public static void main(String[] args) throws IOException {

int quantidade = 10000;

int[] vetor = new int[quantidade];

for (int i = 0; i < vetor.length; i++) {

vetor[i] = (int) (Math.random()\*quantidade);

}

long tempoInicial = System.currentTimeMillis();

selectionSort(vetor);

long tempoFinal = System.currentTimeMillis();

System.out.println("Executado em = " + (tempoFinal - tempoInicial) + " ms");

}

public static void selectionSort(int[] array) {

for (int fixo = 0; fixo < array.length - 1; fixo++) {

int menor = fixo;

for (int i = menor + 1; i < array.length; i++) {

if (array[i] < array[menor]) {

menor = i;

}

}

if (menor != fixo) {

int t = array[fixo];

array[fixo] = array[menor];

array[menor] = t;

}

}

}

SELECT SORT

O SelectionSort guarda a posição do menor elemento na variável “menor” e percorre o array procurando por um valor menor. Caso este valor seja encontrado então a variável “menor” recebe a posição deste valor. Por último é checado se a posição do menor elemento é diferente da posição atual, se isso for verdade então é feita uma troca de valores, colocando o menor elemento na frente.

Vamos entender o passo a passo do algoritmo apresentado: começamos com o laço for, que percorre todos os elementos, guardando o valor da iteração atual na variável “fixo”. Em seguida usamos uma variável auxiliar chamada menor que guardará o valor de fixo.

Tendo em mãos a variável menor, que na primeira iteração é igual ao primeiro elemento do vetor, percorremos (através de outro laço for) todos os elementos do array partindo do segundo elemento até o último.Então imagine a situação, se temos o array com as posições 0,1,2,3 a variável menor guardará o valor '0' e o segundo laço irá percorrer os elementos na posição 1,2,3.

A cada vez que um elemento for visitado, no segundo laço for, é feita uma comparação se o elemento atual é menor que o elemento que está na posição indicada pela variável menor. Se isso for verdade então a variável menor recebe o valor de i, ou seja, a nova posição do menor elemento será o atual lido.

Se a variável menor guarda a posição do menor elemento, então sempre que um elemento menor que o menor atual for encontrado, a posição do menor será alterada para o atual.

No final do segundo laço for teremos na variável menor a posição do menor elemento lido, se esta posição for diferente da posição atual que está na variável fixo então faremos uma troca.

Lembre-se que a variável fixo guarda a posição do elemento que está sendo lido no primeiro laço for, e a variável menor guarda a posição do menor elemento encontrado em todo o vetor. Sendo assim, se ambos forem diferentes capturamos o valor contido na posição da variável “fixo” e armazenamos na variável “t”, apenas para não perder o valor. Depois colocamos o valor do elemento na posição da variável “menor” dentro da posição da variável “fixo”. O que temos até agora é que o menor elemento está na primeira posição do vetor e também em outra posição, ou seja, na sua original, agora precisamos colocar o valor de “t” dentro da posição antiga do menor elemento.

Suponha o seguinte vetor: 20,30,40, 10. Na primeira execução do algoritmo acima a posição “0” (valor 20) é gravada na variável fixo, e a variável menor também recebe “0”. Em seguida, no segundo laço for, é verificado se 30 menor que 20, 40 é menor que 20 e 10 é menor que 20. Como 10 é menor que 20, então a variável menor recebe o valor 3 que corresponde a posição do menor elemento do vetor.

Como o valor de menor (3) é diferente do fixo (0) então ocorre a troca:

10,30,40,10.

Quando colocamos o valor de “t” na posição antiga da variável menor, obtemos o seguinte:

10,30,40,20.

Quando ocorrer a segunda iteração do primeiro laço do algoritmo a leitura se dará a partir do valor 30, então a lista considerada será: 30,40,20. Isso será feito até que não reste mais nenhum elemento a ser lido.

