INSTITUTO DE COMPUTAÇÃO

Universidade Federal do Rio de Janeiro
Centro de Ciências Matemáticas e da Natureza

Primeiro Trabalho Prático de Arquitetura de Computadores e Sistemas Operacionais

Implementação de três algoritmos de ordenação em MIPS usando o simulador MARS MIPS simulator - Missouri State University. Entregar um relatório por grupo (até 3 pessoas por grupo) contendo o código em C usado como base para a tradução MIPS e os mapeamentos feitos para cada algoritmo.

Entrega do relatório e dos códigos: 05/10

Exemplos de algoritmos de ordenação

Existem diversos algoritmos diferentes de ordenação, vou explicar o funcionamento de alguns que podem ser usados no trabalho. Caso queiram implementar algum algoritmo que não esteja aqui, coloquem no relatório uma explicação rápida de como o algoritmo funciona. Os pseudocódigos mostrados podem ser alterados para a versão final em C, contanto que continuem utilizando vetores (e não ponteiros).

Insertion Sort (Ordenação por inserção)

Este algoritmo varre o vetor sequencialmente e vai inserindo o elemento atual na posição ordenada em relação aos elementos anteriores.

Exemplo de Pseudocódigo:

```
FUNÇÃO INSERTION_SORT (A[], tamanho)
        VARIÁVEIS
                i, j, eleito
        PARA i <- 1 ATÉ (tamanho-1) FAÇA
                eleito = A[i];
                j = i-1;
                ENQUANTO ((j>=0) E (eleito < A[j])) FAÇA
                        A[j+1] = A[j];
                        j = j-1;
                FIM_ENQUANTO
                A[j+1] = eleito;
        FIM_PARA
FIM
```

Selection Sort (Ordenação por seleção)

Este algoritmo busca o menor elemento ainda não ordenado do vetor para colocar na posição atual.

Exemplo de Pseudocódigo:

```
FUNÇÃO SELECTION_SORT (A[], tamanho)
        VARIÁVEIS
                i, j, indiceMenor
        PARA i <- 0 ATÉ (tamanho-1) FAÇA
                indiceMenor = i;
                PARA j <- i + 1 ATÉ tamanho FAÇA
                        SE (A[j] < A[indiceMenor]) ENTÃO
                                indiceMenor = j
                        FIM_SE
                FIM_PARA
                SE (I != indiceMenor) ENTÃO
                        TROCA(A, i, indiceMenor);
                FIM_SE
        FIM_PARA
FIM
```

Bubble Sort (Ordenação por flutuação)

Este algoritmo percorre o vetor várias vezes e vai trocando os elementos de par em par de forma que o maior elemento em cada subsequência fica na última posição dessa sequência.

Exemplo de Pseudocódigo:

```
FUNÇÃO BUBBLE_SORT (A[], tamanho)
        VARIÁVEIS
                i, trocado
        trocado = False
        FAÇA
                PARA i <- 0 ATÉ (tamanho-1) FAÇA
                        SE (A[i] > A[i+1]) ENTÃO
                                TROCA(A, i, i+1)
                                trocado = True
                        FIM_SE
                FIM_PARA
        ENQUANTO trocado
FIM
```

Merge Sort (Ordenação por mistura)

Este algoritmo divide o vetor em dois para chamar o algoritmo recursivamente com casos menores. Quando termina de resolver os casos menores, o algoritmo faz um *merge*, uma combinação das duas partes, onde cada parte está ordenada já.

Exemplo de Pseudocódigo:

```
FUNÇÃO MERGE (A[], tamA, B[], tamB)
        VARIÁVEIS
                i, j, C, k
        i, j, k = 0,0, 0;
        ENQUANTO (i < tamA) E (j < tamB) FAÇA
                SE (A[i] > B[j]) ENTÃO
                        C[k] = B[j];
                        j = j+1;
                        k = k+1;
                SENÃO
                        C[k] A[i];
                        i = i+1;
                        k=k+1;
                FIM_SE
        FIM_ENQUANTO
        ENQUANTO (i < tamA) FAÇA
                C[k] = A[i];
                k = k+1;
                i = i+1;
        FIM ENQUANTO
        ENQUANTO (j < tamB) FAÇA
                C[k] = B[j];
                k = k+1;
                j = j+1;
        FIM ENQUANTO
FIM
FUNÇÃO MERGE_SORT (A[], tamA)
        VARIÁVEIS
                V1, V2, tamV1, tamV2
        SE (tamA == 1)
                RETORNAR A
        tamV1 = n/2
        tamV2 = n - tamV2
        V1 = A[0:tamV1]
        V2 = A[tamV1:tamA]
        V1 = MERGE_SORT (V1, tamV1)
```

```
        V2 = MERGE_SORT (V2, tamV2)
        RETORNA MERGE(V1, tamV1, V2, tamV2)
FIM
```

Quick Sort

Este algoritmo é recursivo e seleciona a cada chamada um elemento do vetor para ser o *pivô*. O vetor é então ordenado de forma que todos os elementos menores que o *pivô* estejam à esquerda dele e os maiores à direita. Por fim, o algoritmo chama ele próprio para a ordenação das duas sublistas: à esquerda e à direita do *pivô*.

Exemplo de Pseudocódigo:

```
FUNÇÃO QUICK_SORT (A[], inicio, fim)
        VARIÁVEIS
                i, j, pivo
        i = inicio
        j = fim
        pivo = A[(i + j)/2]
        ENQUANTO (i <= j) FAÇA
                ENQUANTO (A[I] < pivo) FAÇA
                        i = i + 1;
                FIM_ENQUANTO
                ENQUANTO (A[j] > pivo) FAÇA
                        j = j - 1;
                FIM_ENQUANTO
                SE (i <= j) ENTÃO
                        TROCA(A, i, j);
                        i = i + 1;
                        j = j - 1;
                FIM_SE
        FIM_ENQUANTO
        SE (inicio < j) ENTÃO
                QUICK_SORT(A, inicio, j)
        FIM_SE
        SE (i < fim) ENTÃO
                QUICK_SORT(A, i, fim)
        FIM_SE
FIM
```