Tutorial 1

Recurrence: Divide and Conquer Strategy and Analysis

Wenzhong Li
lwz@dislab.nju.edu.cn

1. Math Background

- P21-27
- Harmonic Series: $$\sum_{i=1}^{n}\frac{1}{i}\approx \ln(n)+\gamma$$
- Arithmetic-Geometric Series: $$\sum_{i=1}^{k} i2^{i} = (k-1)2^{k+1}+2$$
- Stirling's Formula: $$n! \approx \sqrt{2\pi n}(\frac{n}{e})^{n}$$

- Example: Asymptotic order:

$$lgn,\ n^{\epsilon}(\epsilon > 0),\ c^{n}(c > 0),\ \text{n!}$$

Proof $a^n(a > 0) = o(n!)$

$$\lim_{n\to\inf}\frac{a^n}{n!} = \lim_{n\to\inf}\frac{a^n}{\sqrt{2\pi n}(\frac{n}{e})^n} = 0$$

(Here we use the Stirling Formular $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$)

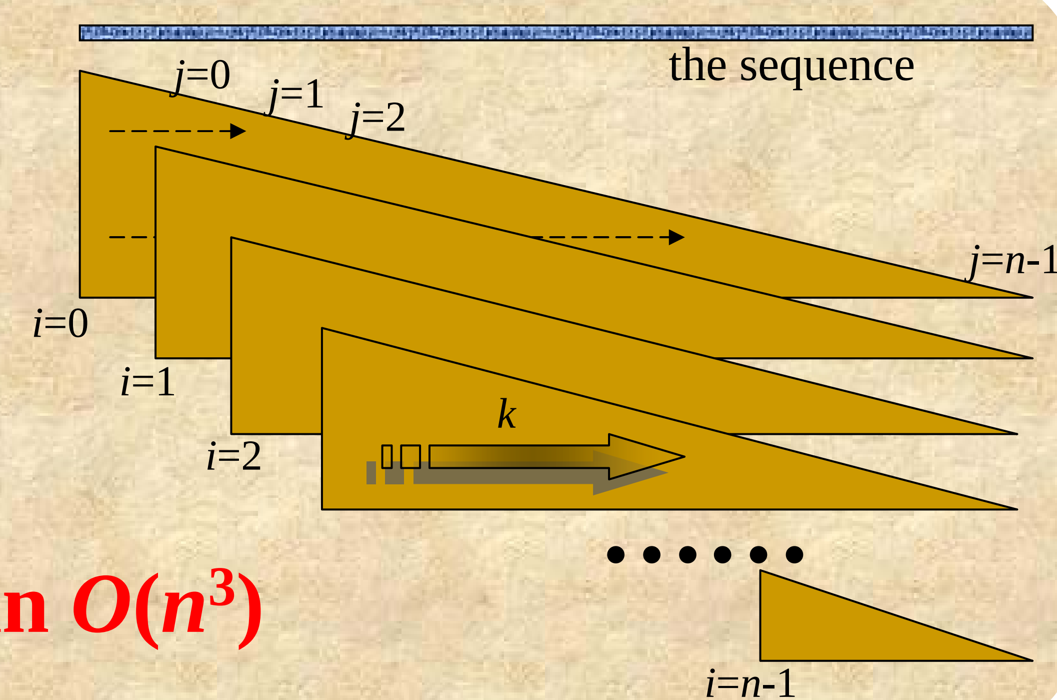
- P52 Theorem 1.13
- Polynomial Series: $$\sum_{i=1}^{n} i^d \sim \theta(n^{d+1})$$
- Geometric Series: $$\sum_{i=a}^{b} r^i \sim \theta(\text{largest})$$
- Logarithmic Series: $$\sum_{i=1}^{n} \log(i) \sim \theta(n \log(n))$$
- Polynomial- logarithmic Series: $$\sum_{i=1}^{n} i^d \log(i) \sim \theta(n^{d+1} \log(n))$$

