Técnicas de Concepção de Algoritmos (1ª parte): divisão e conquista

R. Rossetti, A.P. Rocha, J. Pascoal Faria
CAL, MIEIC, FEUP
Fevereiro de 2014

Divisão e Conquista (*divide and conquer*)

3

Divisão e conquista

- Divisão: resolver recursivamente problemas mais pequenos (até caso base)
- Conquista: solução do problema original é formada com as soluções dos subproblemas
- Há divisão quando o algoritmo tem pelo menos 2 chamadas recursivas no corpo
- Subproblemas devem ser disjuntos
  - Senão, resolver de forma bottom-up com programação dinâmica
- Divisão em subproblemas de dimensão semelhante é importante para se obter uma boa eficiência temporal
- Algoritmos adequados para processamento paralelo

Técnicas de Concepção de Algoritmos, CAL - MIEIC/FEUP, Fev. de 2012

4

Divisão e conquista

- Dada uma instância do problema x, a técnica Divisão-e-Conquista funciona da seguinte maneira:

```
function DAQ( x )
   if x é suficientemente pequeno then
      resolver x directamente
   else
      dividir x em subinstâncias: x1, …, xk
      for i := 1 to k do yi := DAQ( xi )
      y := Σ yi
      return y
```

Técnicas de Concepção de Algoritmos, CAL - MIEIC/FEUP, Fev. de 2012

5

Exemplo: cálculo de x^n

- Resolução iterativa com *n* multiplicações: T(n) = O(n)
- Resolução mais eficiente, com divisão e conquista:

$$x^n = \begin{cases} 1, \text{ se } n = 0 \\ x, \text{ se } n = 1 \\ x^{\frac{n}{2}} \times x^{\frac{n}{2}}, \text{ se } n \text{ par} > 1 \\ x \times x^{\frac{n-1}{2}} \times x^{\frac{n-1}{2}}, \text{ se } n \text{ ímpar} > 1 \end{cases}$$

```
double power(double x, int n) {
  if (n == 0) return 1;
  if (n == 1) return x;
  double p = power(x, n / 2);
  if (n % 2 == 0) return p * p;
  else return x * p * p;
}
```

- Divisão em subproblemas iguais, junção em tempo O(1)
- Nº de multiplicações reduzido para aprox. $\log_2 n$
- T(n) = O(log n) …. mas S(n) = O(log n) (espaço)

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

6

Exemplo: ordenação de *arrays*

- Mergesort
  - Ordenar 2 subsequências de igual dimensão e juntá-las
  - T(n) = 2 T(n/2) + k n (n>1) =>
  - T(n) = O(n log n), tanto no pior caso como no caso médio
- Quicksort
  - Ordenar elementos menores e maiores que *pivot*, concatenar
  - T(n) = O(n^2) no pior caso (1 elemento menor, restantes maiores)
  - T(n) = O(n log n) no melhor caso e no caso médio (*)
    - (*) com escolha aleatória do pivot!

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

7

Exemplo: *Mergesort*

- Seja S = {s_1, ..., s_n} um conjunto que se pretenda ordenar. Se S = ∅ ou S = {s}, então nada é necessário!
- Dividir: remover todos os elementos de S e colocá-los em duas subsequências: S_1 e S_2, cada uma com ~n/2 elementos
- Conquistar: consiste em ordenar S_1 e S_2, utilizando mergesort
- Combinar: colocar os elementos de volta em S, unindo as sequências ordenadas S_1 e S_2 numa sequência ordenada única.

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

8

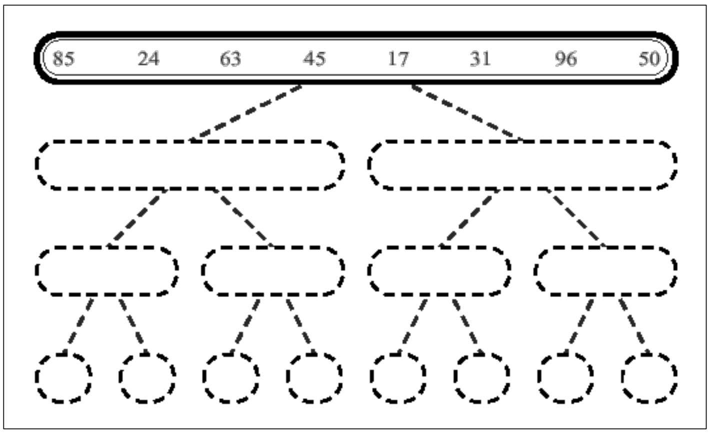
Exemplo: *Mergesort*

```
Merge-Sort(A, p, r)
   if p < r then
      q←(p+r)/2
      Merge-Sort(A, p, q)
      Merge-Sort(A, q+1, r)
      Merge(A, p, q, r)
```

