MC-102 — Aula 17
Busca e ordenação Recursiva

Instituto de Computação – Unicamp

1er Semestre de 2011

Roteiro

1. Busca binária recursiva
2. MergeSort

Relembrando Busca binária - Definição

- Caso uma lista A esteja ordenada, sabemos que, para qualquer i e j, $i < j$ se, e somente se $A[i] \leq A[j]$.
- Portanto, comparando um elemento com o elemento procurado, saberemos:
  - se o elemento procurado é o elemento comparado, ou
  - se ele está antes do elemento comparado ou
  - se ele está depois.

Busca binária

- Se fizermos isso sempre com o elemento do meio da lista, a cada comparação, dividiremos a lista em duas, reduzindo nosso espaço de busca.
- Se em um determinado momento o vetor, após sucessivas divisões, tiver tamanho zero, então o elemento não está no vetor.

Busca binária - iterativa

- Passo 1: inicialização

```
int direita, esquerda, meio;
encontrado = 0; /*Falso*/

esquerda = 0;
direita = TAMANHO - 1;
```

Busca binária - iterativa

- Passo 2: busca

```
while(esquerda<=direita && !encontrado){
 meio=(direita+esquerda)/2;
   if (vetor[meio] == valor)
    encontrado = 1; /*Verdadeiro*/
   else if (valor < vetor[meio])
     direita = meio - 1;
    else
     esquerda = meio + 1;
}
```

Busca binária - iterativa

- Passo 3: tratamento do resultado

```
if(encontrado){
 printf ("Valor %d encontrado na posicao %d\n",
          vetor[meio], meio);
}
else{
 printf ("Valor %d não encontrado\n", valor);
}
```

Busca binária - recursiva

```
int busca(int vet[], int i, int f, int v) {
 int k;
 if (i > f)
   return -1;
 else {
   k = (i+f)/2;
   if (vet[k] == v)
     return k;
   else
     if (v < vet[k])
      return busca(vet,i,k-1, v);
     else
      return busca(vet,k+1,f, v);
  }
}
```

MergeSort

- Uma técnica para resolver um problema é dividi-lo em parte menores, para as quais saibamos resolver o problema, e a partir desses resultados, resolver o problema maior. Essa técnica é chamada de divisão e conquista.
- Os três passos úteis dos algoritmos dividir-para-conquistar, que se aplicam ao Mergesort são:
  - Dividir: Dividir os dados em subsequências pequenas;
  - Conquistar: Classificar as duas metades recursivamente aplicando o merge sort;
  - Combinar: Juntar as duas metades em um único conjunto já classificado.

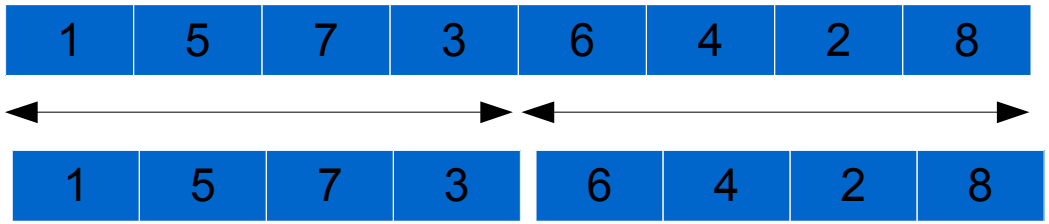
MergeSort

- A idéia do MergeSort (ordenação por intercalação) é dividir o vetor de entrada em dois subvetores com metade do tamanho do vetor original (em caso de tamanho ímpar um dos subvetores terá um elemento a mais que o outro). Cada um dos subvetores é ordenado recursivamente
- Os dois subvetores são intercalados em um vetor temporário.
- A partir dessas listas ordenadas, utiliza-se a intercalação para reunir as duas listas.

