Algorithms Design and Analysis

Lecture 2

Beihang University

2017

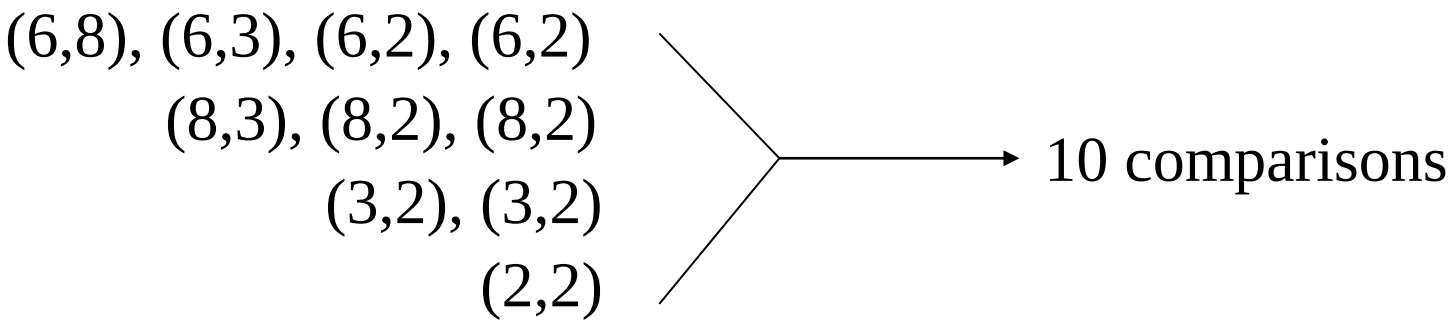
Paradigm (模式) 1: Reduce (归结) to known problem

- Reducing problem A to problem B means that if problem B can be solved then problem A can also be solved.

- Very useful in mathematics.

Example 1: Determine (判断) if an array (数组) of n numbers contains repeated (重复的) elements.

– **Solution 1**: Compare each element to every other element. This uses $O(n^2)$ steps.
– For example, L=(6, 8, 3, 2, 2).



In the worst case, for n elements, the number of operations (运算的次数) needed by this solution method is

$$1+2+3+\ldots+n\text{-}1=\binom{n}{2}=\frac{n(n-1)}{2}=O(n^2)$$

- **Solution 2**: First sort (排序) (by Merge Sort) the n numbers. Then determine (判断) if there is a repeat.

- For example, L=(6, 8, 3, 2, 4).

  Sorting

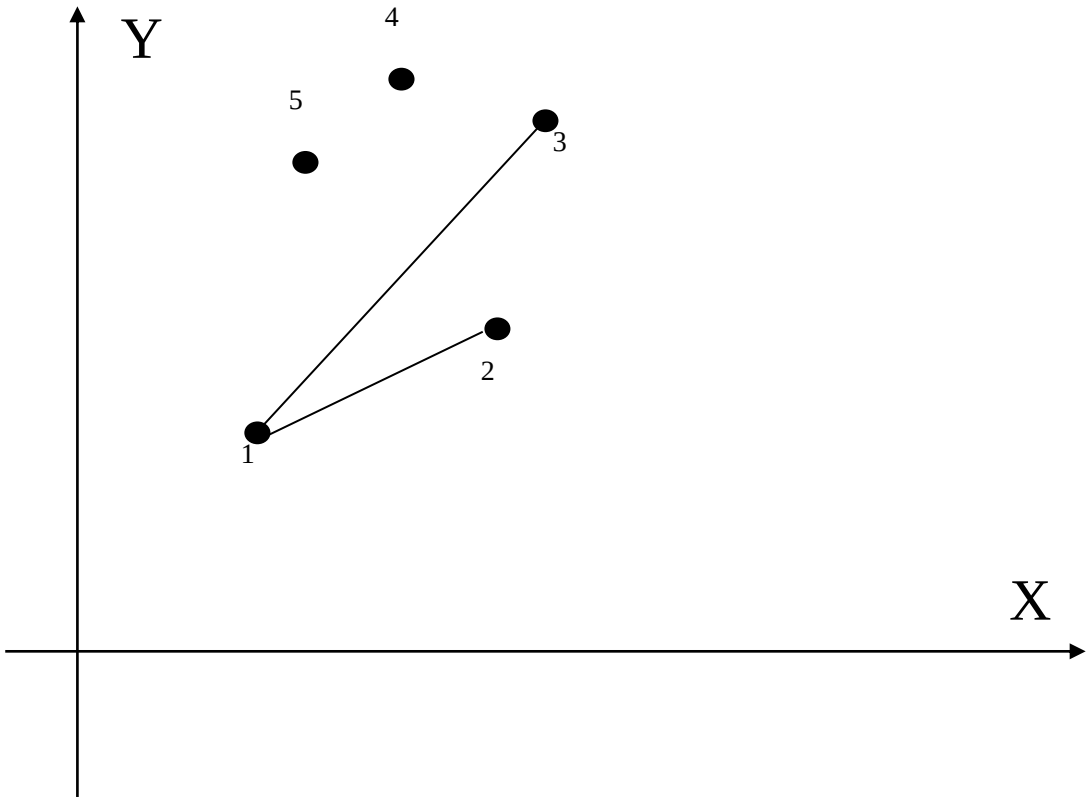
  1. (6, 8, 3, 2, 4) $\Rightarrow$ (2, 3, 4, 6, 8)
  2. Compare (2, 3), (3, 4), (4, 6) and (6, 8).