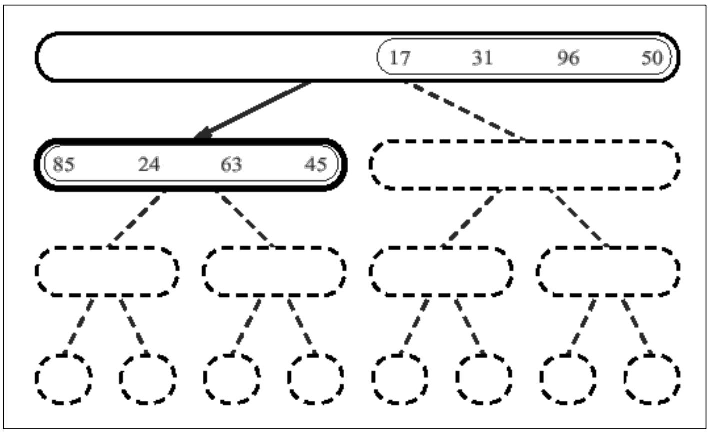
```
Merge(A, p, q, r)
   Take the smallest of the two topmost elements of
sequences A[p..q] and A[q+1..r] and put into the
resulting sequence. Repeat this, until both sequences
are empty. Copy the resulting sequence into A[p..r].
```

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

Exemplo: *Mergesort*

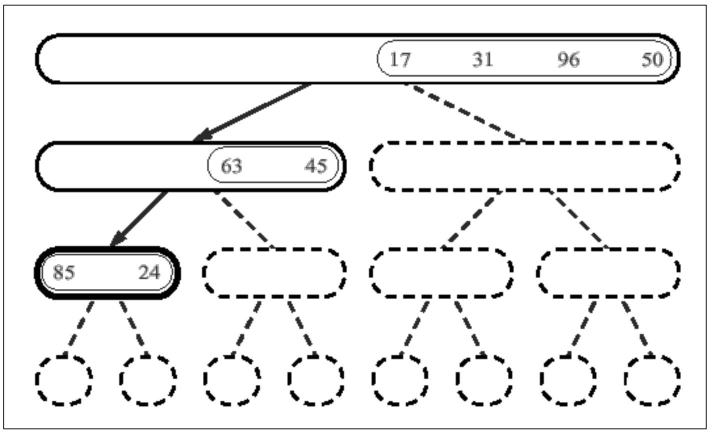


Exemplo: *Mergesort*



11

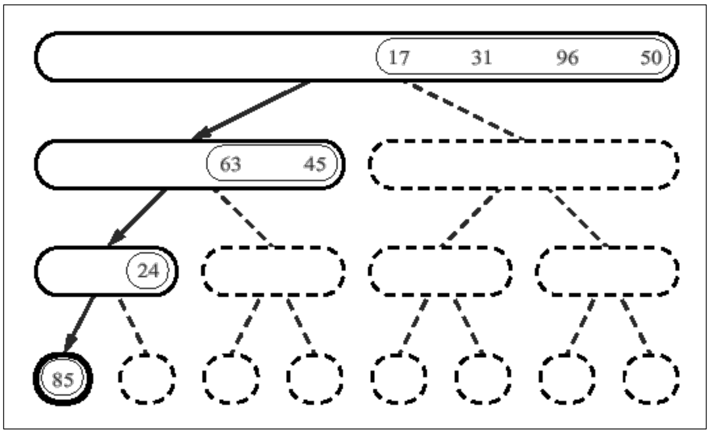
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

