Algoritmos de Ordenação

SCC0201 - Introdução à Ciência de Computação II

Leandro A. Amaral Tiago S. Nazaré Vanessa Q. Marinho

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

22 de junho de 2015

Agenda

1. Merge Sort
   - Introdução
   - Exemplo
   - Código
   - Análise de Complexidade

Agenda

1. Merge Sort
   - Introdução
   - Exemplo
   - Código
   - Análise de Complexidade

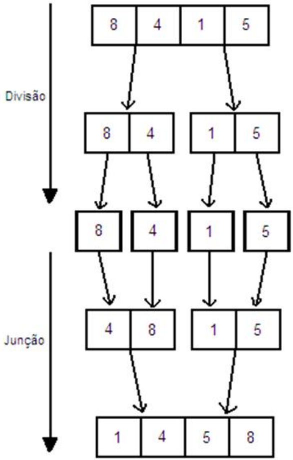
Introdução

- Técnica “dividir-para-conquistar”
- Constituido por duas diferentes fases:
  - Divisão
  - Junção (merge)
- Não é feita nenhuma computação na fase de divisão
- A ordenação acontece na fase de junção (merge)
- Vídeo: http://www.youtube.com/watch?v=PKCMMSXQyJE

Agenda

1. Merge Sort
   - Introdução
   - Exemplo
   - Código
   - Análise de Complexidade

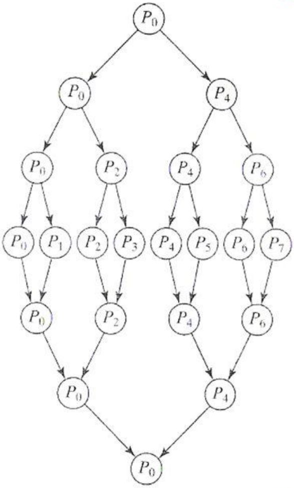
Exemplo - Merge Sort Sequencial



Exemplo - Merge Sort Paralelo

- Na fase de divisão, atribui-se uma lista de elementos a cada processador
- Tem o problema de em algumas partes do algoritmo ser necessária bastante comunicação entre processadores
- Tem a vantagem de ser bastante leve a nível computacional em cada processador

Exemplo - Merge Sort Paralelo



Agenda

1. Merge Sort
   - Introdução
   - Exemplo
   - Código
   - Análise de Complexidade

Merge Sort - Código

```
void partition(int arr[],int low,int high) {

    int mid;

    if(low<high){
        mid=(low+high)/2;
        partition(arr,low,mid);
        partition(arr,mid+1,high);
        mergeSort(arr,low,mid,high);
    }
}

void mergeSort(int arr[],int low,int mid,int high) {

    int i,m,k,l,temp[MAX];

    l=low;
    i=low;
    m=mid+1;
```

Merge Sort - Código

```
        while((l<=mid)&&(m<=high)){

            if(arr[l]<=arr[m]){
                temp[i]=arr[l];
                l++;
            }
            else{
                temp[i]=arr[m];
                m++;
            }
            i++;
        }

        if(l>mid){
            for(k=m;k<=high;k++){
                temp[i]=arr[k];
                i++;
            }
        }
        else{
            for(k=l;k<=mid;k++){
                temp[i]=arr[k];
                i++;
            }
        }

        for(k=low;k<=high;k++){
            arr[k]=temp[k];
        }
    }
```

Agenda

1. Merge Sort
   - Introdução
   - Exemplo
   - Código
   - Análise de Complexidade

Análise de Complexidade - ($\mathcal{O}(nlogn)$)

Segundo Drozdek (2002), o pior caso é quando o último elemento da metade precede somente o último elemento da outra metade, como por exemplo: [1,6,10,12] e [5,9,11,13]. Para uma lista de n elementos, o número de movimentos é calculado pela seguinte relação de recorrência:

$$M(1) = 0$$
$$M(n) = 2M(n/2) + 2n$$

Análise de Complexidade - ($\mathcal{O}(nlogn)$)

M(n) pode ser calculado do seguinte modo:
$\mathsf{M(n) = 2(2M(n/4) + 2(2/n)) + 2n = 4M(n/4) + 4n}$
$\mathsf{= 4(2M(n/8) + 2(n/4) + 4n) = 8M(n/8) + 6n}$

...

$\mathsf{= 2iM(n/2i) + 2in}$

Escolher $\mathsf{i = logn}$, de modo que $\mathsf{n = 2i}$ permite inferir que:
$\mathsf{M(n) = 2iM(n/2i) + 2in = nM(1) + 2nlogn = 2nlogn = O(nlogn)}$

Análise de Complexidade - Pior Caso ($\mathcal{O}(nlogn)$)

O número de comparações, no pior caso, é dado por uma relação similar:
$C(1) = 0$
$C(n) = 2C(n/2) + n-1$

Que também resulta em $C(n)$, sendo $O(nlogn)$

Desvantagens

- Utiliza funções recursivas
- Gasto extra de memória, o algorítimo cria uma cópia do vetor para cada nível da chamada recursiva, totalizando um uso adicional de memória igual a (n log n)

Exercício

Escreva e analise uma versão iterativa do algoritmo Mergesort.

Resposta

Sedgewick chama essa versão de "bottom-up Mergesort", ou seja, "Mergesort de-baixo-para-cima". Ele chama a versão recursiva de "top-down Mergesort".

```
#define min(A, B) (A < B) ? A : B

// Esta funcao rearranja o vetor a[p..r] em ordem crescente.

void mergesortBU(itemType a[], int p, int r) {
    int i, m;
    for (m = 1; m <= r-p; m = m+m)
       for (i = p; i <= r-m; i += m+m)
          partition(a, i, i+m-1, min(i+m+m-1, r));
}
```