Example: Maximum Subsequence Sum

- The problem: Given a sequence *S* of integer, find the **largest sum** of a consecutive subsequence of *S*. (0, if all negative items)
  - An example: -2, 11, -4, 13, -5, -2; the result 20: (11, -4, 13)

A brute-force algorithm:

```
MaxSum = 0;
 for (i = 0; i < N; i++)
  for (j = i; j < N; j++)
  {
   ThisSum = 0;
   for (k = i; k <= j; k++)
   ThisSum += A[k];
   if (ThisSum > MaxSum)
    MaxSum = ThisSum;
  }
 return MaxSum;
```



in $O(n^3)$

More Precise Complexity

The total cost is : $\sum_{i=0}^{n-1} \sum_{j=i}^{n-1} \sum_{k=i}^{j} 1$

$$\sum_{k=i}^{j} 1 = j - i + 1$$

$$\sum_{j=i}^{n-1} (j - i + 1) = 1 + 2 + \ldots + (n - i) = \frac{(n - i + 1)(n - i)}{2}$$

$$\sum_{i=0}^{n-1} \frac{(n - i + 1)(n - i)}{2} = \sum_{i=1}^{n} \frac{(n - i + 2)(n - i + 1)}{2}$$

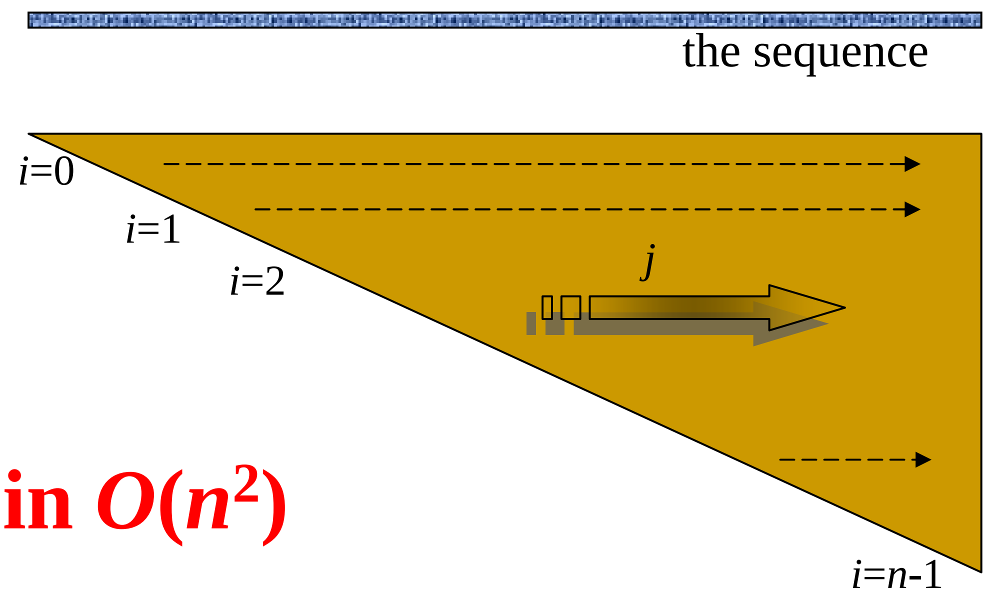
$$= \frac{1}{2} \sum_{i=1}^{n} i^2 - (n + \frac{3}{2}) \sum_{i=1}^{n} i + \frac{1}{2}(n^2 + 3n + 2) \sum_{i=1}^{n} 1$$

