

Vistula, IT Faculty, 2014

# Algorithms and Complexity

Dmitry A. Zaitsev

<http://daze.ho.ua>

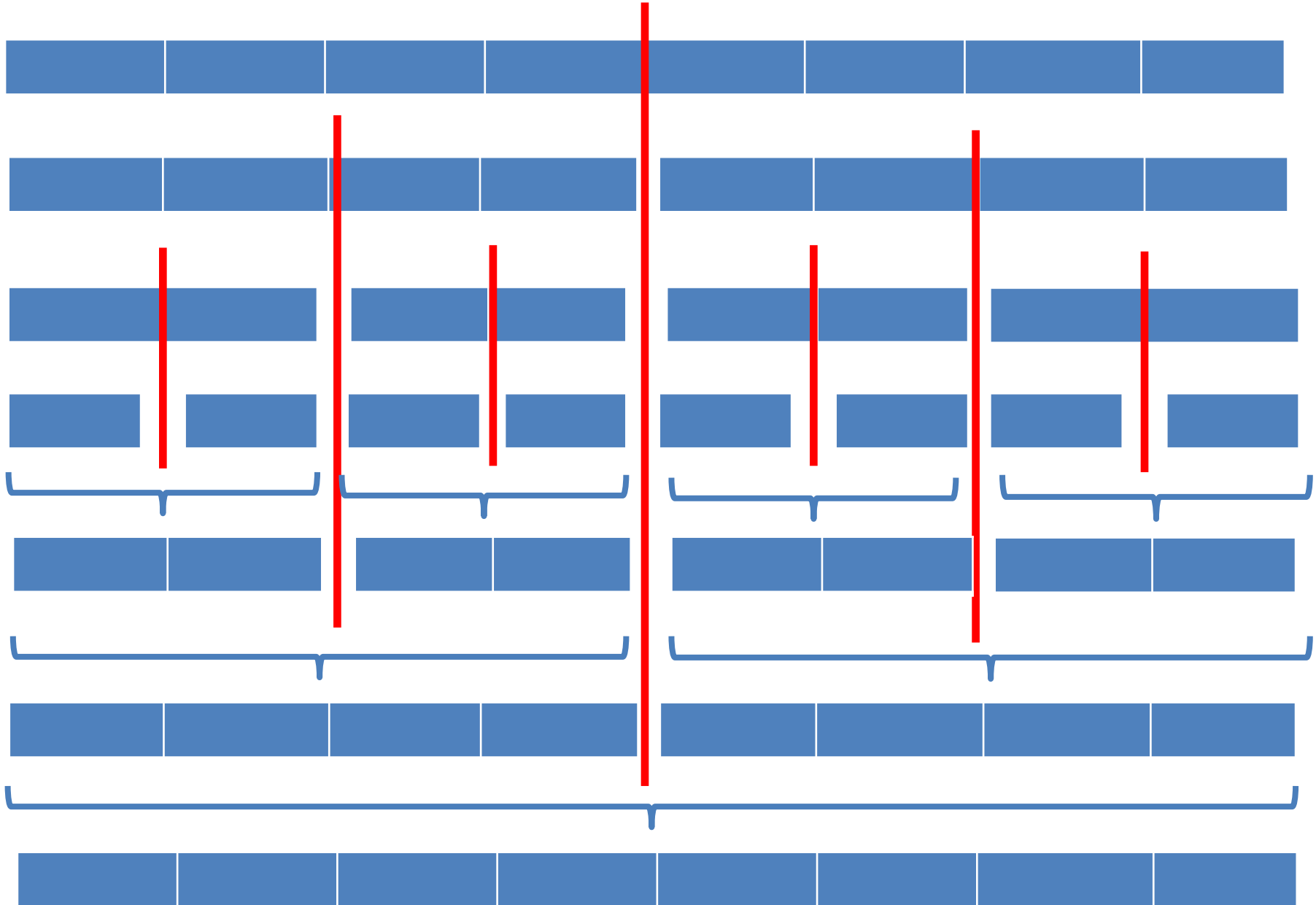
## Lecture 13:

## Divide and conquer approach

## Divide and conquer principles

- Partition a problem into smaller parts
  - Find solutions for the parts
  - Combine the solution for parts into a solution for the whole
- 
- Decomposition
  - Solution
  - Composition