- Complexity analysis (复杂性分析): Step 1 needs $O(n\log n)$ operations. Step 2 needs $O(n)$ operations. Total: $O(n\log n)+ O(n)= O(n\log n)$ operations.

- If problem A can be solved by reduction to problem B, then the cost of solving A = the cost of reduction + the cost of solving B.

n	n^2	$n\ln n$	$\frac{n^2}{n\ln n}$
100	10^4	460	22
10000	10^8	9.2×10^4	1086
1000000	10^{12}	1.4×10^7	72382

Example 2: Given a list of *n* points in the plane (平面), determine (判断) if any 3 of them are collinear (同线的) (lie on the same line)


Y
4
5
3
2
1
X

Solution 1: Using a triple loop (三重循环), compute all distinct (不同的) triples of points (三个点所组组成的组合).

For i=1 to n

    For j=i+1 to n

        For k=j+1 to n

        Determine if i, j and k lie on the same line

For n points, there are totally $\binom{n}{3} = \frac{n(n-1)(n-2)}{6} = O(n^3)$ distinct triples of points. So this solution method takes $O(n^3)$ time.

Solution 2: Reduction to Example 1.

For each point P in the list do

For each point Q in the list do

Compute the slope (斜率) of the line connecting (连线) and save (保存) it in a list

Determine (Example 1) if the list contains repeated elements.

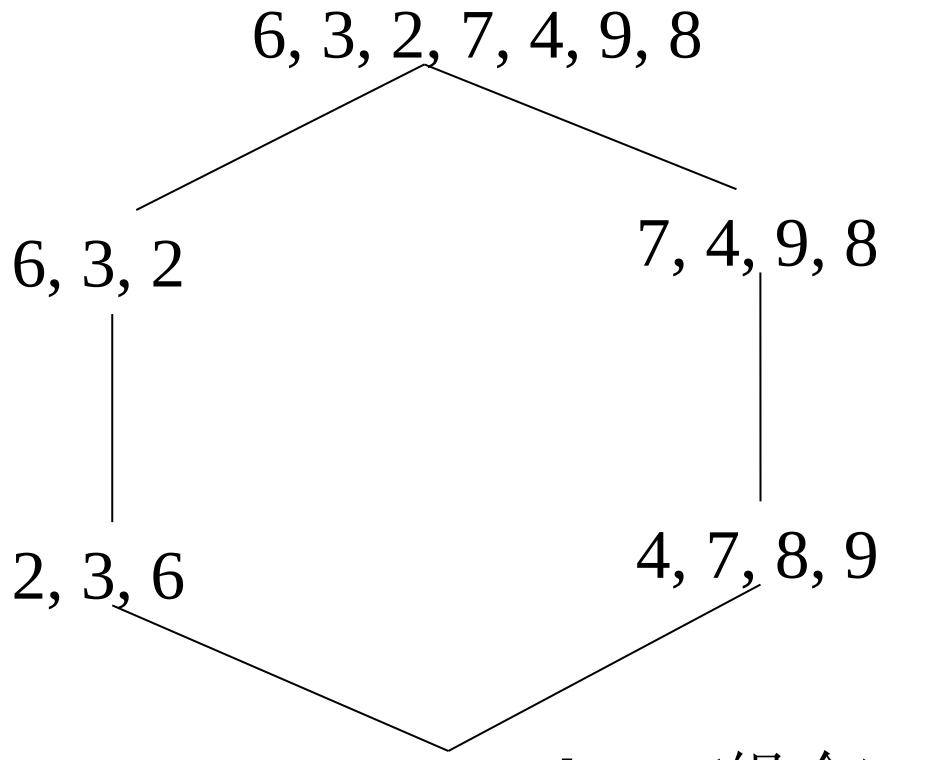
Complexity: $(O(n)+O(n\log n))n=O(n^2\log n)$

