Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

Techniques for the design of Algorithms

The classical techniques are as follows:

1. Divide and Conquer
2. Dynamic Programming
3. Greedy Algorithms
4. Backtracking Algorithms
5. Branch and Bound Algorithms

The first example: Fibonacci Sequence

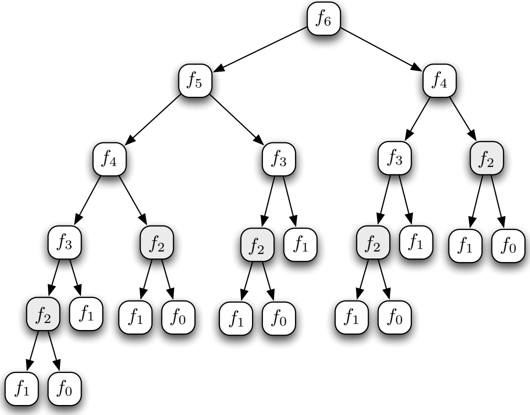
Review

$$f_n = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ f_{n-1} + f_{n-2} & \text{if } n \geq 2. \end{cases}$$

The first example: Fibonacci Sequence

Review

$$f_n = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ f_{n-1} + f_{n-2} & \text{if } n \geq 2. \end{cases}$$

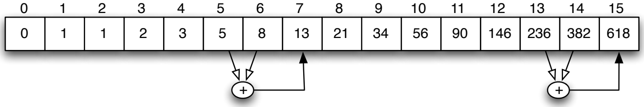


The first example: Fibonacci Sequence

Since there are a lot of common subproblems, therefore by using Divide and Conquer approach we have to solve all of them and this cause the exponential time complexity. Instead, we can solve each subproblem exactly once and save its answer for the future usage.

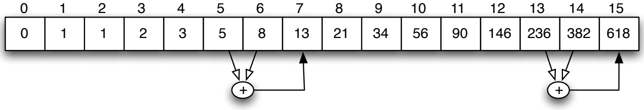
The first example: Fibonacci Sequence

Since there are a lot of common subproblems, therefore by using Divide and Conquer approach we have to solve all of them and this cause the exponential time complexity. Instead, we can solve each subproblem exactly once and save its answer for the future usage.



The first example: Fibonacci Sequence

Since there are a lot of common subproblems, therefore by using Divide and Conquer approach we have to solve all of them and this cause the exponential time complexity. Instead, we can solve each subproblem exactly once and save its answer for the future usage.



$Fib(n)\{$

$\quad A \leftarrow Array[0 \cdots n];$

$\quad A[0] \leftarrow 0;$

$\quad A[1] \leftarrow 1;$

$\quad for\ i \leftarrow 2\ to\ n\ do\{$

$\quad\quad A[i] \leftarrow A[i-1] + A[i-2];$

$\quad \}$

$\quad return(A[n]);$

$\}$

The first example: Fibonacci Sequence

The space complexity of the previous algorithm is $O(n)$. Can we reduce it?

The first example: Fibonacci Sequence

The space complexity of the previous algorithm is $O(n)$. Can we reduce it?

$Fib(n)\{$

$\quad$ $if\ n \leq 1\ then\ return(n);$

$\quad$ $s_0 \leftarrow 0;$

$\quad$ $s_1 \leftarrow 1;$

$\quad$ $for\ i \leftarrow 2\ to\ n\ do\{$

$\quad\quad$ $s_2 \leftarrow s_1 + s_0;$

$\quad\quad$ $s_0 \leftarrow s_1;$

$\quad\quad$ $s_1 \leftarrow s_2;$

$\quad$ $\}$

$\quad$ $return(s_2);$

$\}$

The second example: Choosing k objects among n objects

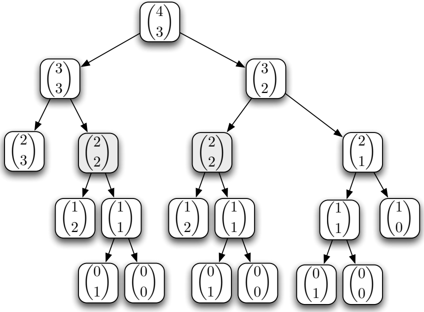
Review

$$\binom{n}{k} = \begin{cases} 0 & \text{if } n < k, \\ 1 & \text{if } n = 0 \text{ or } k = 0, \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{otherwise.} \end{cases}$$