# Megesort



## Minimum and maximum of a set $n=2^m$

```
void maxmin(int *a, int l, int r, int *mx, int *mn)
{
    int mx1, mn1, mx2, mn2;
    if( r-l==1 )
        *mx = max(a[l],a[r]); *mn = min(a[l],a[r]);
    else
    {
        mid=(r-l)/2;
        maxmin(a,l, mid, &mx1, &mn1); maxmin(a,mid+1, r, &mx2, &mn2)
        *mx=max(mx1,mx2); *mn=min(mn1,mn2);
    }
    return;
}
```

## Multiplication of integers

x and y n-bit numbers

$$x = \begin{array}{|c|c|} \hline a & b \\ \hline \end{array}$$

$$y = \begin{array}{|c|c|} \hline c & d \\ \hline \end{array}$$

$$x \cdot y = \left(a \cdot 2^{\frac{n}{2}} + b\right) \cdot \left(c \cdot 2^{\frac{n}{2}} + d\right) =$$
$$a \cdot c \cdot 2^n + (a \cdot d + b \cdot c) \cdot 2^{\frac{n}{2}} + b \cdot d$$

$$u = (a+b) \cdot (c+d);$$

$$v = a \cdot c;$$

$$w = b \cdot d;$$

$$z = v \ll n + (u - v - w) \ll (n/2) + w;$$

$$T(n) = \begin{cases} k, & n = 1 \\ 3 \cdot T\left(\frac{n}{2}\right) + k \cdot n, & k > 1 \end{cases}$$

$$T(n) = 3 \cdot k \cdot n^{\log 3} \approx 3 \cdot k \cdot n^{1.59}$$

## Multiplication of integers – estimation of complexity

$$T(n) = 3 \cdot k \cdot n^{\log 3} - 2 \cdot k \cdot n$$

induction on n for n a power of 2

It holds for n=1

Let it satisfies recurrent expression

for n=m

$$T(n) = \begin{cases} k, n = 1 \\ 3 \cdot T\left(\frac{n}{2}\right) + k \cdot n, k > 1 \end{cases}$$

$$\begin{aligned} T(2 \cdot m) &= 3 \cdot T(m) + 2 \cdot k \cdot m \\ &= 3 \cdot (3 \cdot k \cdot m^{\log 3} - 2 \cdot k \cdot m) + 2 \cdot k \cdot m \\ &= 3 \cdot k \cdot (2 \cdot m)^{\log 3} - 2 \cdot k \cdot (2 \cdot m) \end{aligned}$$

thus

$$T(n) \leq 3 \cdot k \cdot n^{\log 3}$$

## Block multiplication of matrices

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

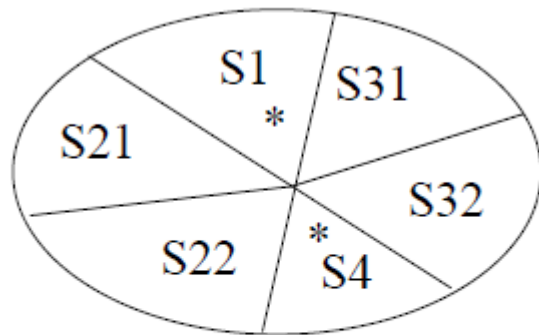
$$T(n) = 8T\left(\frac{n}{2}\right) + O(n^2)$$