Paradigm 2: Divide and Conquer (分而治之)

“Divide et impera” [Divide and rule]
-- Ancient political maxim cited by Machiavelli
“凡治众如治寡，分数是也”
《孙子兵法》

- DIVIDE (划分) problem into smaller subproblems (子问题)
- CONQUER (征服): Solve each subproblem (directly or recursively)
- COMBINE (组合) results together to solve original (原始) problem

Example l. Merge (合并) Sort (J.V. Neumann, 1945)

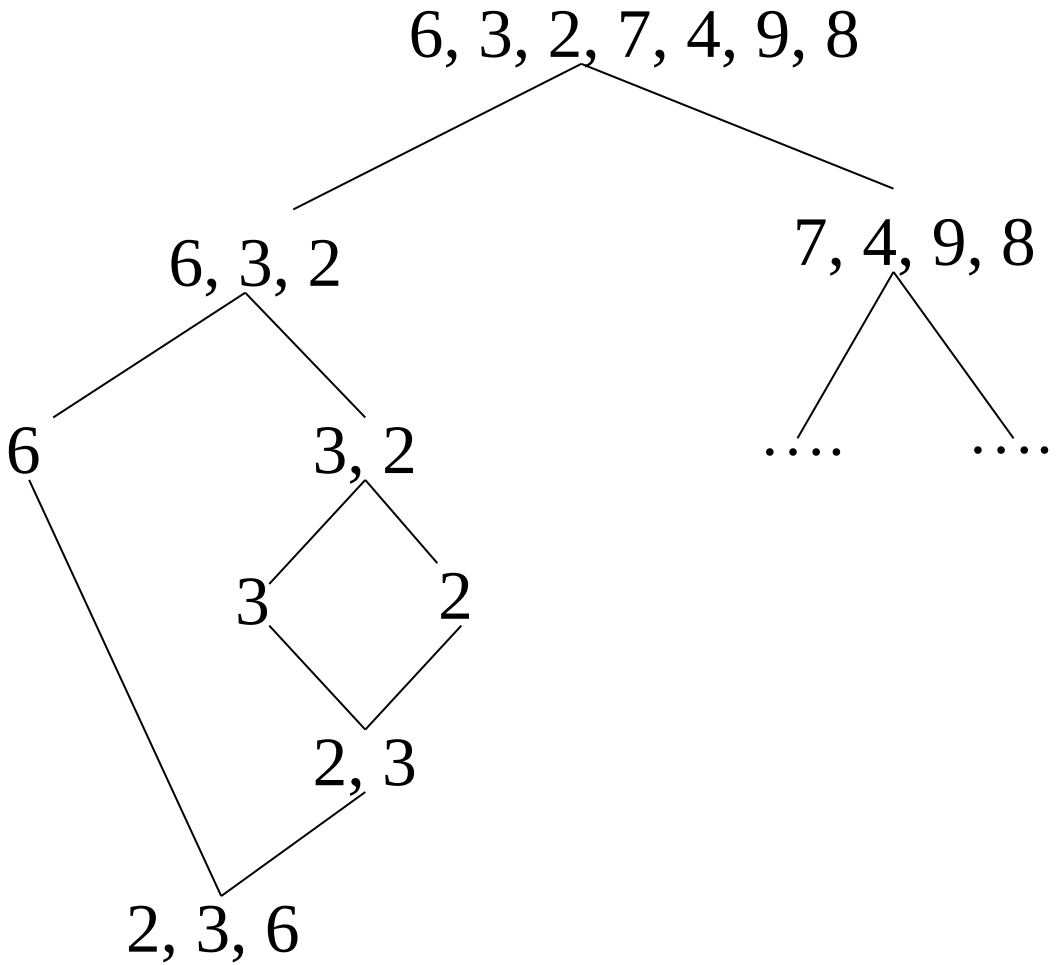


Let us look at the combine step first.