12

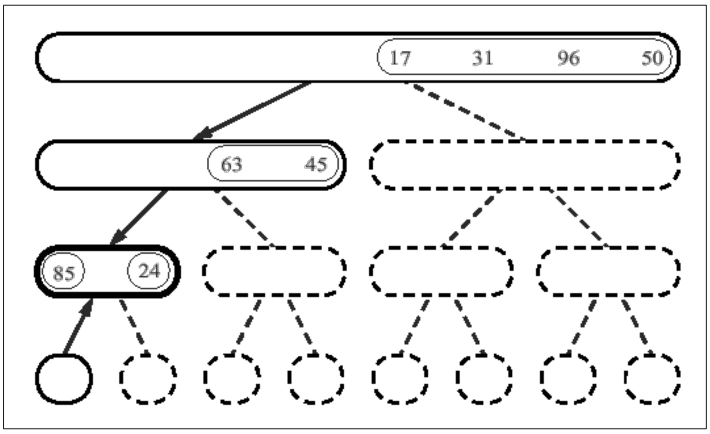
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

13

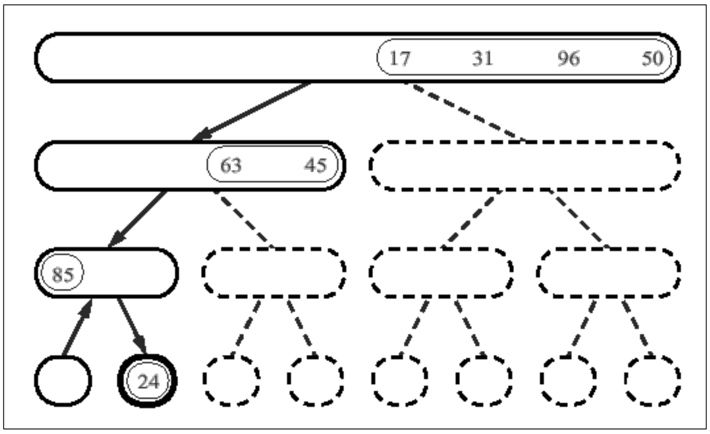
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

14

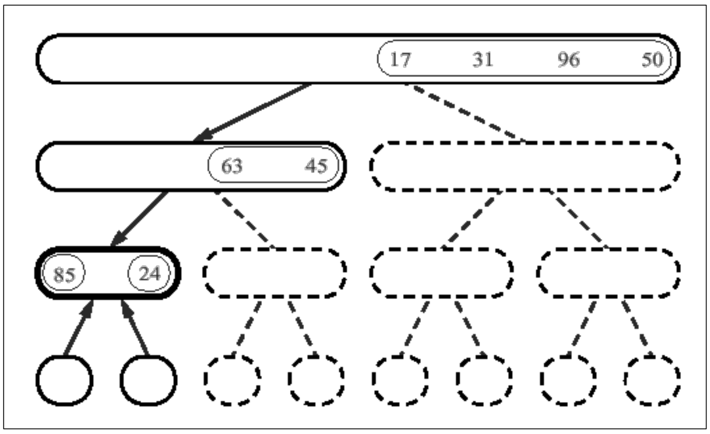
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

15

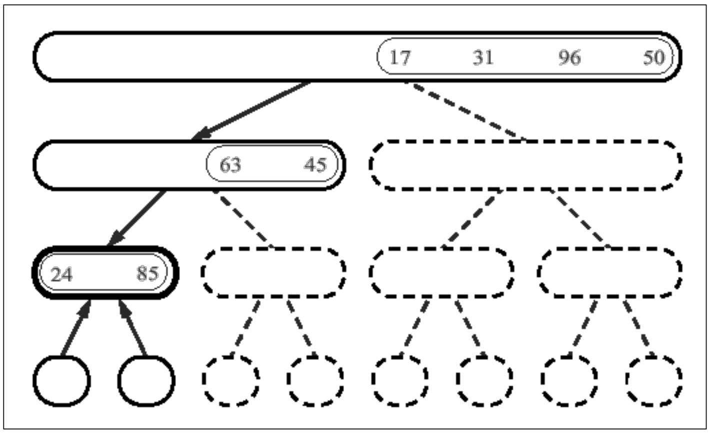
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

