Dois Algorítmos de Ordenação : Bubble Sort e Cocktail Sort

Daniel Henrique de Freitas Macedo
Gardênia Georgia Barbosa de Siqueira
Lucas Decesares Cunha Lima
Israel Fonseca Neto

25 de novembro de 2018

0.1 Resumo

O objetivo desse trabalho é comparar de forma qualitativa e quantitativa o desempenho entre dois algoritmos de ordenação. Os tipos escolhidos foram o Bubble Sort e o Cocktail Sort por motivos pedagógicos. Foi levado em consideração o tempo de execução para escolher o melhor.

Palavras-Chave: Algorítmo de Ordenação, BubbleSort, Cocktailsort.

0.2 Abstract

The main objective of this work is compare the performance of two sort algorithms in a qualitative and quantitative way. The two choosen are Bubble Sort and Cocktail Sort for pedagogical reasons. Our methodology considered time of execution to choose wich is the best.

Key-words: Sort algorithm, Bubble sort, Cocktail sort

0.3 Introdução

Os algorítmos de ordenação tem grande importância para as ciências da computação e informática geral, quando se estuda essa área o primeiro exemplo que aparece é o Bublle Sort devido a sua simplicidade de código e a lógica básica e é a partir dele que se introduzem novas formas de algorítmos de ordenação, e entre elas a Cocktail Sort que como será visto adiante, é uma variação de Bubble Sort. Nesse trabalho irá se explicar alguns conceitos básicos de algorítmos de ordenação no intuito de trazer ao leitor como funciona o trabalho com tais algorítmos e no desenvolvimento iremos comparar seus desempenhos.

0.3.1 O que é um algorítmo de ordenação ?

Algorítmos de ordenação são códigos de computador utilizados para colocar um lista de elementos em uma determinada ordem, por exemplo, dada uma da lista de 10 numeros aleatórios coloca-los em ordem crescente. O uso de algorítmos de ordenação é útil para vários setores da sociedade, os supermercados por exemplo precisam ordenar suas listas de produtos de acordo com a data de validade, preço, quantidades vendidas ou não vendidas e seu desempenho comercial depende da capacidade de manipular esses dados. Na estatística, esses algorítmos podem ser utilizados para colocar os dados em rol crescente, fundamental para extrair mais informação dos sistemas que os envolvem.

0.3.2 O algorítmo Bubble Sort

O algorítmo de ordenação mais básico é o Bubble Sort, ordenação por bolha ou ordenação por flutuação. A ideia é :
Dado um vetor com N células tendo cada uma delas um número aleatório num dado intervalo numérico, faça um programa que coloque todas essas células em ordem crescente.

Uma das formas de implementar esse algorítmo no computador é em linguagem C com o código a seguir:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<unistd.h>
#define MAX 4
#define MAXSORT 100

int main(void)
{
int x;
int indicador;
int vetor[MAX];
srand(time(NULL)=getpid());
for(i=0;i<MAX;i++)
{
vetor[x]=rand()%MAXSORT;
printf("vetor[%d]=%d",i+1,vetor[i]);
}
do
{
indicador=0;
for(ix=0;x<MAX;x++)
if(vetor[x]>vetor[x+1])
{
auxiliar=vetor[x+1];
vetor[x+1]=vetor[i];
vetor[x]=auxiliar;
indicador=indicador+1;
}
}
while(indicador!=0)
for(x=0;x<TOTAL;x++)
printf("vetor[%d]=%d",x+1,vetor[x]);
return EXIT_SUCCES;
}
```

Um possível resultado do algorítmo acima é:

```
vetor[1]=3
vetor[2]=35
vetor[3]=47
vetor[4]=23
vetor[1]=3
vetor[2]=12
vetor[3]=23
vetor[4]=35
```

O processamento do Bubble Sort funciona comparando um número da célula da lista com o seu sucessor e se esse último for menor, então troca de lugar, cada vez que ele realiza uma troca a variavel `indicador` ganha um ponto, ao final de cada ciclo, essa variável é verificada e se seu valor for diferente de zero o ciclo se repete novamente e a variável indicador recebe valor zero. Em algum momento todos os números já estarão ordenados, dessa forma a variável `indicador` terá valor nulo e o laço para. Qundo isso acontecer outro laço `for` irá rodar para que o novo vetor seja impresso na tela.

1. vetor[1]=3
   vetor[2]=35
   vetor[3]=47
   vetor[4]=23

2. indicador=0
3. vetor[1]=3 > vetor[2]=35 ? Falso
4. vetor[2]=35 > vetor[3]=47 ? Falso
5. vetor[3]=47 > vetor[4]=23 ? Verdadeiro
6. auxiliar=vetor[4]
7. vetor[4]=vetor[3]
8. vetor[3]=auxiliar
9. indicador=0+1=1

   Nova lista
   vetor[1]=3
   vetor[2]=35
   vetor[3]=23
   vetor[4]=47

10. indicador!=0 ? Verdadeiro

11. indicador=0

    Nova lista
    vetor[1]=3
    vetor[2]=35
    vetor[3]=23
    vetor[4]=47

12. vetor[1]=3 > vetor[2]=35 ? Falso
13. vetor[2]=35 > vetor[3]=23 ? Verdadeiro
14. auxiliar=vetor[3]
15. vetor[3]=vetor[2]
16. vetor[2]=auxiliar
17. indicador=0+1=1

    Nova lista
    vetor[1]=3
    vetor[2]=23
    vetor[3]=35
    vetor[4]=47

18. vetor[3]=35 > vetor[4]=47 ? Falso
19. indicador !=0 ? Verdadeiro
20. indicador=0

    Nova lista
    vetor[1]=3
    vetor[2]=23
    vetor[3]=35
    vetor[4]=47

21. vetor[1]=3 > vetor[2]=23 ? Falso
22. vetor[2]=23 > vetor[3]=35 ? Falso
23. vetor[3]=35 > vetor[4]=47 ? Falso
24. indicador !=0 ? Falso
25. Imprime lista na tela
26. Finaliza o programa.

0.4 Algorítmo Cocktail Sort

O algorítmo de ordenação Cocktail Sort é uma variação de Bubble Sort, nele o vetor com números aleatórios é percorrido "na ida e na volta "sendo que na volta a troca é inversa, se o vetor[i] for menor que o vetor[i-1] então os valores são trocados atrávés da variável auxiliar.

Abaixo está um exemplo de algorítmo Cocktail Sort :

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<unistd.h>
#define MAX 4
#define MIN 0
#define MAXSORT 10

int main(void)
{
int indicador,auxiliar;
int x;
int vetor[MAX]={0};
srand(time(NULL)+getpid());

for(x=MIN;x<MAX;x++)
{
vetor[x]=rand()%MAXSORT
praintf("vetor[%d]=%d",i+1,vetor[i]);
{
do
{
indicador=0;
for(x=0;x<MAX-1;x++)
{
if(vetor[x]>vetor[x+1])
{
auxiliar=vetor[x];
vetor[x+1]=vetor[x];
vetor[x]=auxiliar;
indicador=indicador+1;
}
if(indicador==0)
break;
}
for(x=MAX-1;x>1;x-)
{
```

```
if(vetor[x]<vetor[x-1];
{
auxiliar=vet[x];
vetor[x]=vetor[x-1];
vetor[x-1]=auxiliar;
indicador=indicador+1;
}
if(indicador==0)
break;
}
}while(indicador!=0);
for(x=1;x<MAX;x++)
printf("vetor[%d]=%d",x,vetor[x]);
return EXIT_SUCCESS;
}
```

Como já foi dito, a diferença do Cocktail Sort para o Bubble Sort consiste que o primeiro percorre a lista de lementos numéricos na ida e na volta enquanto que o segundo percorre apenas na ida. Abaixo está o procedimento do Cocktail Sort, utilizaremos a mesma sequencia :

1. vetor[1]=3
   vetor[2]=35
   vetor[3]=47
   vetor[4]=23

2. indicador=0
3. vetor[1]=3 > vetor[2]=35 ? Falso
4. vetor[2]=35 > vetor[3]=47 ? Falso
5. vetor[3]=47 > vetor[4]=23 ? Verdadeiro
6. auxiliar=vetor[4]
7. vetor[4]=vetor[3]
8. vetor[3]=auxiliar
9. indicador=0+1=1

   Nova lista
   vetor[1]=3
   vetor[2]=35
   vetor[3]=23
   vetor[4]=47

10. vetor[4]=47 < vetor[3]=23 Falso
11. vetor[3]=23 < vetor[2]=35 Verdadeiro
12. auxiliar=vetor[2]
13. vetor[2]=vetor[3]
14. vetor[3]=auxiliar
15. indicador=1+1=2

    Nova Lista

    vetor[1]=3
    vetor[2]=23
    vetor[3]=35
    vetor[4]=47

16. vetor[2]=23<vetor[1]=3 ? Falso
17. indicador !=0 ? Verdadeiro

    Nova Lista

    vetor[1]=3
    vetor[2]=23
    vetor[3]=35
    vetor[4]=47

18. indicador = 0
19. vetor[1]=3 > vetor[2]=23 ? Falso
20. vetor[2]=23 > vetor[3]=35 ? Falso
21. indicador !=0 ? Falso
22. Imprimir lista na tela
23. Finaliza o programa

Comparando os dois algorítmos vemos que o Cocktail Sort tem um percurso mais elegante, nesse caso a diferença de comandos foi pequena, o Bubble Sort teve 25 comandos e o Cocktail teve 21, 4 unidades de diferença. Normalmente se pensa que algo tão pequeno possa ser desprezível mas devemos lembrar que a nosso vetor tinha apenas 4 células, que foi colocado proprositalmente para que a explicação do algorítmo não ficasse tão longa e mais didática.