$$= \frac{n^3 + 3n^2 + 2n}{6}$$

Decreasing the number of loops

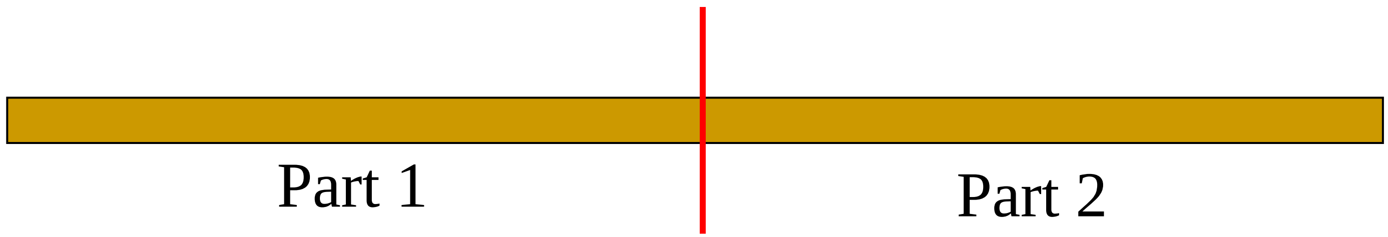
An improved algorithm

```
MaxSum = 0;
 for (i = 0; i < N; i++)
 {
  ThisSum = 0;
  for (j = i; j < N; j++)
  {
   ThisSum += A[j];
   if (ThisSum > MaxSum)
    MaxSum = ThisSum;
  }
 }
 return MaxSum;
```

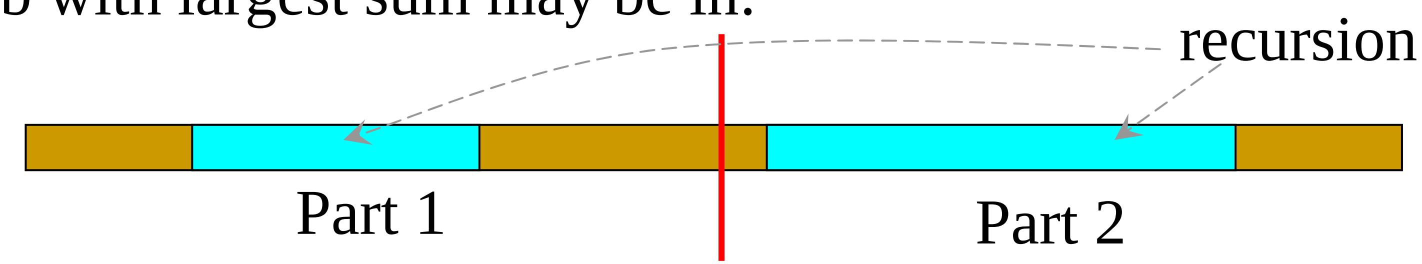


in $O(n^2)$

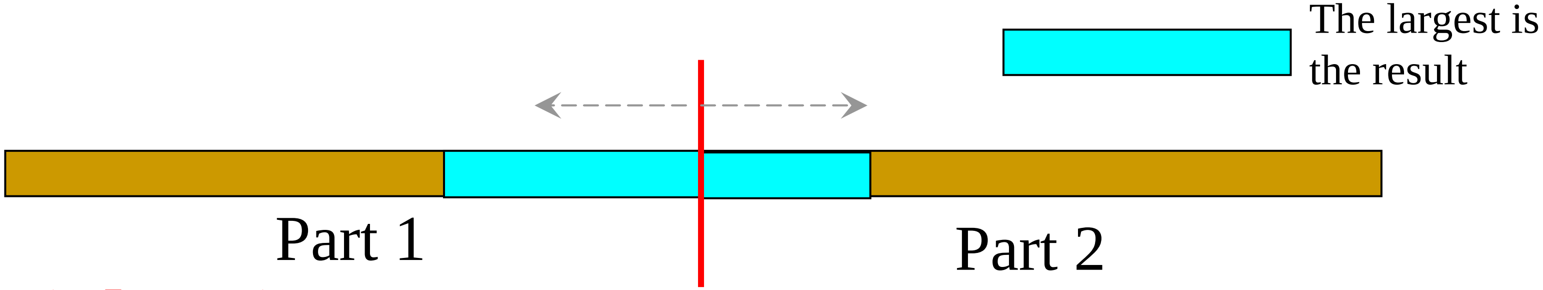
Power of Divide-and-Conquer



the sub with largest sum may be in:



or:



in $O(n\log n)$

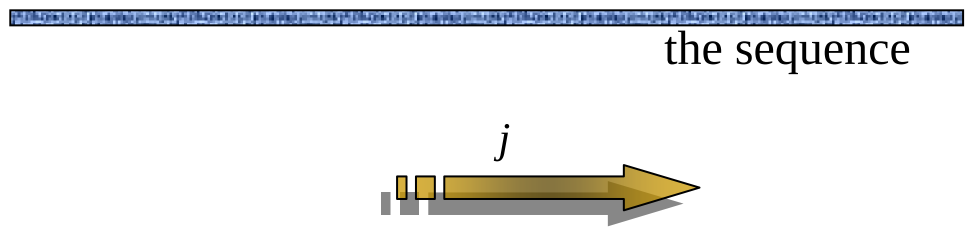
Divide-and-Conquer: the Procedure

```
Center = (Left + Right) / 2;
 MaxLeftSum = MaxSubSum(A, Left, Center);  MaxRightSum = MaxSubSum(A, Center + 1, Right);

 MaxLeftBorderSum = 0; LeftBorderSum = 0;
 for (i = Center; i >= Left; i--)
 {
  LeftBorderSum += A[i];
  if (LeftBorderSum > MaxLeftBorderSum)    MaxLeftBorderSum = LeftBorderSum;
 }

 MaxRightBorderSum = 0; RightBorderSum = 0;
 for (i = Center + 1; i <= Right; i++)
 {
  RightBorderSum += A[i];
  if (RightBorderSum > MaxRightBorderSum)  MaxRightBorderSum = RightBorderSum;
 }
 return Max3(MaxLeftSum, MaxRightSum,
      MaxLeftBorderSum + MaxRightBorderSum);
```

Note: this is the core part of the procedure, with base case and wrap omitted.

A Linear Algorithm

```
ThisSum = MaxSum = 0;
  for (j = 0; j < N; j++)
  {
    ThisSum += A[j];
    if (ThisSum > MaxSum)
      MaxSum = ThisSum;
    else if (ThisSum < 0)
      ThisSum = 0;
  }
  return MaxSum;
```



This is an example of
"online algorithm"

Negative item or subsequence cannot be a prefix of the subsequence we want.

in $O(n)$

Comparison

- Brute force
  - $O(n^3)$
- Improved brute force
  - $O(n^2)$
- Divide and Conquer
  - $O(n \log n)$
- Double pointer
  - $O(n)$

2. Analysis to Recurrence

- Recursive Algorithm and Recurrence equation
- How to analyze?
  - Guess and Proving
  - Recursion tree

Guess and Proving

- Example: $T(n)=2T(\lfloor n/2 \rfloor) +n$
- Guess
  - $T(n) \in O(n)$?
    - $T(n) \le cn$, to be proved for c large enough
  - $T(n) \in O(n^2)$?
    - $T(n) \le cn^2$, to be proved for c large enough
  - $T(n) \in O(n\log n)$?
    - $T(n) \le cn\log n$, to be proved for c large enough

Try to prove $T(n) \le cn$:

$$T(n)=2T(\lfloor n/2 \rfloor)+n \le 2c(\lfloor n/2 \rfloor)+n$$
$$\le 2c(n/2)+n = (c+1)\mathrm{n},$$ **Fail!**