16

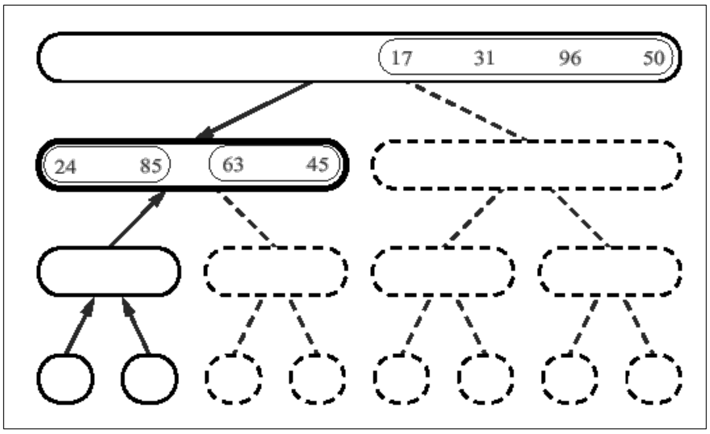
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

17

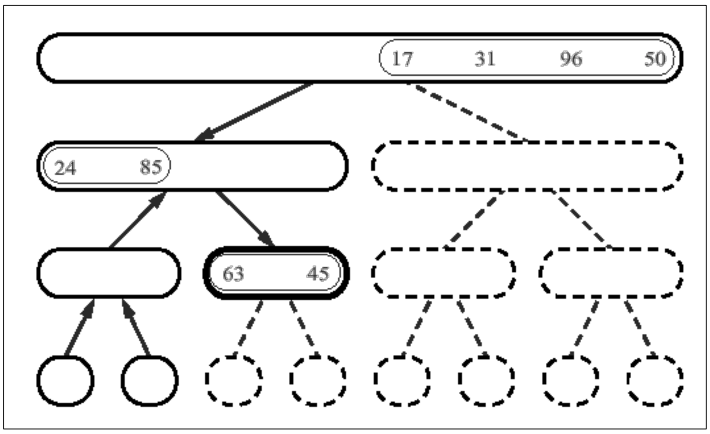
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

18

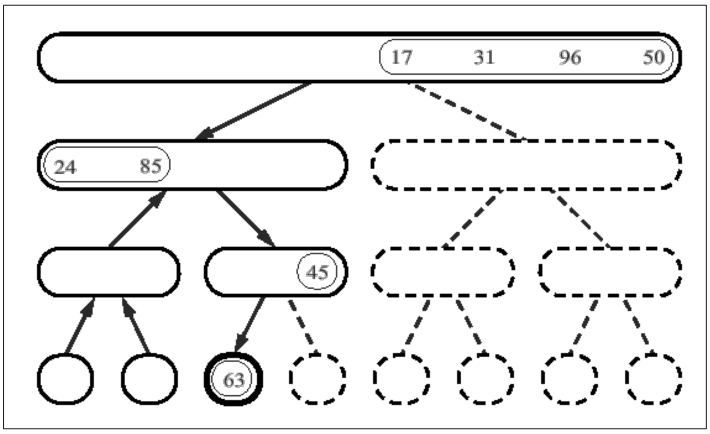
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