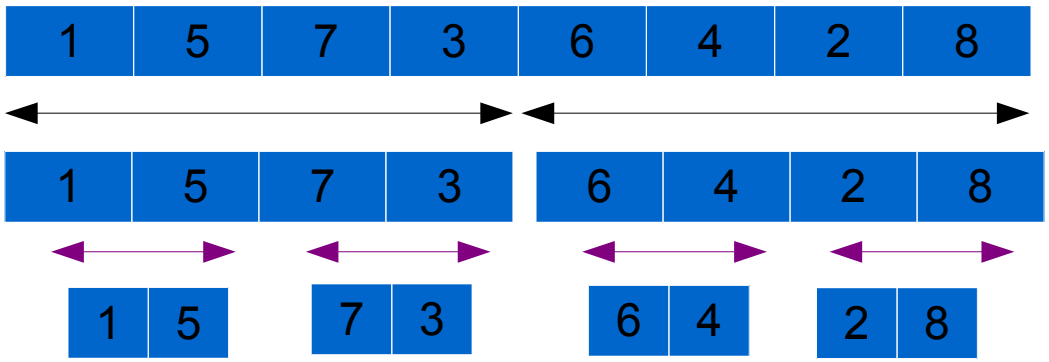
Veja detalhes em `mergesort.c`

Exemplo MergeSort

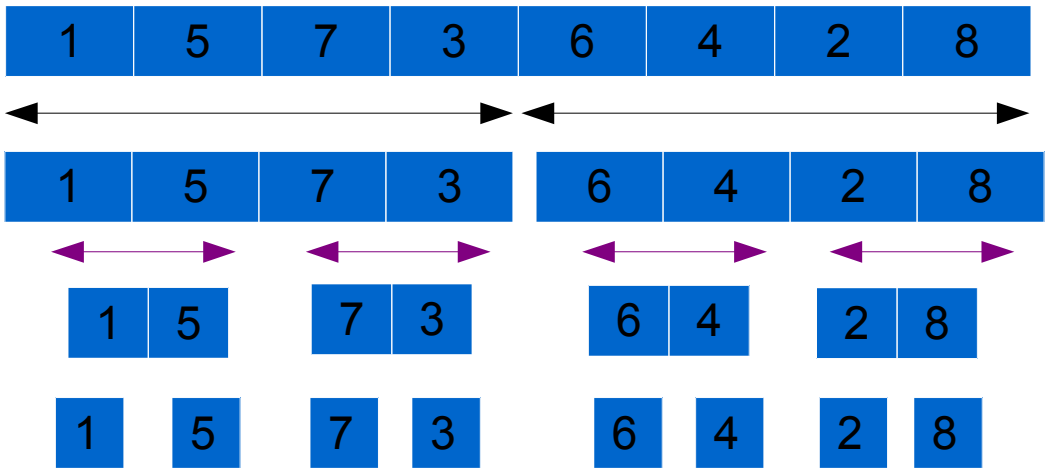
Exemplo MergeSort



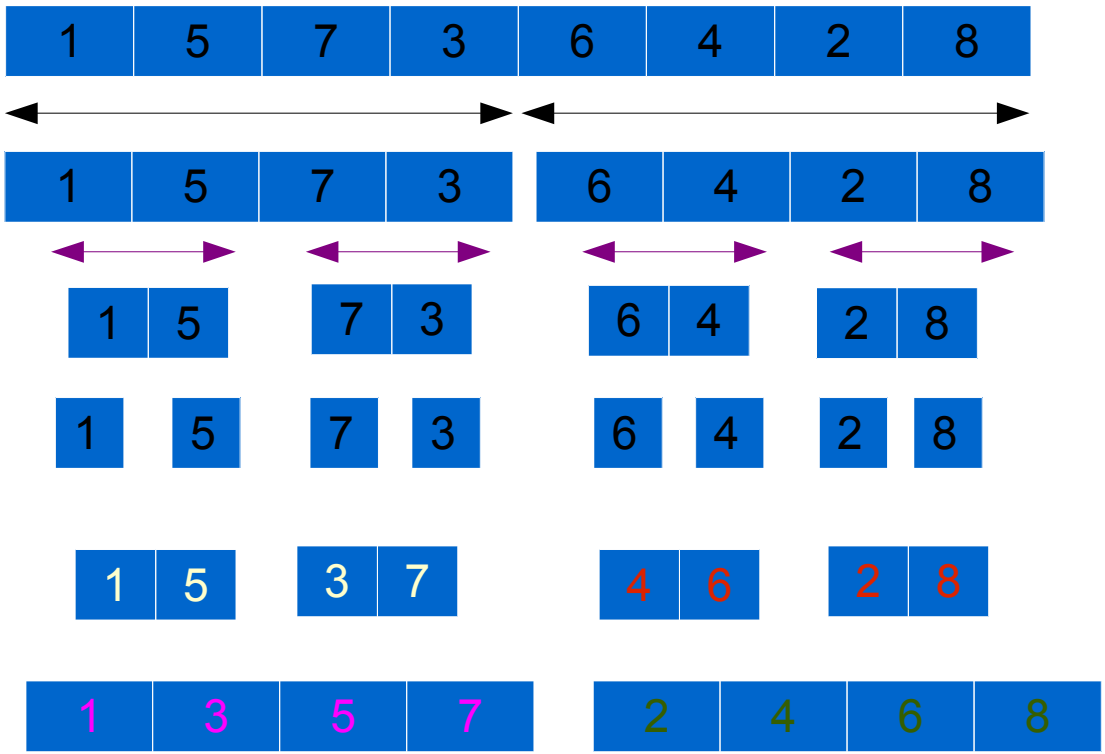
Exemplo MergeSort



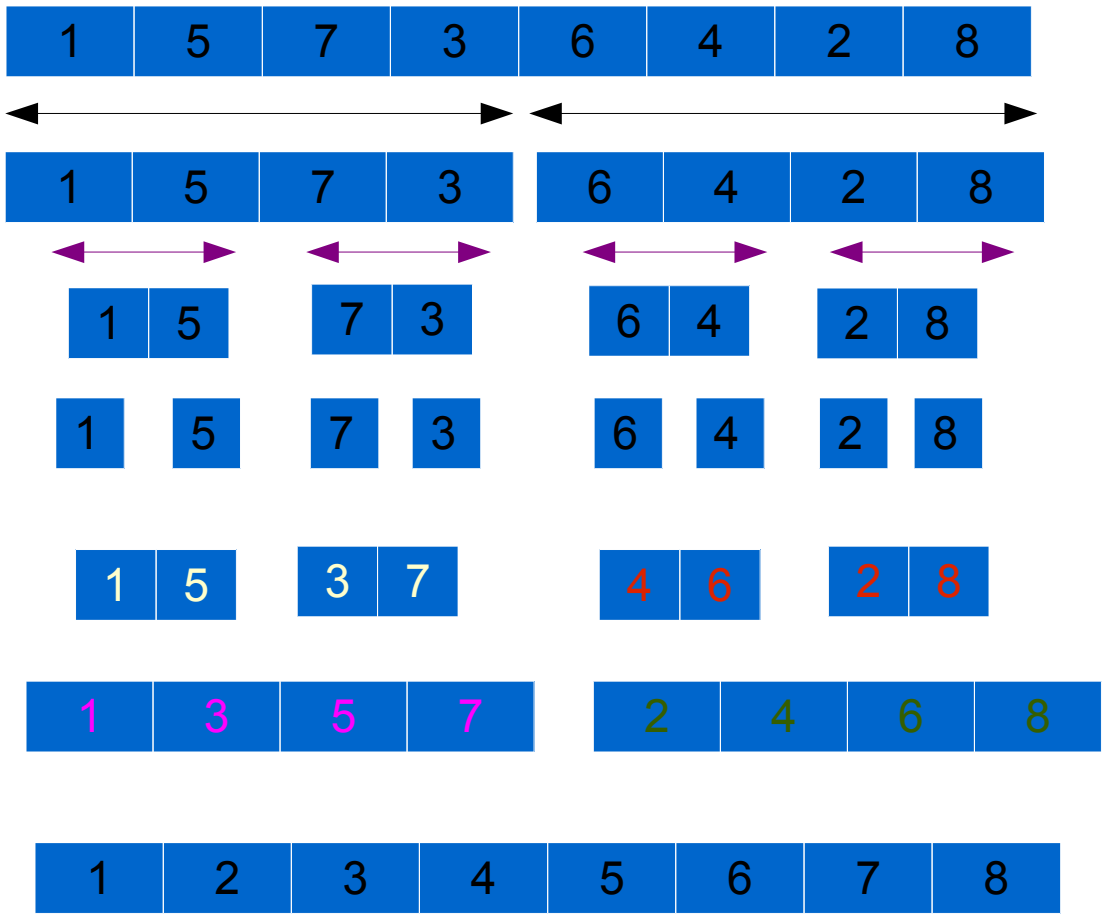
Exemplo MergeSort



Exemplo MergeSort

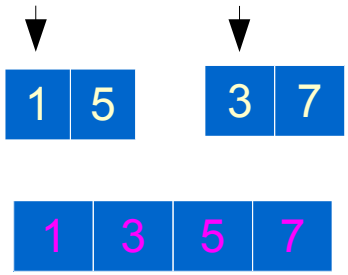


Exemplo MergeSort



MergeSort

- Número máximo de comparações?



(1,3) → 1
(5,3) → 3
(5,7) → 5
→ 7

3 Comparações para
4 elementos (n=4)

Ou seja máximo de comparações
n-1

MergeSort

- O algoritmo *mergesort* executa $n \log n$ comparações, que é o menor número possível de ordenações para algoritmos sem qualquer tipo de restrição e baseados em comparações. (nenhum dos dois fatos da sentença acima pode ser demonstrado de forma trivial).
- Entretanto, ele necessita de um espaço de memória extra proporcional ao tamanho do vetor para fazer a intercalação.

Referências

- http://www.cse.iitk.ac.in/users/dsrkg/cs210/applets/sortingII/mergeSort/mergeSort.html
- http://www.inf.ufsc.br/ ronaldo/ine5384/18-OrdenacaoDados4.pdf
- http://pt.wikipedia.org/wiki/Merge_sort
- http://www.liv.ic.unicamp.br/ bergo/mc102e/mqsort.pdf