The second example: Choosing k objects among n objects

Review

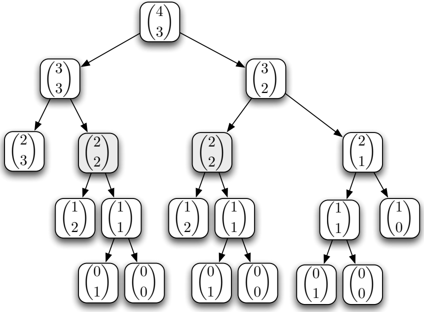
$$\binom{n}{k} = \begin{cases} 0 & \text{if } n < k, \\ 1 & \text{if } n = 0 \text{ or } k = 0, \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{otherwise.} \end{cases}$$



The second example: Choosing *k* objects among *n* objects

Review

$$\binom{n}{k} = \begin{cases} 0 & \text{if } n < k, \\ 1 & \text{if } n = 0 \text{ or } k = 0, \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{otherwise.} \end{cases}$$



k \ n	0	1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	1	1	1
1	0	1	2	3	4	5	6	7	8
2	0	0	1	3	6	10	15	21	28
3	0	0	0	1	4	10	20	35	56
4	0	0	0	0	1	5	15	35	70

The second example: Choosing k objects among n objects

$Choose(k,\ n)\{$
$\quad A \leftarrow Array[0 \cdots k,\ 0 \cdots n];$
$\quad fori \leftarrow 0\ to\ n\ do\{$
$\quad\quad A[0,\ i] \leftarrow 1;$
$\quad\}$
$\quad fori \leftarrow 1\ to\ k\ do\{$
$\quad\quad A[i,\ i] \leftarrow 1;$
$\quad\}$
$\quad fori \leftarrow 1\ to\ k\ do\{$
$\quad\quad forj \leftarrow i\ to\ n\ do\{$
$\quad\quad\quad A[i,\ j] \leftarrow A[i,\ j-1] + A[i-1,\ j-1];$
$\quad\quad\}$
$\quad\}$
$\quad return(A[k,\ n]);$
$\}$

The second example: Choosing k objects among n objects

Review

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

The second example: Choosing k objects among n objects

Review

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$Choose(k,\ n)\{$

$\quad S_n \leftarrow 1;$

$\quad fori \leftarrow 1\ to\ n\ do\{$

$\quad\quad S_n \leftarrow S_n \times i;$

$\quad\quad if\ (i = k)\ then\ S_k \leftarrow S_n;$

$\quad\quad if\ (i = n-k)\ then\ S_{n-k} \leftarrow S_n;$

$\quad\}$

$\quad return(S_n/(S_k \times S_{n-k}));$

$\}$

Matrix Chain Multiplication

Definition

Suppose that we want to calculate the following matrix multiplication:

$$M = M_1 \times M_2 \times M_3 \times \cdots \times M_n$$

, where M_i has d_{i-1} rows and d_i columns. The goal of this problem is determining an order for these matrix multiplication in such a way that the overall number of multiplications becomes minimum.

Matrix Chain Multiplication

Definition

Suppose that we want to calculate the following matrix multiplication:

$$M = M_1 \times M_2 \times M_3 \times \cdots \times M_n$$

, where M_i has d_{i-1} rows and d_i columns. The goal of this problem is determining an order for these matrix multiplication in such a way that the overall number of multiplications becomes minimum.

- How many different order exist for multiplying n matrices?
- How we can determine the best one?

