UFRR

UNIVERSIDADE FEDERAL DE RORAIMA

Divisão e Conquista

Aula Baseada nos slides do Profº. Rosiane Rodrigues - UFAM

Prof. Dr. Herbert Oliveira Rocha
herberthb12@gmail.com

Divisão e Conquista

Técnicas de Projeto

- Divisão & Conquista
- Programação Dinâmica
- Método Guloso (*Greedy*)
- Enumeração Explícita x Implícita
- Métodos Exatos x Aproximados

Divisão e Conquista

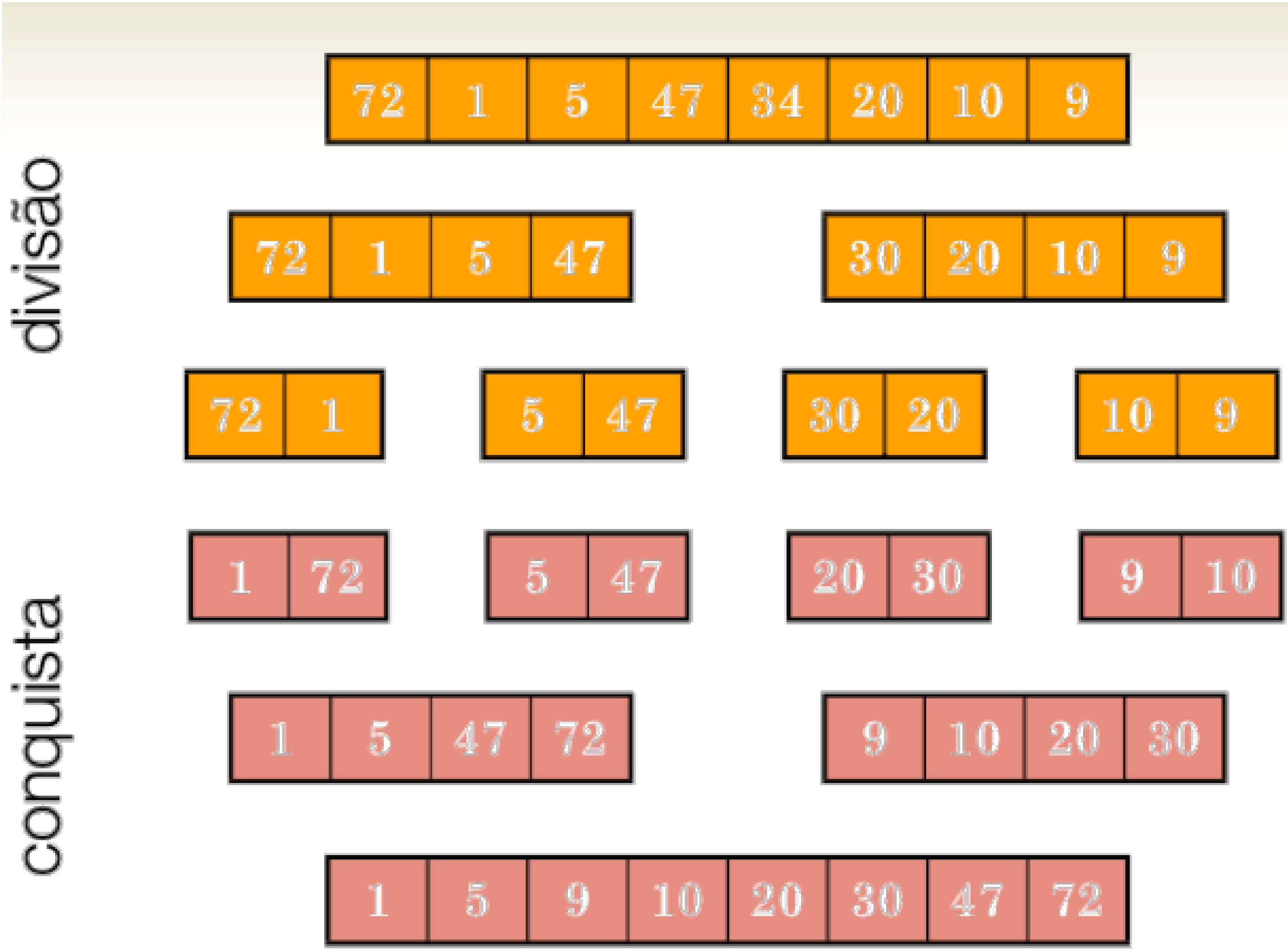
- É preciso revolver um problema com uma entrada grande

