UNIVERSIDADE DO VALE DO RIO DOS SINOS

Jayme Riegel Gomes Neto

Relatório sobre métodos simples de classificação

Aplicação de algoritmos simples de classificação em arrays de inteiros unidimensionais.

Porto Alegre – Rio Grande do Sul
2020

1- Introdução

O objetivo deste trabalho é fazer uma comparação dos métodos de classificação simples estudados em aula, são eles: Bubble Sort, Insertion Sort e Selection Sort. Aplicaremos os métodos em arrays simples de inteiros, formatados de diferentes modos e quantidades. Dito isso, cronometraremos os tempos de execução e iremos entender as efetividades de cada um em cima dos diferentes arrays.

2- Materiais e métodos

Foi feito testes de tempo de execução, utilizando arrays de inteiros que foram organizados em diferentes tamanhos e formas de composição, assim gerando dados para fazer as comparações sobre os métodos. São três tipos de organização dos dados propostos: Array Aleatório, Array Crescente e Array Descrescente. As quantidades de inteiros em cada array testadas foram respectivamente: 100000 (cem mil), 200000 (duzentos mil) e 300000 (trezentos mil). O objetivo é ordenar os três tipos de arrays em formato decrescente.

3- Análise descritiva dos dados

Aqui irei apresentar os dados gerados de cada um dos três métodos de classificação, bem como explicar como cada um se comporta.

3.1- Bubble Sort

Bubble Sort é o método de classificação mais simples dos três estudados, ele faz comparações entre os dados vizinhos e, caso o objetivo da ordenação seja verdadeiro, os troca de posição, ou seja, sendo meu algoritmo setado para ordenar inteiros de forma crescente, ele irá fazer a comparação entre n e n+1, caso n+1 seja maior que n, os dois trocam de posição e o algoritmo segue para a próxima comparação. O método precisa de uma variável de corte, que iniciará do tamanho do array e irá diminuindo uma unidade a cada volta inteira no array, isso é utilizado porque sabemos que os números depois da variável de corte já estão ordenadas.

Com os testes, podemos notar que este método é o mais demorado quando se trata de um array de inteiros composto de forma aleatória, como é mostrado na tabela abaixo. Por outro lado, quando utilizado em arrays compostos de forma decrescente, ele se sai melhor que os outros em duas quantidades.

Array Aleatório

Método	Quantidade	Tempo (em seg)
Bubble Sort	300000	333
Insertion Sort	300000	102
Select Sort	300000	60
Bubble Sort	200000	75
Insertion Sort	200000	28
Select Sort	200000	20
Bubble Sort	100000	23
Insertion Sort	100000	8
Select Sort	100000	12

Array Decrescente

Método	Quantidade	Tempo (em seg)
Bubble Sort	200000	14
Insertion Sort	200000	27
Select Sort	200000	19
Bubble Sort	100000	4
Insertion Sort	100000	8
Select Sort	100000	6

3.2- Insertion Sort

Insertion Sort divide o array em dois, elementos já ordenados e elementos que ainda não foram ordenados. O seu estado inicial é com o primeiro elemento ordenado, assim a comparação se inicia no segundo elemento, onde ele é comparado com o primeiro. O próximo elemento (terceiro) irá ser comparado com o seu anterior repetidamente, até que a condição não seja mais atendida, assim o próximo elemento (quarto) irá fazer as comparações, e assim suscetivelmente até o final do array, quando o método se encerra.

O método se sai muito bem lidando com arrays que já estejam ordenados, e tem pior desempenho quando precisa fazer uma ordenação inversa.

Array Crescente

Método	Quantidade	Tempo (em seg)
Bubble Sort	300000	28
Insertion Sort	300000	0
Select Sort	300000	53
Bubble Sort	200000	5
Insertion Sort	200000	0
Select Sort	200000	14
Bubble Sort	100000	1
Insertion Sort	100000	0
Select Sort	100000	4

Array Decrescente

Método	Quantidade	Tempo (em seg)
Bubble Sort	300000	66
Insertion Sort	300000	96
Select Sort	300000	50
Bubble Sort	200000	14
Insertion Sort	200000	27
Select Sort	200000	19
Bubble Sort	100000	4
Insertion Sort	100000	8
Select Sort	100000	6

3.3- Selection Sort

Selection Sort é parecido com o Insertion em sua composição, também divide o array em duas partes, ordenado e não ordenado, porém, diferente do Insertion, ele começa sem nenhum elemento em sua parte ordenada. O método seleciona seu primeiro elemento e compara com todos os demais do array, assim achando o menor entre eles (em caso do algoritmo estar setado para ordenar de forma crescente) e fazendo a troca do elemento selecionado com o menor do array, caso o primeiro elemento não seja o menor, ou seja, se o primeiro elemento é o número 9 e o terceiro é o número 1 e o no restante do array não existe um número mais baixo que o 1, o primeiro e o terceiro trocam de lugar, assim o primeiro elemento do array passa a estar em ordem. O processo é feito em todos os elementos do array até o final do mesmo e não houver mais parte em desordem.

O método foi mais eficaz com arrays setados de forma aleatória em duas quantidades, além de ser o mais demorado em arrays que já estão setados igual o objetivo final.

Array Aleatório

Método	Quantidade	Tempo (em seg)
Bubble Sort	300000	333
Insertion Sort	300000	102
Select Sort	300000	60

Bubble Sort	200000	75
Insertion Sort	200000	28
Select Sort	200000	20

Array Crescente

Método	Quantidade	Tempo (em seg)
Bubble Sort	300000	28
Insertion Sort	300000	0
Select Sort	300000	53

Bubble Sort	200000	5
Insertion Sort	200000	0
Select Sort	200000	14

Bubble Sort	100000	1
Insertion Sort	100000	0
Select Sort	100000	4

4- Conclusão

Métodos de classificação simples são eficazes quando trabalhamos com uma quantidade de elementos baixas, pois normalmente são feitas muitas comparações, o que pode acarretar em uma lentidão quando submetidos a grandes quantidades de elementos. Todos os três métodos estudados lidam com variáveis auxiliares para guardarem números temporariamente e tem uma dinâmica parecida no seu macro. Com o estudo, foi possível desenvolver e fixar dinâmicas de laços de repetição e troca de posição em arrays.