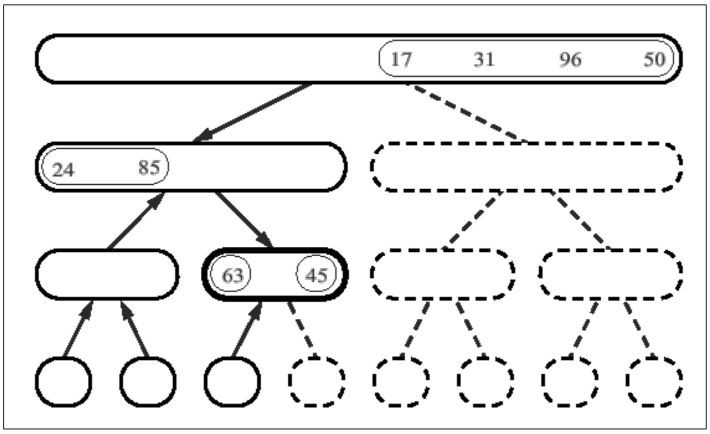
19

Exemplo: *Mergesort*



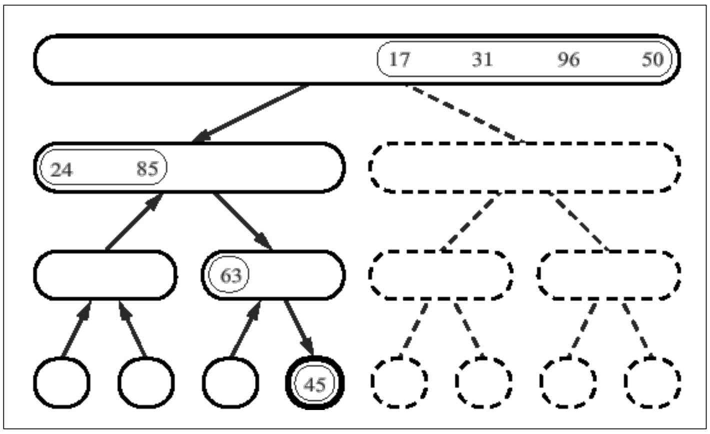
20

Exemplo: *Mergesort*



21

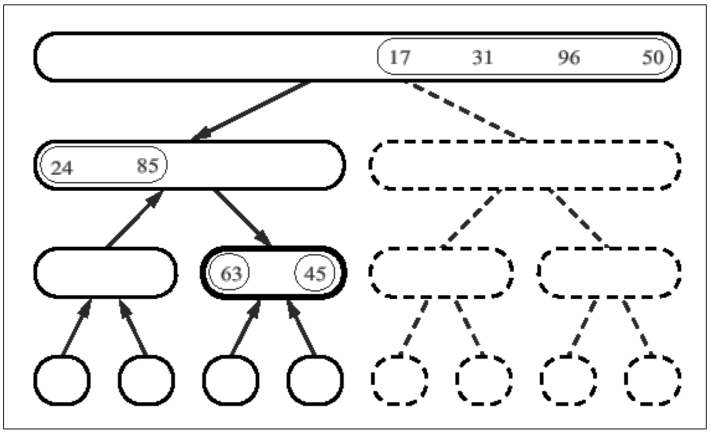
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

22

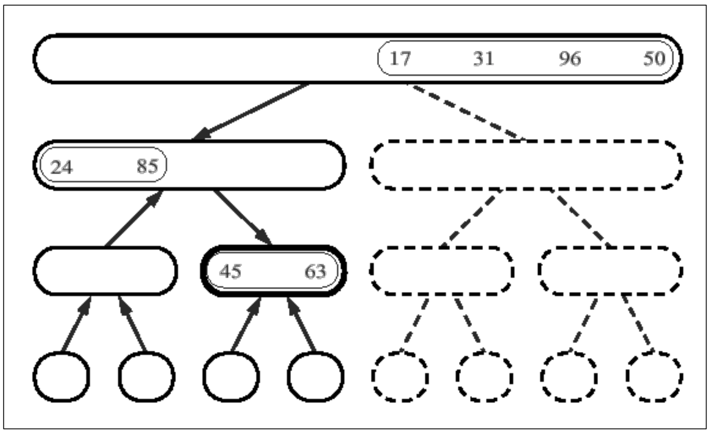
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

23

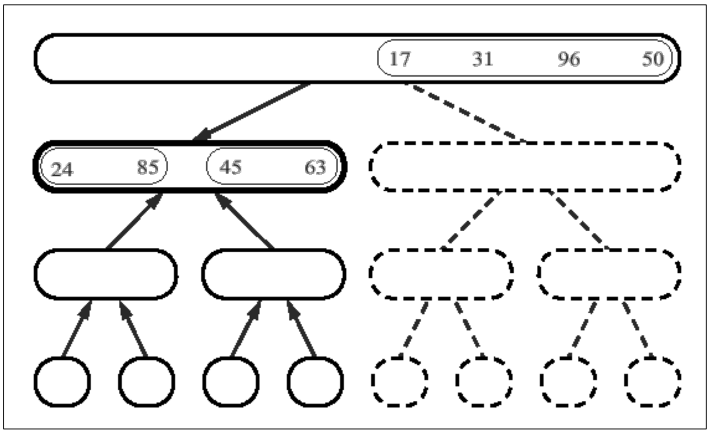
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