Strassen offered a scheme to provide

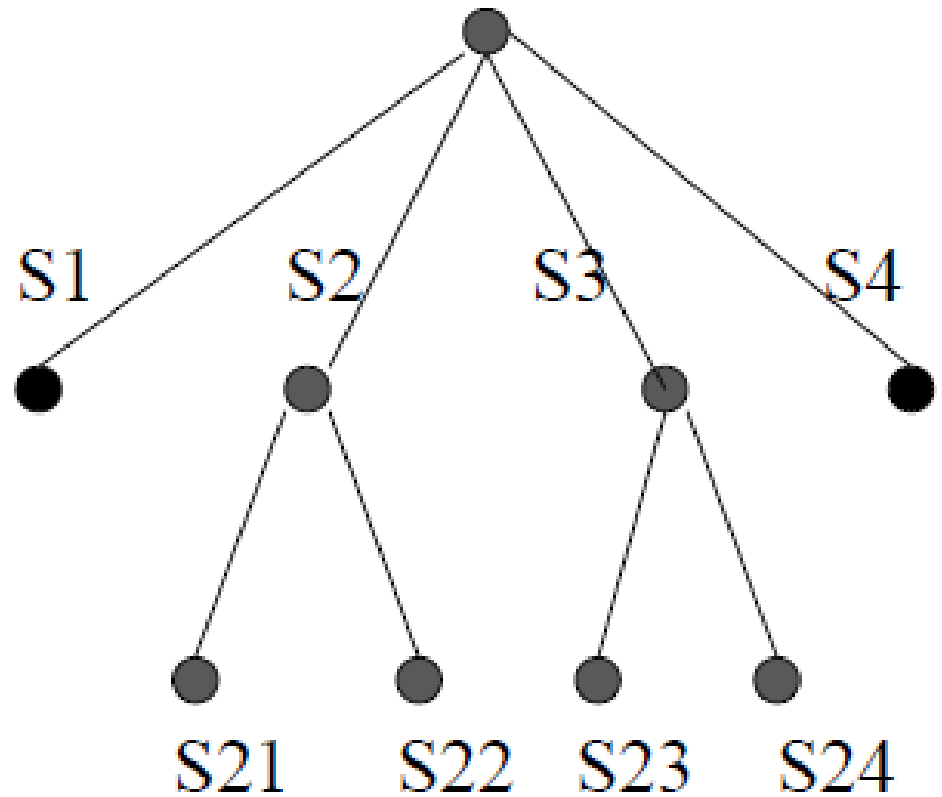
$$T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$$

$$T(n) = O(n^{\log 7}) \approx O(n^{2.81})$$

# Branch & bound



\* = does not contain  
optimal solution





## B&B algorithm and bounding function

1. a *bounding function* providing for a given subspace of the solution space a lower bound for the best solution value obtainable in the subspace,
2. a *strategy for selecting* the live solution subspace to be investigated in the current iteration, and
3. a *branching rule* to be applied if a subspace after investigation cannot be discarded, hereby subdividing the subspace considered into two or more subspaces to be investigated in subsequent iterations.

1.  $g(P_i) \leq f(P_i)$  for all nodes  $P_i$  in the tree

2.  $g(P_i) = f(P_i)$  for all leaves in the tree

3.  $g(P_i) \geq g(P_j)$  if  $P_j$  is the father of  $P_i$

## Eager B&B

Initialize:  $\text{Incumbent} := \infty$ ;  $\text{LB}(P_0) := g(P_0)$ ;  $\text{Live} := \{(P_0, \text{LB}(P_0))\}$

Repeat until  $\text{Live} = \emptyset$

    Select the node  $P$  from  $\text{Live}$  to be processed;  $\text{Live} := \text{Live} \setminus \{P\}$ ;

    Branch on  $P$  generating  $P_1, \dots, P_k$ ;

    For  $1 \leq i \leq k$  do

        Bound  $P_i$  :  $\text{LB}(P_i) := g(P_i)$  ;

        If  $\text{LB}(P_i) = f(X)$  for a feasible solution  $X$   
        and  $f(X) < \text{Incumbent}$  then

$\text{Incumbent} := f(X)$ ;  $\text{Solution} := X$ ;

            go to EndBound;

        If  $\text{LB}(P_i) \geq \text{Incumbent}$  then fathom  $P_i$

        else  $\text{Live} := \text{Live} \cup \{(P_i, \text{LB}(P_i))\}$

    EndBound;

$\text{OptimalSolution} := \text{Solution}$ ;  $\text{OptimumValue} := \text{Incumbent}$

## Lazy B&B

Initialize:  $\text{Incumbent} := -\infty$ ;  $\text{Live} := \{(P_0, -\infty)\}$

Repeat until  $\text{Live} = \emptyset$

    Select the node  $P$  from  $\text{Live}$  to be processed;  $\text{Live} := \text{Live} \setminus \{P\}$ ;

    Bound  $P$  :  $\text{LB}(P) := g(P)$

        If  $\text{LB}(P) = f(X)$  for a feasible solution  $X$

            and  $f(X) < \text{Incumbent}$  then

$\text{Incumbent} := f(X)$ ;  $\text{Solution} := X$ ;

                go to EndBound;

        If  $\text{LB}(P) \geq \text{Incumbent}$  then fathom  $P$

        else Branch on  $P$  generating  $P_1, \dots, P_k$ ;

            For  $1 \leq i \leq k$  do

$\text{Live} := \text{Live} \cup \{(P_i, \text{LB}(P))\}$ ;

    EndBound;

OptimalSolution := Solution; OptimumValue := Incumbent;