Porem, se o algorítmo for rodado para um vetor maior, contendo 10000 células, por exemplo, então essas pequenas diferênças quando somadas terão enorme significância. Utilizando o comando `time` do linliux podemos medir o tempo de execução, sua sintaxe ficou:

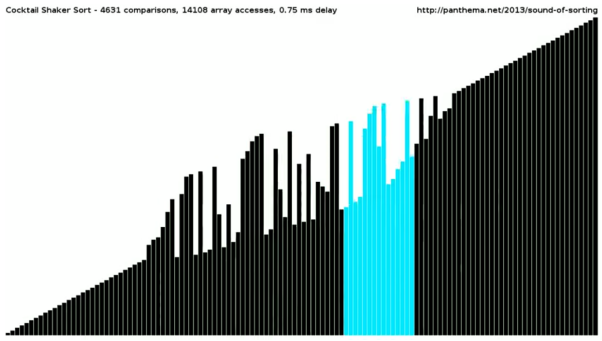
```
time ./algoritmobubblesort.x
time ./algoritmococktailsort.x
```

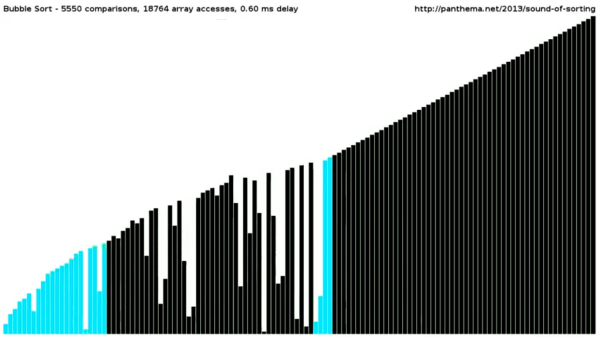
Tabela 1: Título

	Bubble Sort	Cocktail Sort	Diferença [s]
Tempo de execução N=10	0.007	0.007	0
Tempo de execução N=100	0.008	0.008	0
Tempo de execução N=500	0.012	0.010	0.002
Tempo de execução N=1000	0.019	0.017	0.006
Tempo de execução N=5000	0.951	0.108	0.843
Tempo de execução N=10000	1.856	0.313	1.543
Tempo de execução N=50000	20.141	9.772	10.369
Tempo de execução N=100000	1m4.138	37.518	42,465

Acima está uma tabela relacionando o número de células N do vetor com o tempo de processamento do algorítmo. Nesse experimento a quantidade de células no vetor varia e o valor máximo (MAXSORT) foi mantido 100. Até 1000 células no vetor a diferença de tempo entre Buble sort e Cocktail Sort não é tão notável, no entanto a partir de 5000 células fica abservada uma diferença considerada.

O motivo de ser mais rápido é que no processo de volta alguns dos valores desordenados ficam mais estruturados. Abaixo está um figura ilustrando os dois processos caso tivessem sido interrompidos na metade do tempo, o Cacktail Sort na esquerda e Bubble Sort na direita. Podemos observar que o Cocktail Sort consegue estruturar melhor as duas extremidades enquanto que o Buble Sort estrutura apenas a extremidade direita. Levando em consideração a propriedade de que o Cocktail Sort é mais rápido, ele conseguirá estruturar melhor os dados no sentido extremidades $\rightarrow$ meio, caso o processo seja interrompido na metade ou que não se tenha tempo para organizar tudo, o programa do Cocktail Sort entregará um resultado mais bem estruturado que o Buble Sort, se tornando então mais prático.





0.5 Referências

DAMAS, Luiz. Linguagem C. 10 ed. LTC

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. Fundamentos de metodologia científica. 3. ed. rev. e ampl. São Paulo: Atlas, 1991. 270 p.

ALVES, Maria Bernardete Martins; ARRUDA, Suzana Margret de. Como elaborar um Artigo Científico.

BARROS, Aidil Jesus; Lehfeld, Neide Aparecida. Fundamentos da Metodologia Científica. 2 ed. São Paulo, Person Education