- Para facilitar a resolução do problema, a entrada é quebrada em pedaços menores (**DIVISÃO**)

- Cada pedaço da entrada é então tratado separadamente (**CONQUISTA**)

- Ao final, os resultados parciais são combinados para gerar o resultado final procurado

Divisão e Conquista



Divisão e Conquista

A técnica de divisão e conquista consiste de 3 passos:

- **Divisão**: Dividir o problema original em subproblemas menores
- **Conquista**: Resolver cada subproblema recursivamente
- **Combinação**: Combinar as soluções encontradas, compondo uma solução para o problema original

Divisão e Conquista

Divisão e Conquista

- Na **técnica de decomposição** conhecida por divisão&conquista há dois critérios que devem ser levados em consideração para que se obtenha algoritmos mais eficientes:

1) A soma dos tamanhos dos subproblemas deve ser a menor possível.

2) Os subproblemas devem ser de tamanhos tão iguais quanto possível.

Divisão e Conquista

- Na **técnica de decomposição** conhecida por divisão&conquista há dois critérios que devem ser levados em consideração para que se obtenha algoritmos mais eficientes:

1) A soma dos tamanhos dos subproblemas deve ser a menor possível.

2) Os subproblemas devem ser de tamanhos tão iguais quanto possível.

BALANCEAMENTO

Algoritmo Genérico

```
def divisao_e_conquista(x):
    if x é pequeno ou simples:
        return resolve(x)
    else:
        decompor x em n conjuntos menores x0,x1,...,xn-1
        for i in [0,1,...,n-1]:
            yi = divisao_e_conquista(xi)
        combinar y0,y1,...,yn-1 em y
        return y
```

Divisão e Conquista

Problema de Ordenação

Seja $S = s_1, s_2, \ldots, s_n$ uma sequência de elementos comparáveis entre si. O problema consiste em ordenar S.

- Algoritmos
  - *Bublesort.*
  - *Mergesort.*
  - *Quicksort.*
  - *etc*

Divisão e Conquista: Vantagens

- Resolução de problemas difíceis
  - Exemplo clássico: Torre de Hanói
- Pode gerar algoritmos eficientes
  - Ótima ferramenta para busca de algoritmos eficientes, com forte tendência a complexidade logarítmica
- Paralelismo
  - Facilmente paralelizável na fase de conquista
- Controle de arredondamentos
  - Em computação aritmética, divisão e conquista traz resultados mais precisos em operações com pontos flutuantes

Divisão e Conquista: Desvantagens

- Recursão ou Pilha explícita
- Tamanho da Pilha
  - Número de chamadas recursivas e/ou armazenadas na pilha pode ser um inconveniente
- Dificuldade na seleção dos casos bases
- Repetição de sub-problemas
  - Situação que pode ser resolvida através do uso de **memoização**

Mergesort

Seja $S=s_1,s_2,\ldots,s_n$ uma sequência de elementos comparáveis entre si. O problema consiste em ordenar S.

Idéia do método

1 – Decompor o problema (sequência S) de tamanho n em dois subproblemas de tamanhos $n/2$ (piso e teto, respectivamente).

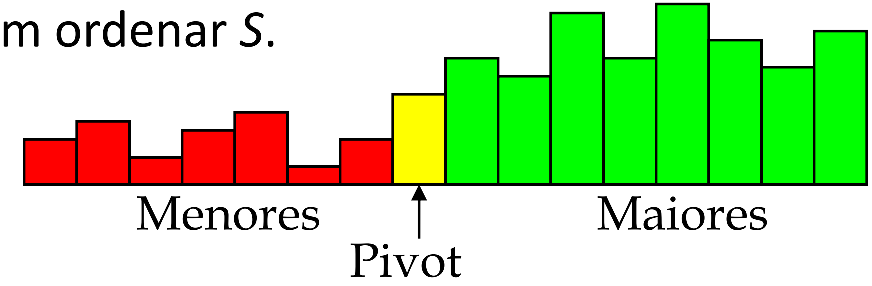
2 – Aplicar o mesmo método (passos 1), recursivamente, para a subsequência formada pelos $n/2$ elementos de cada subproblema até restarem subproblemas de tamanho 1.

