilizar vetores com 50000 posições (**[Melhor caso]** ordenado crescente, 1 até 50000);
 - Utilizar vetores com 50000 posições (**[Pior caso]** ordenado decrescente, 50000 até a);
 - Utilizar vetores com 50000 posições (**[Caso médio]** números aleatórios de 1 até 50000);
 - Identificar as linhas com comparações;
 - Identificar as linhas com trocas;
 - Comentário no final do código de sugestão de melhoria e porquê dessa sugestão (vale usar pesquisa na internet).
- **Parte 2** (2 pontos) - O que vai ser avaliado no projeto:
 - Cabeçalho do código exemplo preenchido, nomes dos integrantes;
 - Dizer se conseguiu chegar no resultado dos gráficos;
 - Comentários no código (seja específico, preciso analisar se entendeu ou não o código);
 - Padronização no código;
 - Mínimo de 10 execuções para cada caso (melhor, pior e médio);
 - Gráfico de 10 execuções para cada um dos casos.
 - Contendo: número de comparações, trocas e tempo.
 - Gráfico comparativo dos métodos.
 - Faça uma média de execução de cada um dos casos (melhor, pior e médio) e crie um gráfico com esses valores

O que enviar:

- Código.
- Apresentação em PPT ou PDF, ou link do YouTube (máximo de 10 min de vídeo).
 - Gráfico(s) de 10 execuções do melhor caso.
 - Gráfico(s) de 10 execuções do pior caso.
 - Gráfico(s) de 10 execuções do caso médio.
 - Gráfico(s) comparativo das execuções.
 - Igual ao exibido no início desse arquivo.

Bom trabalho.