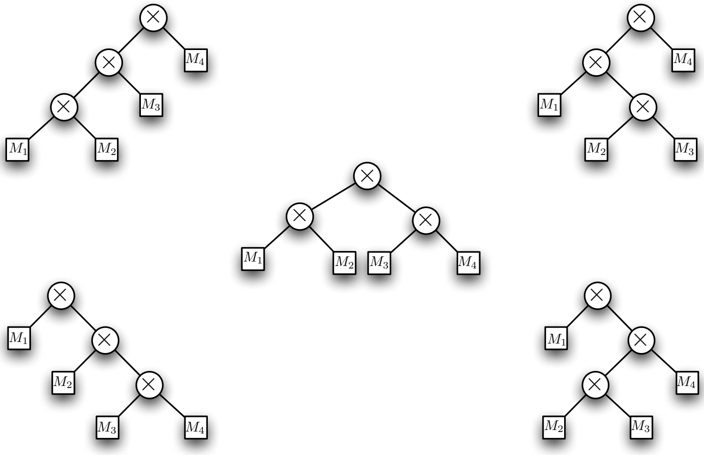
Matrix Chain Multiplication

Example

For $M = M_1 \times M_2 \times M_3 \times M_4$ where $d_0 = 10$, $d_1 = 20$, $d_2 = 50$, $d_3 = 1$, and $d_4 = 100$, the following orders are possible:

$$\begin{aligned}
((M_1 \times M_2) \times M_3) \times M_4 &\implies cost = 11500 \\
(M_1 \times (M_2 \times M_3)) \times M_4 &\implies cost = 2200 \\
(M_1 \times M_2) \times (M_3 \times M_4) &\implies cost = 15000 \\
M_1 \times ((M_2 \times M_3) \times M_4) &\implies cost = 23000 \\
M_1 \times (M_2 \times (M_3 \times M_4)) &\implies cost = 125000
\end{aligned}$$

Matrix Chain Multiplication



Catalan number

Definition

The number of different ordering is called the Catalan number and it can be computed as follows:

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

Catalan number

Definition

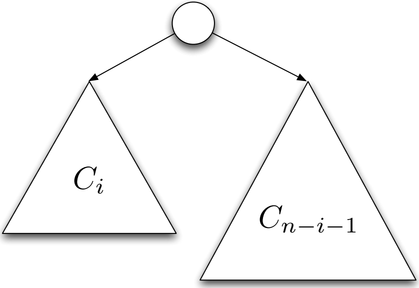
The number of different ordering is called the Catalan number and it can be computed as follows:

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

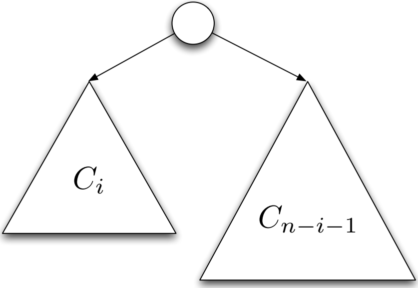
n	0	1	2	3	4	5	6
C_n	1	1	2	5	14	42	128

C_n is the number of binary trees with exactly n internal nodes.

Analyzing Catalan number



Analyzing Catalan number



$$C_n = C_0 C_{n-1} + C_1 C_{n-2} + C_2 C_{n-3} + \cdots + C_{n-1} C_0$$

In other words:

$$C_n = \begin{cases} 1 & \text{if } n = 0 \\ \sum_{i=0}^{n-1} C_i C_{n-i-1} & \text{if } x \geq 1. \end{cases}$$