3 – Intercalar os elementos dos subproblemas ordenando-os.

Divisão e Conquista

Quicksort

Seja $S=s_1,s_2,\ldots,s_n$ uma sequência de elementos comparáveis entre si. O problema consiste em ordenar S.



Idéia do método

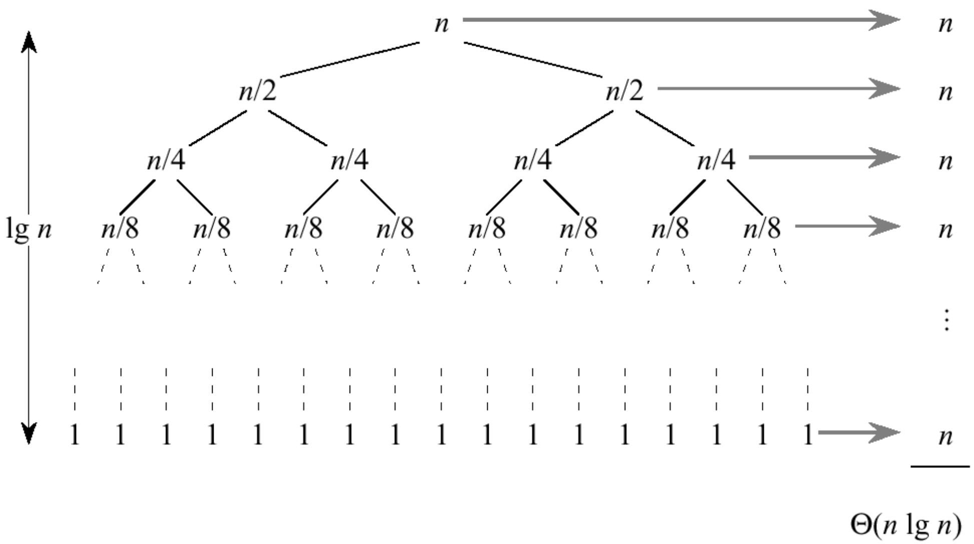
1 – Decompor o problema (sequência *S*) de tamanho *n* “particionando-o” em dois subproblemas, de tal forma que todos os elementos no **subproblema 1 sejam menores que os elementos no subproblema 2**.

2 – Aplicar o mesmo método (passos 1), recursivamente, para a subsequência formada pelos elementos de cada subproblema até restarem subproblemas de tamanho 1.

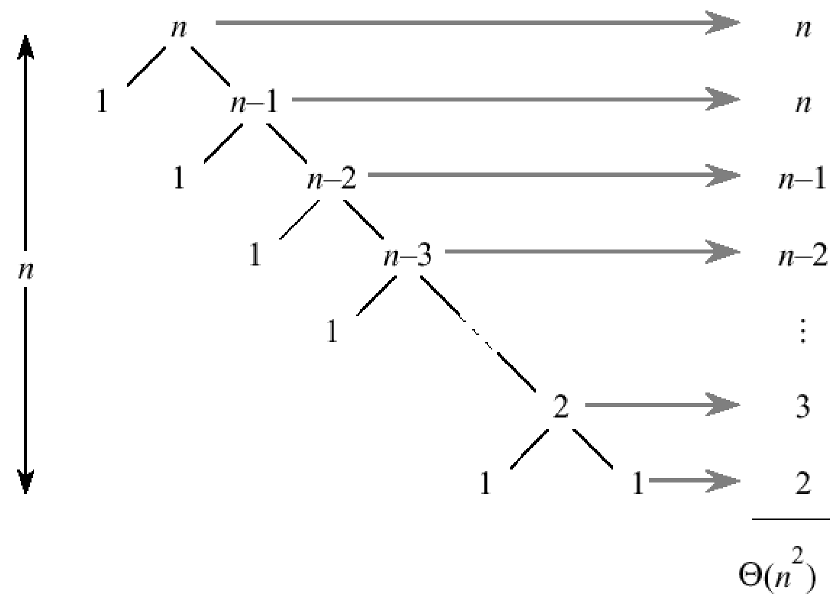
3 – Combinar os elementos dos subproblemas ordenando-os.