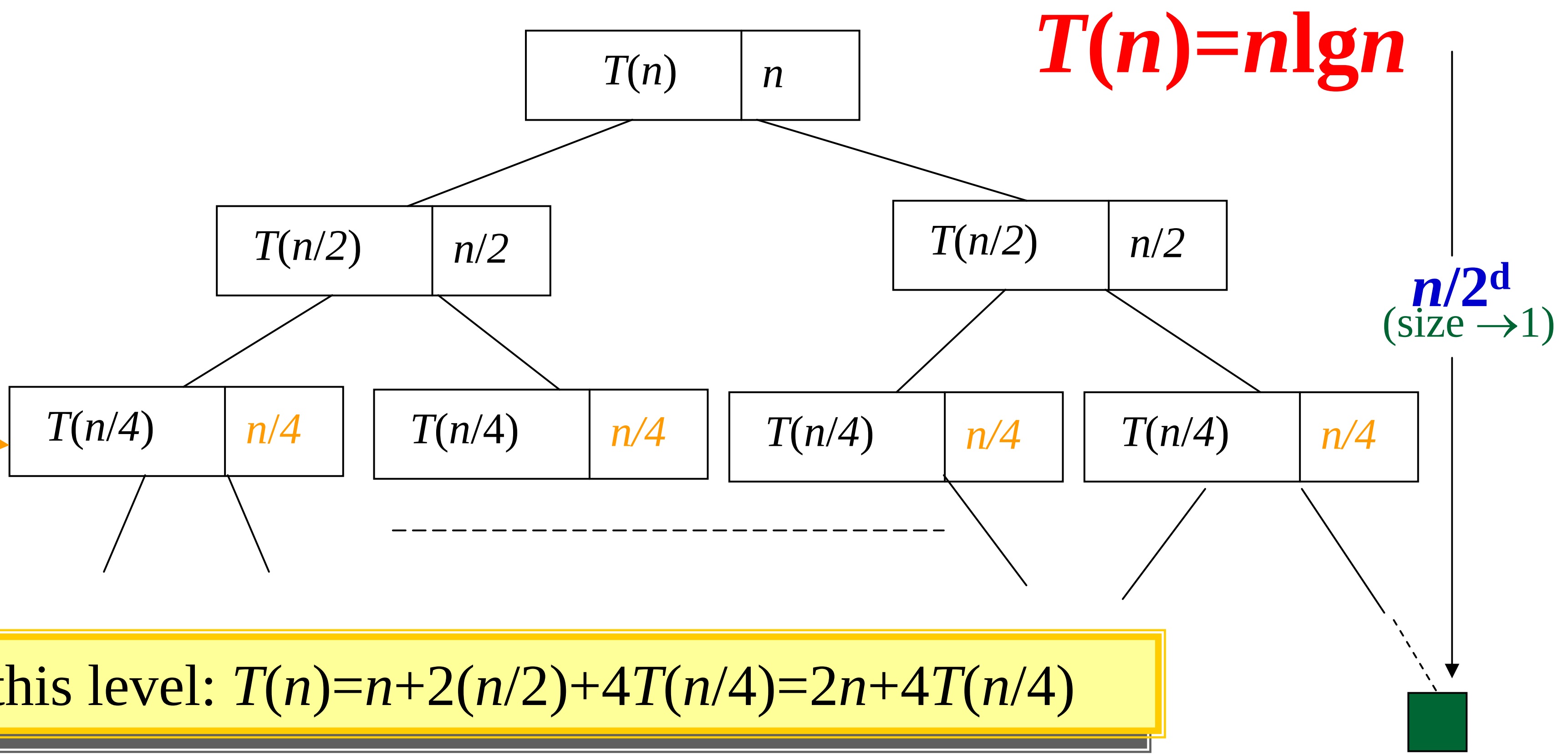
However:

$$T(n) = 2T(\lfloor n/2 \rfloor)+n \ge 2c\lfloor n/2 \rfloor+n$$
$$\ge 2c[(n\text{-}1)/2]+n = cn+(n\text{-}c) \ge cn$$

$$\begin{aligned}
T(n) &= 2T(\lfloor n/2 \rfloor)+n \\
&\le 2(c\lfloor n/2 \rfloor \log (\lfloor n/2 \rfloor))+n \\
&\le cn\log (n/2)+n \\
&= cn \log n - cn \log 2 +n \\
&= cn \log n - cn + n \\
&\le cn \log n \quad \text{for } c \ge 1
\end{aligned}$$