Analyzing Catalan number

Suppose that:

$$C(x) = \sum_{n=0}^{\infty} C_n x^n$$

Analyzing Catalan number

Suppose that:

$$C(x) = \sum_{n=0}^{\infty} C_n x^n$$

Now we have:

$$\begin{aligned}
\left(\sum_{n=0}^{\infty} C_n x^n\right)^2 &= \sum_{n=0}^{\infty} C_{n+1} x^n \\
&\Downarrow \\
x\left(\sum_{n=0}^{\infty} C_n x^n\right)^2 &= \sum_{n=0}^{\infty} C_{n+1} x^{n+1} \\
&\Downarrow \\
x\left(C(x)\right)^2 &= C(x) - 1 \\
&\Downarrow \\
C(x) &= \frac{1}{2x}\left(1 - \sqrt{1-4x}\right)
\end{aligned}$$

Analyzing Catalan number

We know that:

$$\sqrt{1+x} = 1 - 2\sum_{n=1}^{\infty} \binom{2n-2}{n-1} \left(\frac{-1}{4}\right)^n \frac{x^n}{n}$$

Analyzing Catalan number

We know that:

$$\sqrt{1+x} = 1 - 2\sum_{n=1}^{\infty} \binom{2n-2}{n-1} \left(\frac{-1}{4}\right)^n \frac{x^n}{n}$$

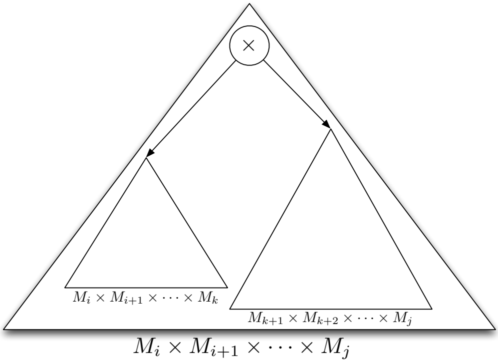
Now we have:

$$\begin{aligned}
C(x) &= \frac{1}{2x}\left(1-\sqrt{1-4x}\right) \\
&= \frac{1}{x}\sum_{n=1}^{\infty} \binom{2n-2}{n-1} \left(\frac{-1}{4}\right)^n \frac{(-4x)^n}{n} \\
&= \sum_{n=1}^{\infty} \binom{2(n-1)}{n-1} \frac{x^{n-1}}{n} \\
&= \sum_{n=0}^{\infty} \binom{2n}{n} \frac{x^n}{n+1}
\end{aligned}$$

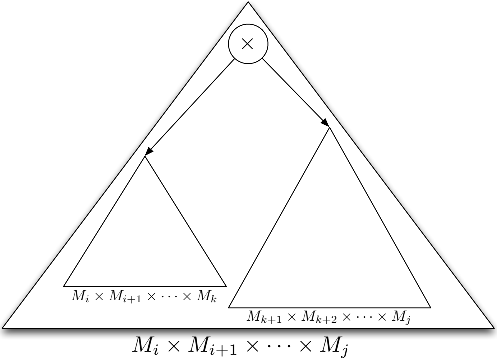
where implies:

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

Dynamic Programming for Matrix Chain Multiplication



Dynamic Programming for Matrix Chain Multiplication



Suppose that $m_{i,j}$ denotes the minimum cost for multiplying $M_i \times M_{i+1} \times \cdots \times M_j$. We can express it as follows:

$$m_{i,j} = \begin{cases} 0 & \text{if } i = j \\ min_{i \le k < j}\{m_{i,k} + m_{k+1,j} + d_{i-1} \times d_k \times d_j\} & \text{if } i < j. \end{cases}$$

Matrix Chain Multiplication

Example

For $M = M_1 \times M_2 \times M_3 \times M_4$ where $d_0 = 10$, $d_1 = 20$, $d_2 = 50$, $d_3 = 1$, and $d_4 = 100$, the following orders is optimal:

$$(M_1 \times (M_2 \times M_3)) \times M_4 \quad \Longrightarrow \quad cost = 2200$$

m

	1	2	3	4
1	0	10000	1200	2200
2	0	0	1000	3000
3	0	0	0	5000
4	0	0	0	0

k

	1	2	3	4
1	0	1	1	3
2	0	0	2	3
3	0	0	0	3
4	0	0	0	0