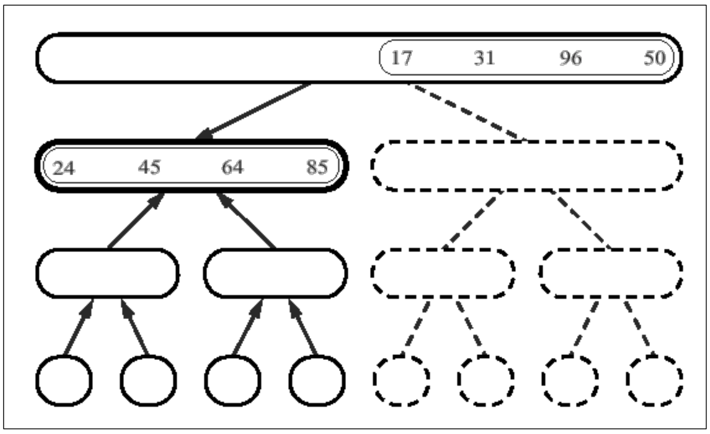
24

Exemplo: *Mergesort*

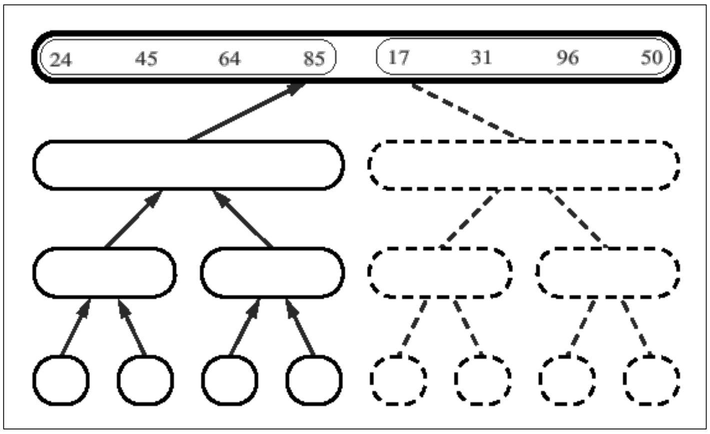


Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

Exemplo: *Mergesort*

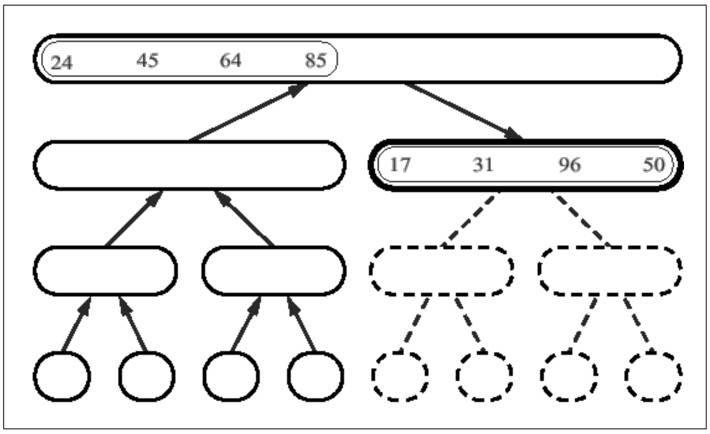


Exemplo: *Mergesort*



27

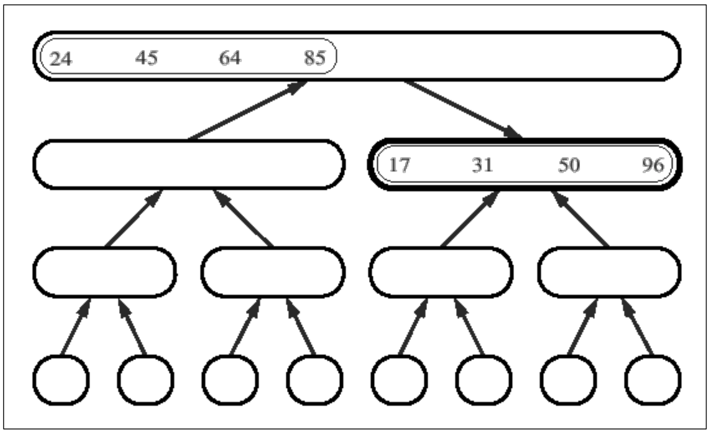
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

