Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

DCC001
ANÁLISE E PROJETO DE ALGORITMOS

Trabalho Prático

Rafael Terra
Mateus Coutinho Marim
Aleksander Yacovenco
Matheus Soares

Professor - Stênio Soares

Juiz de Fora - MG
24 de abril de 2017

Sumário

Lista de Figuras

Lista de Programas

Lista de Tabelas

1 Introdução

Escrever aqui a introdução do trabalho...

1.1 Considerações iniciais

- Ambiente de desenvolvimento do código fonte: Code Blocks (por exemplo).
- Linguagem utilizada: Linguagem C.
- Ambiente de desenvolvimento da documentação: TeXnicCenter 1 BETA 7.50-Editor de LaTeX.

1.2 Especificação do problema

Você deverá implementar um tipo abstrato de dados TVetor para representar vetores no espaço R^n. Esse tipo abstrato deverá armazenar a dimensão do vetor e suas respectivas componentes. Considere que a dimensão dos vetores será determinada em tempo de execução.

2 Algoritmo e estruturas de dados

Em [1], são apresentadas estruturas de dados...
O código resultante desse processo será apresentado no Programa 1.

```
//Timer
//Inicializa a contagem
void tStartTimer(stopWatch *timer)
{
  QueryPerformanceCounter(&timer->start);
}
//Para a contagem
void tStopTimer(stopWatch *timer)
{
  QueryPerformanceCounter(&timer->stop);
}
//Converte o tempo computado pelo stopWatch para segundos
double tLIToSecs(LARGE_INTEGER *L)
{
  LARGE_INTEGER frequency;
  QueryPerformanceFrequency(&frequency);
  return ((double)L->QuadPart /(double)frequency.QuadPart);
}
//Retorna o numero de segundos passados na contagem
double tGetElapsedTime(stopWatch *timer)
{
  LARGE_INTEGER time;
  time.QuadPart = (timer->stop).QuadPart - (timer->start).QuadPart;
  return tLIToSecs(&time) ;
}
```

Programa 1: Timer

Tabela 1: Dados referentes aos experimentos

Algoritmo	Tempo 1	Tempo 2	Tempo 3
Quicksort	10	20	30
HeapSort	10	60	530
BublleSort	100	100	1000

3 Análise de complexidade dos algoritmos

3.1 Análise do Bubble Sort

O bubble sort faz múltiplas passadas em uma lista, em cada passada ele verifica se um par de elementos adjacentes estão em ordem, caso não estejam, a posição deles é trocada de forma que o maior deles fique após o menor, isso é repetido até que não sejam mais necessárias trocas.

O melhor caso do bubble sort é quando a sua entrada é uma lista ordenada, neste caso os elementos já estão em ordem nenhuma troca é efetuada e o algoritmo termina na primeira passada, logo sua complexidade é na ordem de $O(n)$.

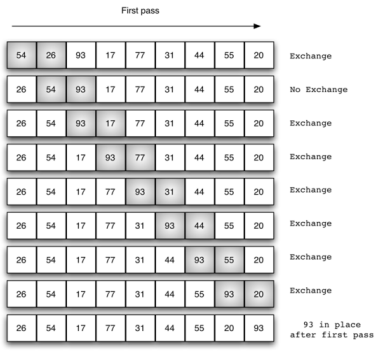

Figura 1: Uma passada do bubble sort.

No pior caso podemos levar em consideração quando os elementos estão ordenados em ordem decrescente, neste caso na iteração 0 o block roda $n-1-0$ vezes, na iteração 1 ele roda $n-1-(n-1)=0$ vezes. Então no total, o bloco roda a quantidade de vezes expressa na Equação 1.

$$O(n) = \sum_{i=0}^{n-1} n - i - 1 = n^2 - \sum_{i=0}^{n-1} i - n = n^2 - n*(n-1)/2 = n^2/2 - n/2 \qquad (1)$$

Dando um pior caso na ordem de $O(n^2)$.

4 Testes

Estas estruturas são apresentadas na Figura 1.

5 Conclusão

Neste trabalho foram revistos conceitos sobre...[2].
Muito dos algoritmos são extraídos de:[3].

Referências

[1] Rasmus Pagh. Hash and displace: Efficient evaluation of minimal perfect hash functions. In *Workshop on Algorithms and Data Structures*, pages 49–54, 1999.

[2] Gabriel Torres. Clube do hardware, 2009. `http://clubedohardware.com.br`, visitado em 08/08/2009.

[3] N. Ziviani. *Projeto de Algoritmos: com implementações em Pascal e C*. Cengage Learning (Thomson / Pioneira), São Paulo, 1st edition, 2004.