Assume (假定) that we have two sorted (有序) lists L1=(2, 3, 6) and L2=(4, 7, 8, 9).

Step 1. X=empty
Step 2. X=(2), L1=(3, 6), L2=(4, 7, 8, 9).
Step 3. X=(2,3) L1=(6), L2=(4, 7, 8, 9)
Step 4. X=(2,3,4) L1=(6), L2=(7, 8, 9)
Step 5. X=(2,3,4,6) L1=empty, L2=(7, 8, 9)
Step 6. X=(2,3,4,6,7,8,9)

```
MERGE(L1, L2)
    {
    list X = empty
    while (neither L1 nor L2 empty)
    {
        compare first items of L1 & L2
        remove smaller of the two from its list
        add to end of X
    }
    catenate (连接) the remaining list to the end of X
    return X
    }
```


6, 3, 2, 7, 4, 9, 8
6, 3, 2
7, 4, 9, 8
6
3, 2
….
….
3
2
2, 3
2, 3, 6

MERGE-SORT(L)

```
{
 if (length(L) ≤ 1) return L
 else {
        split (分裂) L into lists L1 and L2, each of n/2 elements // O(n)
        L1 = MERGE-SORT(L1)    // T(n/2)
        L2 = MERGE-SORT(L2)    // T(n/2)
        return MERGE(L1,L2)    // O(n)
        }
 }
```

Let T(n) denote the running time of merge-sort on n elements.

Complexity Analysis of Merge-Sort (复杂性分析)

$$T(n)=1 \text{ for } n=1$$

$$T(n)=2T(n/2)+O(n)$$

Applying (应用) the Master Theorem : $O(n \log n)$

Deduction (推导) without using Master Theorem

$$T(n)=2T(n/2)+O(n)$$

$$2T(n/2)=2^2T(n/4)+O(n)$$

$$\cdots$$

$$2^{k-2}T(4)=2^{k-1}T(2)+O(n)$$

$$2^{k-1}T(2)=2^kT(1)+O(n)$$

where $n=2^k$. So we have $T(n)=2^k+kO(n)=n+O(n)\log_2 n=O(n\log n)$.

Remark (注): $O(n)$ can be replaced by cn in the deduction above.

Exercise

Suppose that you take part in a game show. The anchor (主持人) chooses (选择) an integer (整数) between 0 and n: You can only ask questions such as if the integer is 100? He will give answers of YES, LESS or GREATER. How many questions do you need to ask, if you are a smart (聪明) person?

Divide and Conquer (Continue)

Example 2. Multiplication (乘法)

Naive (自然的) pencil-and-paper algorithm

```
  11
×12
----
  22
 11
----
 132
```

```
   31415962
×27182818
----------
  251327696
   31415962
  251327696
   62831924
 251327696
   31415962
 219911734
 62831924
----------
853974377340916
```

Complexity analysis: This uses n^2 multiplications (乘法) and at most $n^2–n$ (加法) additions. So, T(n)=O(n^2). In fact, this is also divide and conquer.

Let (令) X and Y be two *n*-digit(位) numbers. Write

$$X = a\ b$$

$$Y = c\ d$$

where a, b, c and d are $n/2$ digit numbers, e.g. $1364=13\times10^2+64$.