Quicksort

MELHOR CASO



PIOR CASO



Maior valor de um vetor

- É possível aplicar Divisão em Conquista para encontrar o maior valor em um vetor?

- Opção 1:

```
int maxVal1(int A[], int n) {
    int max = A[0];
    for (int i = 1; i < n; i++) {
        if( A[i] > max ) max = A[i];
    }
    return max;
}
```

- Melhor alternativa??

Maior valor de um vetor

```
int maxVal2(int A[], int init, int end) {
    if (end - init <= 1)
        return max(A[init], A[end]);
    else {
        int m = (init + end)/2;
        int v1 = maxVal2(A,init,m);
        int v2 = maxVal2(A,m+1,end);
        return max(v1,v2);
    }
}
```

- E agora? Melhorou?

Exponenciação

```
int pow1(int a, int n) {
    int p = 1;
    for (int i = 0; i < n; i++)
        p = p * a;
    return p;
}
```

Exponenciação

```c
int pow1(int a, int n) {
    int p = 1;
    for (int i = 0; i < n; i++)
        p = p * a;
    return p;
}
```

```c
int pow2(int a, int n) {
    if (n == 0)
        return 1;
    if (n % 2 == 0)
        return pow2(a,n/2) * pow2(a,n/2);
    else
        return pow2(a,(n-1)/2) * pow2(a,(n-1)/2) * a;
}
```

Multiplicação de inteiros grandes (bignum)

- O problema consiste em multiplicar dois números inteiros grandes (bignum)

- A multiplicação clássica (a que aprendemos a fazer na escola) requer tempo O(n2). Isso porque fazemos multiplicação dígito a dígito.