Recursion Tree

Work copy: T(k)=T(k/2)+T(k/2)+k



At this level: $T(n)=n+2(n/2)+4T(n/4)=2n+4T(n/4)$

2.1 $T(n)=bT(n/c)+f(n)$

The positive ε is critical, resulting gaps between cases as well

- Master Theorem
- Loosening the restrictions on $f(n)$
  - Case 1: $f(n)\in O(n^{E-\varepsilon})$, $(\varepsilon>0)$, then:

    $$T(n)\in\Theta(n^{E})$$

  - Case 2: $f(n)\in\Theta(n^{E})$, as all node depth contribute about equally:

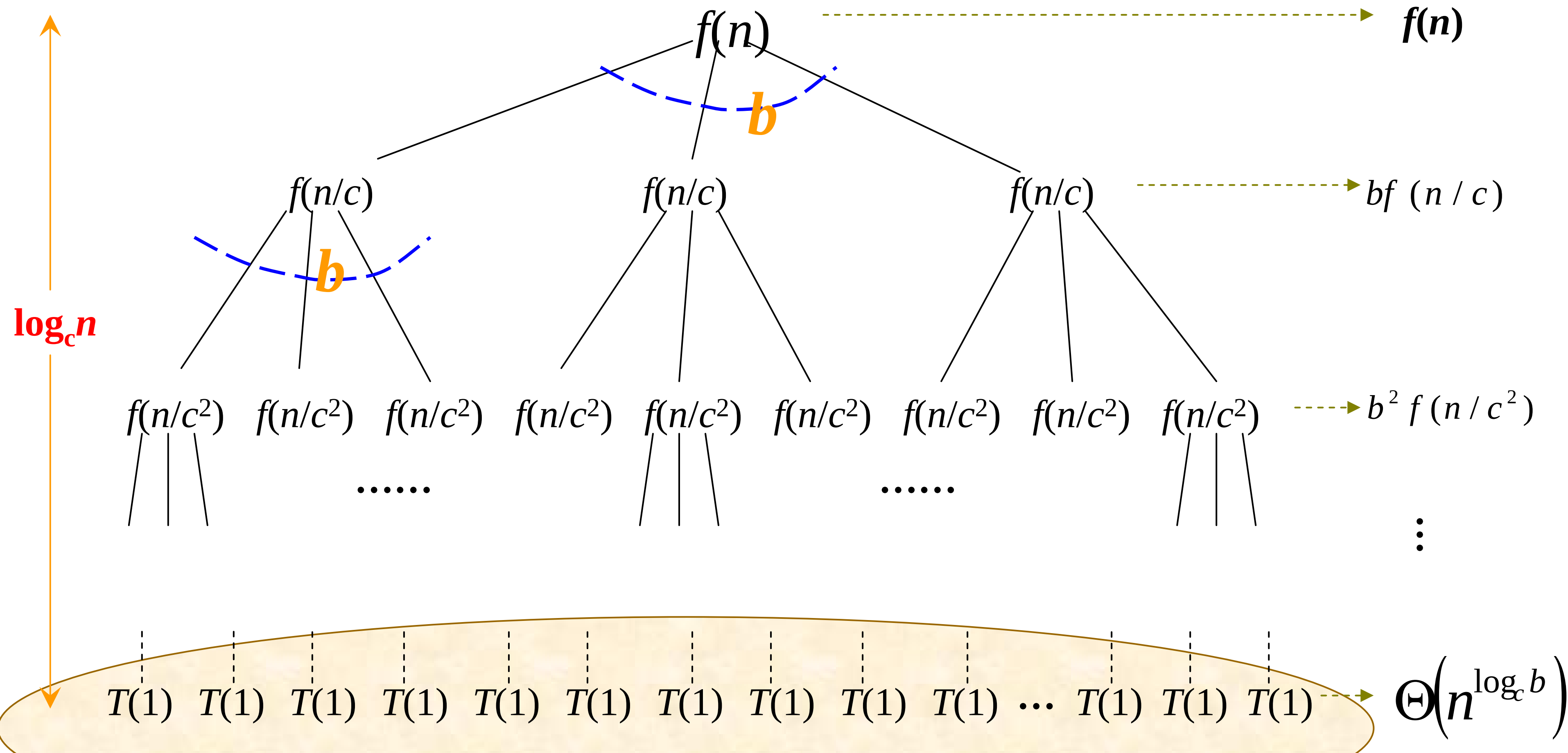
    $$T(n)\in\Theta(f(n)\log(n))$$

  - case 3: $f(n)\in\Omega(n^{E+\varepsilon})$, $(\varepsilon>0)$, and $f(n)\in O(n^{E+\delta})$, $(\delta\geq\varepsilon)$, then:

    $$T(n)\in\Theta(f(n))$$

This is regular condition, we can use $bf(n/c) \leq rf(n)$ $(r<1)$ instead

Recursion Tree for $T(n)=bT(n/c)+f(n)$



Note: $b^{\log_c n} = n^{\log_c b}$

Total ?

- Examples
- (1) $T(n)=4T(n/2)+n$

    case 1, $T(n) \in \Theta(n^2)$
- (2) $T(n)=4T(n/2)+n^2$

    case 2, $T(n) \in \Theta(n^2 \lg n)$
- (3) $T(n)=4T(n/2)+n^3$

    case 3, $T(n) \in \Theta(n^3)$
- (4) $T(n)=4T(n/2)+n^2/\lg n$

    none of the three cases, gap

2.2 T(n)=bT(n-c)+f(n) (P139)

If b=1,

$T(n) \approx \frac{1}{c}\int_0^n f(x)dx$

(1) if f(n) is polynomial n^α, then $T(n) \in \Theta(n^{\alpha+1})$

(2) if f(n)=log(n) then $T(n) \in \Theta(nlgn)$

2.3 $T(n)=r_1T(n-1)+r_2T(n-2)$

- Characteristic Equation
- If the characteristic equation $x^2 - r_1x - r_2 = 0$ of the recurrence relation $a_n = r_1 a_{n-1} + r_2 a_{n-2}$ has two distinct roots s_1 and s_2, then

$$a_n = us_1^n + vs_2^n$$

where u and v depend on the initial conditions, is the explicit formula for the sequence.

$$f_1 = us_1 + vs_2 \quad and \quad f_2 = us_1^2 + vs_2^2$$

3. Divide and Conquer

Basic Strategy

- Divide: devide the problem into smaller instances of the same problem
- Conquer: solve the smaller problem recursively
- Combine: combine the solutions to obtain the solution for the original input

Example: Matrix Multiplication

$$\left.\begin{array}{ll}\textbf{Input:} & A = [a_{ij}], B = [b_{ij}]. \\ \textbf{Output:} & C = [c_{ij}] = A \cdot B.\end{array}\right\} \quad i, j = 1, 2, \ldots, n.$$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Standard Algorithm – by definition

- Run time = Θ (n^3)

for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
        do $c_{ij} \leftarrow 0$
            for $k \leftarrow 1$ **to** n
                do $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

Divide-and-conquer Algorithm

- Idea: n*n matrix = 2*2 of (n/2) * (n/2) sub-matrices:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C_{11} = A_{11} B_{11} + A_{12} B_{21}$$

$$C_{12} = A_{11} B_{12} + A_{12} B_{22}$$

$$C_{21} = A_{21} B_{11} + A_{22} B_{21}$$

$$C_{22} = A_{21} B_{12} + A_{22} B_{22}$$

- Analysis:
- 8 muls of (n/2)*(n/2) submatrices
- 4 adds of (n/2)*(n/2) submatrices
- $T(n)=8T(n/2)+n^2$
- Still, $T(n)=O(n^3)$.
- No improvement!

