

Escola de Artes, Ciências e Humanidades - EACH-USP

Introdução à Análise de Algoritmos

Exercício de Programação 1

Bruno Alves Catão Silva, nº USP: 7553278, Turma 04

18 de novembro de 2021

Resumo

O presente trabalho apresenta uma comparação entre dois algoritmos que buscam resolver o problema da mediana. Uma vez elaborados os algoritmos de ordenação e busca, baseados nos modelos trabalhados em classe, seus tempos de execução foram analisados graficamente e suas respectivas eficácias comparadas.

1 Introdução

De acordo com [SRdAL17], algoritmos de ordenação são coleções de instruções que levam uma série de vetores de modo a facilitar sua localização dentro de um arranjo. O exemplo citado pelos autores é uma lista de presença escolar na qual os nomes devem ser organizados alfabeticamente.

Veras et al.[VASdPJ10] afirmam que 25 por cento do uso da CPU é usado por algoritmos de ordenação, o que mostra a necessidade de se otimizar esse tipo de processo.

Como o enunciado do EP pede, utilizaremos aqui o Quick Sort para fazer a ordenação, posto que é um algoritmo de ordem $\theta(n \log(n))$

```
QUICKSORT(A)
1  if  $n > 0$ 
2    then  $q \leftarrow \text{Particao}(A)$ 
3          $\text{QuickSort}(A[1 : q - 1])$ 
4          $\text{QuickSort}(A[q + 1 : n])$ 
```

Figura 1: QuickSort como está na apostila.

Por sua vez, algoritmos de busca são uma série de comandos utilizados para encontrar determinado valor num arranjo [Alv]. Em seu artigo, Alves [Alv] compara os algoritmos Busca Sequencial e Busca Binária, frisando que, ainda que a eficiência de cada processo dependa da situação, a Busca Sequencial tende a demorar mais devido conforme aumentam o número de vetores na sequência.

Estes algoritmos serão a base para criar os algoritmos pedidos no enunciado, Selecao1(A) e Selecao2(A), que se encontram no meu [Github](#).

2 Objetivo

O objetivo deste trabalho é comparar os algoritmos usados em Selecao1(A) e Selecao2(A) no tocante à eficácia de seus processos, em especial no quesito tempo utilizado para sua execução.

```

SELECAO1( $A, i$ )
1  Ordene( $A$ )
2  return  $a_i$ 

```

Figura 2: Do enunciado.

```

SELECAO2( $A, i$ )
1   $q \leftarrow \text{Particao}(A)$ 
2  if  $n = 1$ 
3      then return  $a_1$ 
4  if  $i < q$ 
5      then return Selecao2( $A[1 : q - 1], i$ )
6      else if  $i > q$ 
7          then return Selecao2( $A[q + 1 : n], i - (q + 1)$ )
8      else return  $a_q$ 

```

Figura 3: Do enunciado.

3 Configurações

Os algoritmos foram rodados nas seguintes configurações:

- Processador: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz
- RAM instalada: 8,00 GB.

4 Esboço da correção

Utilizaremos o Princípio da Indução para esboçar a correção dos algoritmos.

Além de uma série de atribuições e mensagens ao usuário, a função [Seleção 1](#) tem em seu âmago uma função QuickSort, que se inicia com a função Partition. Como as atribuições e mensagens ao usuário podem ser representadas por constantes, serão ignoradas neste esboço, nos importando apenas o estudo da Partição e QuickSort.

```

Particao( $A$ )
1  $pivo \leftarrow a_n$ 
2  $i \leftarrow -1$ 
4 for  $j \leftarrow 1$  até  $n$ 
5 do if  $a_j \leq pivo$ 
6 then  $i \leftarrow i + 1$ 
7  $a_i \leftrightarrow a_j$ 
8  $a_{i+1} \leftarrow a_j$ 
9 return  $i + 1$ 

```

O ponto invariante do algoritmo *Partição*(A) é se encontra na linha 5. Este é o ponto de comparação ao qual a função voltará n vezes, até que a_j seja maior ou igual ao pivô de valor a_n . Quando isso ocorre, a_{i+1} recebe o valor de a_j , e a função retorna $i + 1$, que será utilizado no QuickSort.

```

    QuickSort(A)
1 If  $n > 0$ 
2 Then  $q \leftarrow Particao(A)$ 
3 QuickSort( $A[1 : q - 1]$ )
4 QuickSort( $A[q + 1 : n]$ )

```

Com $n > 0$, o QuickSorte recebe o resultado da Partição na variável q , e usa esta para organizar os elementos de acordo com sua distância do pivô.

Por fim, para o algoritmo [Seleção2](#), como já provamos a correção de Partição(A), precisamos analisar os comandos seguintes:

Inicialização: Para $n = 1$, o arranjo já está em ordem e, portanto, a_1 é a única resposta possível.

Manutenção: Para $n \geq 1$, temos $i < q$, que nos retorna a recursão $Selecao2(A[q : 1 - 1], i)$ ou $i > q$, que nos dá a recursão $Selecao2(A[q + 1 : n], i - (q + 1))$.

Término: por fim, saímos do laço quando $i = q$, e a função retorna o vetor desejado, representado por a_q .

5 Resultados

Os algoritmos base do EP foram criados de modo que o usuário pudesse inserir fatores como o tamanho desejado do array e o vetor procurado neste. Para o teste, foram padronizados os valores possíveis, de modo que o array nunca fosse maior que as centenas de milhões de elementos, limite a partir do qual a diferença entre os algoritmos se tornava óbvia.

A tabela a seguir demonstra as diferenças entre os tempos utilizados pelas iterações dos algoritmos:

Iteração	Selection1	Selection2
1	0	0
2	0	0
3	0	0
4	0	0
5	1	0
6	2	0
7	5	0
8	11	1
9	22	2
10	58	2
11	120	3
12	214	6
13	569	8
14	1457	16
15	3108	25
16	16451	56
17	65243	117
18	178133	189

Tabela 1: Comparação dos tempos dos algoritmos



Figura 4: Comparação entre os tempos dos algoritmos.

6 Aplicações

Além das aplicações citadas na introdução, os algoritmos criados neste trabalho podem ser úteis em questões práticas da estatística. Como o próprio enunciado cita, eles foram criados para identificar a mediana de um arranjo, mas também podem ser utilizados para descobrir qualquer valor dentro deste.

7 Conclusão

Como ficou demonstrado pelos resultados mostrados na tabela e nos gráficos, a função Seleção2 se torna cada vez mais superior à função Seleção1 à medida em que aumenta o número de elementos nos arranjos. A diferença fica ainda mais patente quando se leva em consideração que, ao menos na máquina testada, o algoritmo Seleção1 levou cerca de 2 minutos para entregar um resultado de um arranjo de grandeza na casa de 10^7 , enquanto a função Seleção2 entregava o mesmo resultado em 189 milissegundos.

Referências

- [Alv] Julia Rezende Alves. Análise de desempenho dos algoritmos de busca sequencial e de busca binária como ferramentas de um sistema de gerenciamento bancário.

- [SRdAL17] Jackson EG Souza, João Victor G Ricarte, and Náthalee Cavalcanti de Almeida Lima. Algoritmos de ordenação: Um estudo comparativo. *Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA (ISSN 2526-7574)*, (1), 2017.
- [VASdPJ10] Rodrigo MS Veras, Flavio HD Araujo, Romuere RV Silva, and Iális C de Paula Jr. Ferramenta computacional para o ensino de algoritmos de ordenação. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 1, 2010.