28

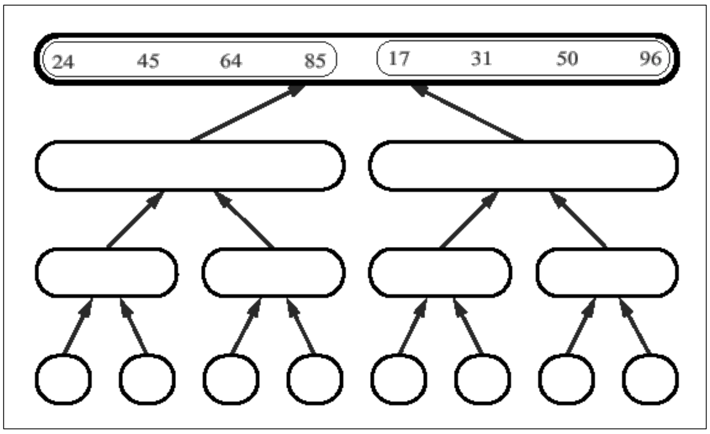
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

29

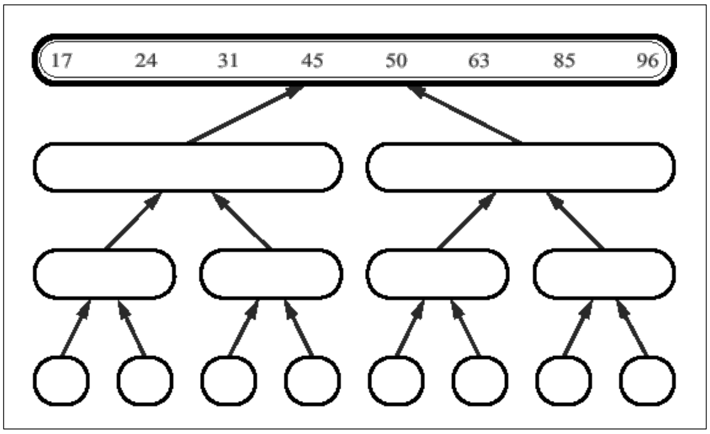
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

30

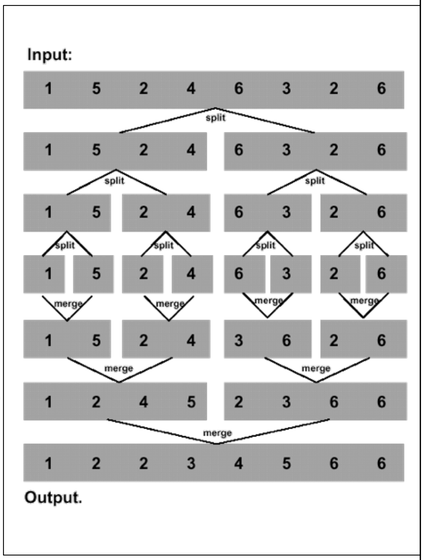
Exemplo: *Mergesort*



Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

Exemplo: *Mergesort*

- P/ ordenar *n* elementos:
  - Se *n* = 1, está feito!
  - Recursivamente ordenar 2 listas de ⌊n/2⌋ elementos
  - Combinar as duas listas ordenadas em tempo Θ(*n*)
- Estratégia:
  - Dividir o problema em sub-problemas menores
  - Resolver recursivamente
  - Combinar as subsoluções



Exemplo: pesquisa binária

- Seja S uma sequência ordenada de *n* elementos, e s_x um elemento que se pretende procurar dentro de S.
- Se s_x é o elemento médio, então retorna-se s_x!
- Senão:
  - Dividir: divide-se S em duas sequências, S_1 e S_2, com ~*n/2* elementos; se s_x < que o elemento médio, escolhe-se S_1 para continuar; se s_x > que o elemento médio, escolhe-se S_2 para continuar.
  - Conquistar: tenta-se resolver a subsequência para determinar se s_x está presente.
  - Obtém-se a solução para a sequência a partir da solução do problema para as subsequências!