2952 32799 03960 41408 47618 60964 35200

X

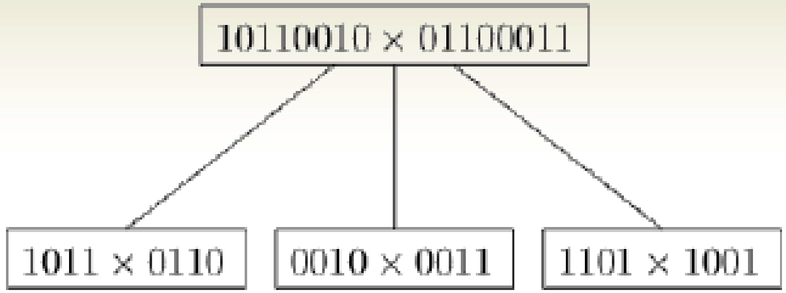
86 83317 61881 18864 95518 19440 12800

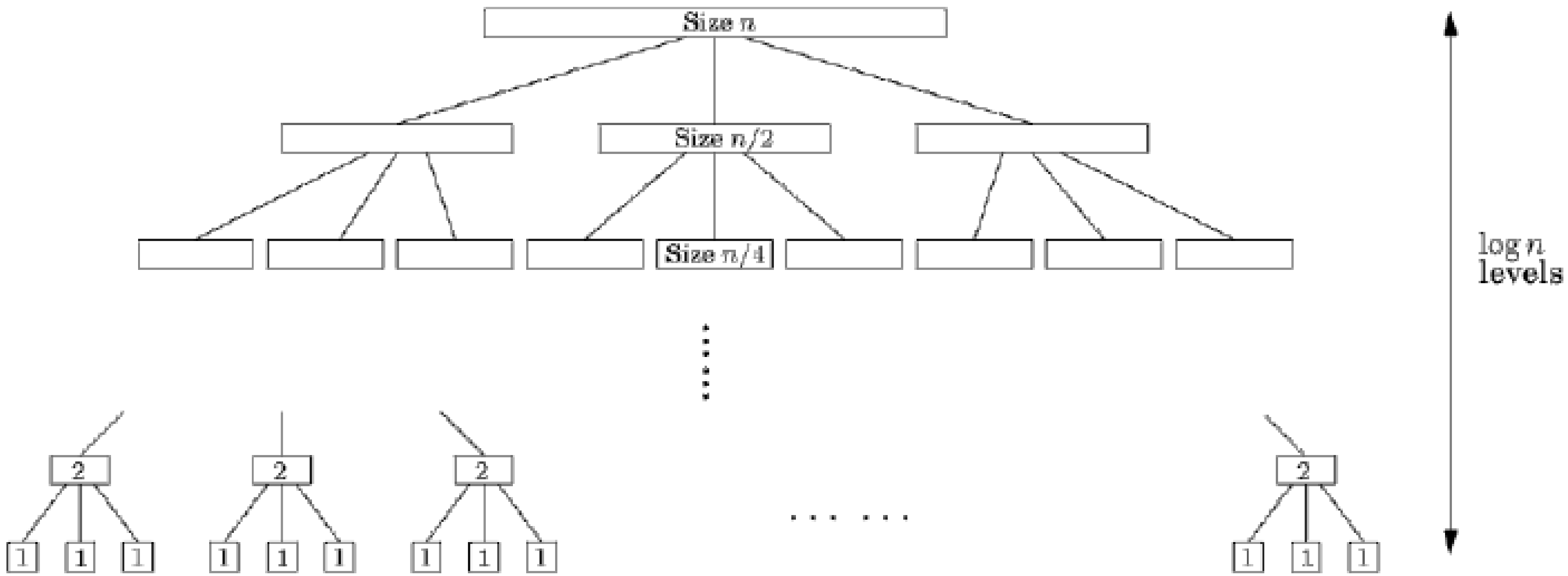
- Há uma solução alternativa por Divisão e Conquista?

Multiplicação de inteiros grandes (bignum)

- **Sim !!!**
- Solução alternativa por Divisão e Conquista
- Para evitar maiores complicações, vamos assumir que o número de digitos em cada número é potência de 2
- A multiplicação de um número A por um número B pode ser efetuada dividindo-se o número original em dois super-dígitos e procedendo a multiplicação

Multiplicação de inteiros grandes (bignum)





Multiplicação de inteiros grandes (bignum)

Resumindo...

- Multiplicando bignums por Divisão e Conquista:

- **Divisão**: Dividir cada número em dois números com a metade da quantidade de dígitos

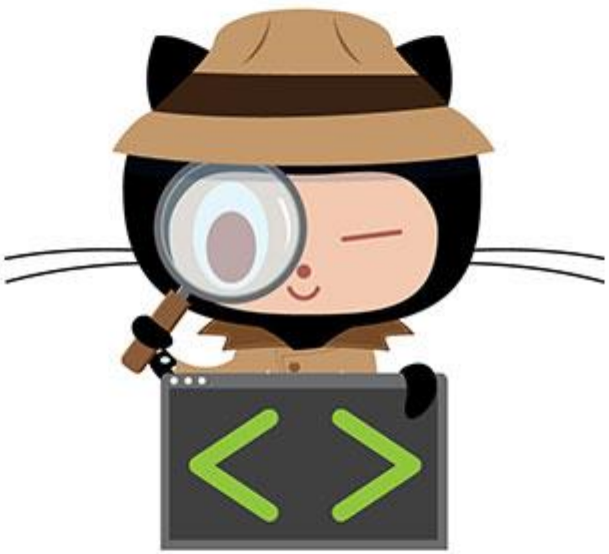
- **Conquista**: Proceder a multiplicação das quatro partes

- **Combinação**: Combinar os resultados através dos respectivos deslocamentos e adições

Divisão e Conquista

- Na aplicação da **técnica de decomposição**, frequentemente há casos em que um mesmo subproblema aparece diversas vezes ao longo do processo.
- A decomposição pura e simples é incapaz de reconhecer este fato.
- Nestes casos, é conveniente utilizar uma variação da decomposição denominada de **PROGRAMAÇÃO DINÂMICA**.

See you



Perguntas?