Strassen Algorithm

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$M_1 = A_{11}(B_{12} - B_{22})$$
$$M_2 = (A_{11} + A_{12})B_{22}$$
$$M_3 = (A_{21} + A_{22})B_{11}$$
$$M_4 = A_{22}(B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$M_6 = (A_{12} - A_{22})(B_{21} + B_{22})$$
$$M_7 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$C_{11} = M_5 + M_4 - M_2 + M_6$$
$$C_{12} = M_1 + M_2$$
$$C_{21} = M_3 + M_4$$
$$C_{22} = M_5 + M_1 - M_3 - M_7$$

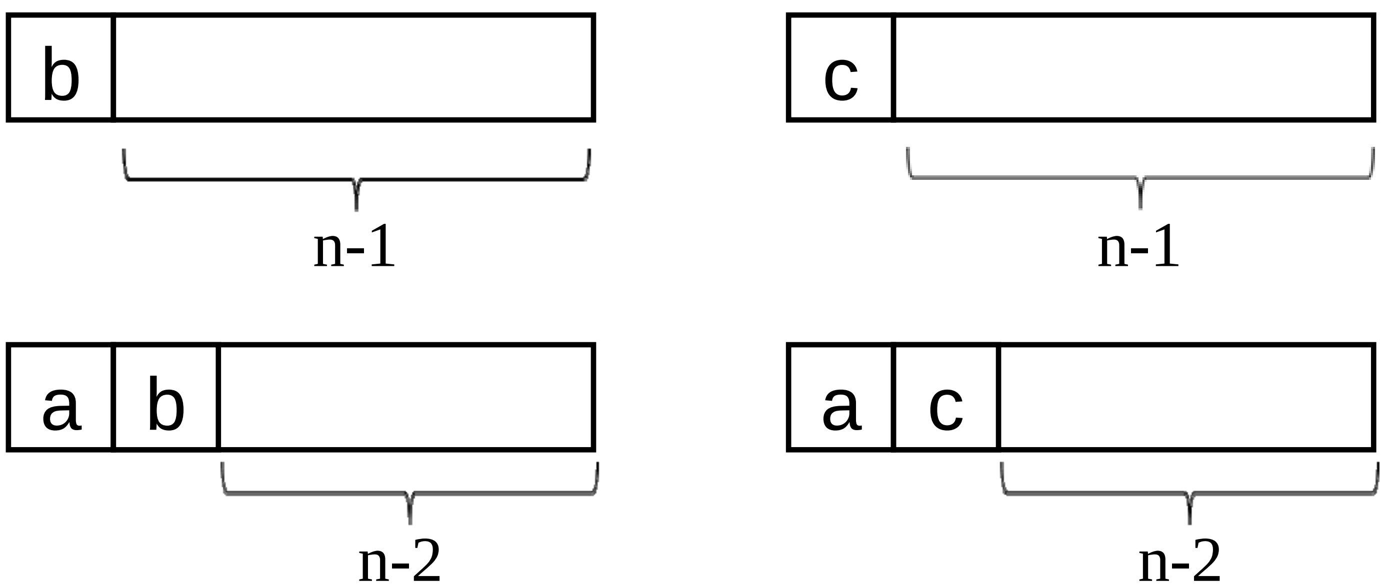
- Analysis:
- 7 muls of (n/2)*(n/2) submatrices
- 18 adds of (n/2)*(n/2) submatrices
- $T(n)=7T(n/2)+n^2$
- $T(n)=O(n^{\log(7)})=O(n^{2.81})$.
- Great improvement!

Example: Number of Valid Strings

- String to be transmitted on the channel
  - Length n
  - Consisting of symbols ‘a’, ‘b’, ‘c’
  - If “aa” exists, cannot be transmitted
  - E.g. strings of length 2: ‘ab’, ‘ac’, ‘ba’, ‘bb’, ‘bc’, ‘ca’, ‘cc’, ‘cb’
- Number of valid strings ?

Divide and conquer

- $f(n)=2f(n-1)+2f(n-2), n>2$
  - $f(1)=3, f(2)=8$



Analysis of the D&C solution

- Characteristic equation

$$x^2 - 2x - 2 = 0$$

- Solution

$$f(n) = \frac{2+\sqrt{3}}{2\sqrt{3}}(1+\sqrt{3})^n + \frac{-2+\sqrt{3}}{2\sqrt{3}}(1-\sqrt{3})^n$$