33

Exemplo: pesquisa binária

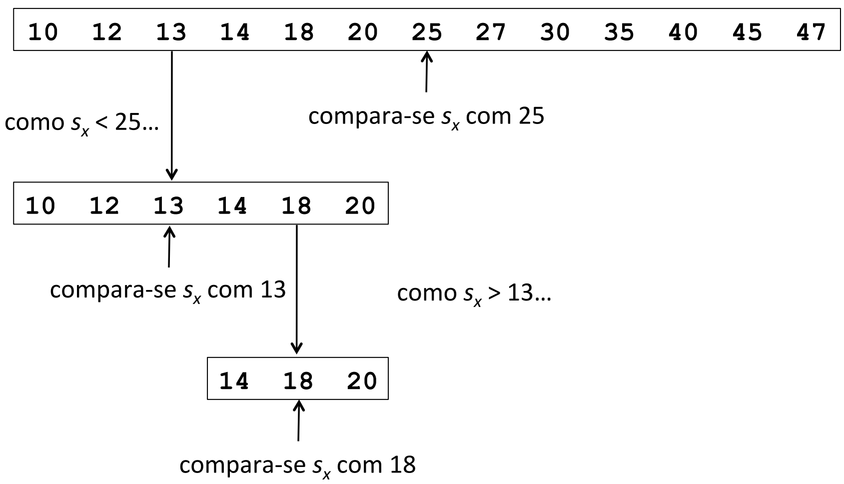
- Suponha que s_x = 18 e que S e dado por:
  {10 12 13 14 18 20 25 27 30 35 40 45 47}

  termo médio

- Dividir a sequência: sendo s_x < 25, pequisa-se em
  {10 12 13 14 18 20}
- Conquistar a subsequência, determinando-se se s_x está presente.
- Obtém-se a solução para a sequência S, pela solução da pesquisa nas subsequências. R: Sim! $s_x \in$ S

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

34



Conclui-se que s_x está presente
pois s_x = 18!

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

35

Problema *x*
de tamanho *n*.

Dados: inteiro positivo *n*, array ordenado S, indexado de 1..*n*,
uma chave *x*.

Resultado: *location*, localização de *x* em S (0 se *x* não está em S)

```
index location(index low, index high)
   index mid;
   if ( low > high )
      return 0;
   else
      mid = [ (low + high ) / 2 ];
      if ( x == S[mid] )
         return mid;
      else if ( x < S[mid] )
         return location(low, mid – 1);
      else
         return location(mid + 1, high);
```

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

36

Referências

- Mark Allen Weiss. Data Structures & Algorithm Analysis in Java. Addison-Wesley, 1999
- Steven S. Skiena. The Algorithm Design Manual. Springer 1998
- Robert Sedgewick. Algorithms in C++. Addison-Wesley, 1992

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

37

Em suma…

- Programação dinâmica (*dynamic programming*)
  - Contexto: Problemas de solução recursiva.
  - Objectivo: Minimizar tempo e espaço.
  - Forma: Induzir uma progressão iterativa de transformações sucessivas de um espaço linear de soluções.
- Algoritmos gananciosos (*greedy algorithms*)
  - Contexto: Problemas de optimização (max. ou min.)
  - Objectivo: Atingir a solução óptima, ou uma boa aproximação.
  - Forma: tomar uma decisão óptima localmente, i.e., que maximiza o ganho (ou minimiza o custo) imediato

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012

38

Em suma…

- Algoritmos de retrocesso (*backtracking*)
  - Contexto: problemas sem algoritmos eficientes (convergentes) para chegar à solução.
  - Objectivo: Convergir para uma solução.
  - Forma: tentativa-erro. Gerar estados possíveis e verificar todos até encontrar solução, retrocedendo sempre que se chegar a um "beco sem saída".
- Divisão e conquista (*divide and conquer*)
  - Contexto: Problemas passíveis de se conseguirem sub-dividir.
  - Objectivo: melhorar eficiencia temporal.
  - Forma: agregação linear da resolução de sub-problemas de dimensão semelhantes até chegar ao caso-base.

Técnicas de Concepção de Algoritmos, CAL – MIEIC/FEUP, Fev. de 2012