Let $m= n/2$. Then

$$XY = (10^m a+b)(10^m c+d)$$

$$=10^{2m}ac+10^m(bc+ad)+bd$$

Multiply(X; Y; *n*):

if $n = 1$

return $X \times Y$

else

$m = \lceil n/2 \rceil$

/$\lfloor x \rfloor$ is equal to the greatest integer smaller than or equal to x, e.g. $\lfloor 2.5 \rfloor = 2$./

$a = \lfloor X/10^m \rfloor$; $b = X \bmod 10^m$

$c = \lfloor Y/10^m \rfloor$; $d = Y \bmod 10^m$

e = Multiply(a; c; m)

f = Multiply(b; d; m)

g = Multiply(b; c; m)

h = Multiply(a; d; m)

return $10^{2m}e + 10^m(g + h) + f$

Complexity analysis

$$T(1)=1,$$

$$T(n)=4T(\lceil \frac{n}{2} \rceil)+O(n).$$

Applying Master Theorem, we have

$$T(n)= O(n^2).$$

Karatsuba's algorithm

Note that (注意到) $XY = 10^{2m}ac+10^{m}(bc+ad)+bd$, and $bc + ad = ac + bd - (a - b)(c - d)$. So, we have

FastMultiply(X; Y; *n*):

if $n = 1$

    return X×Y

else

    $m = \lceil n/2 \rceil$

    $a = \lfloor X/10^{m} \rfloor$; $b = X \bmod 10^{m}$

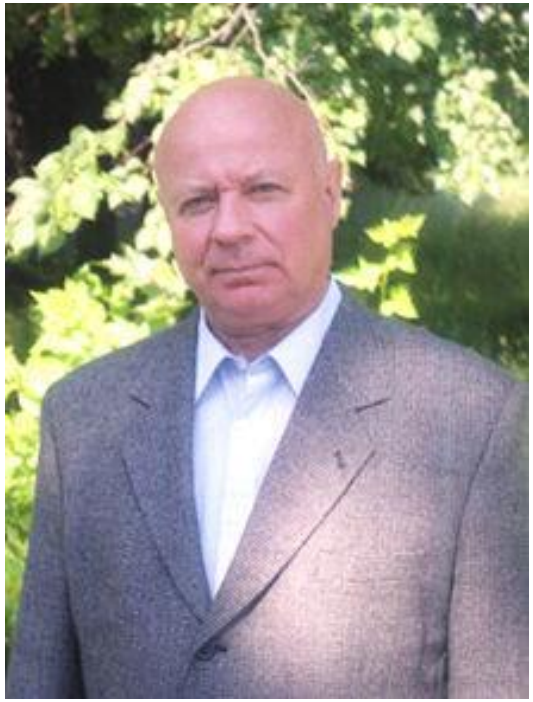
    $c = \lfloor Y/10^{m} \rfloor$; $d = Y \bmod 10^{m}$

    e = FastMultiply(a; c; m)

    f = FastMultiply(b; d; m)

    g = FastMultiply($a-b$; $c-d$; m)

    return $10^{2m}e + 10^{m}(e + f - g) + f$



Anatolii Alexeevitch Karatsuba

(1937-2008)

Complexity analysis

$$T(1)=1,$$

$$T(n) = 3T(\lceil \frac{n}{2} \rceil)+O(n).$$

Applying (应用) Master Theorem, we have

$$T(n)= O(n^{\log_2 3})=O(n^{1.59}).$$

Example 3. Matrix (矩阵) multiplication, C = AB

To multiply a pair of $n \times n$ matrices, standard method computes

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

For example,

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$$c_{11}=a_{11}b_{11}+a_{12}b_{21},\ c_{12}=a_{11}b_{12}+a_{12}b_{22}$$

$$c_{21}=a_{21}b_{11}+a_{22}b_{21},\ c_{22}=a_{21}b_{12}+a_{22}b_{22}$$

Complexity: $O(n^3)$ multiplications and additions (加法).

Divide and Conquer

An $n \times n$ matrix can be divided into four $n/2 \times n/2$ matrices, e.g.

$$\mathbf{A} = \left[\begin{array}{cc:cc} 1 & 2 & 3 & 2 \\ 8 & 3 & 0 & 1 \\ \hdashline 4 & 2 & 7 & 8 \\ 3 & 9 & 5 & 3 \end{array}\right]$$

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$C_{11} = A_{11}B_{11} + A_{12}B_{21},\ C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21},\ C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

Complexity analysis

Total: 8 multiplications (subproblems), and 4 additions.

$$T(1)=1,$$
$$T(n) = 8T(\lceil n/2 \rceil)+n^2.$$

Assume $n=2^k$, we have

$$T(n) = 8T(n/2)+ n^2$$
$$8T(n/2) = 8^2T(n/4)+ 2n^2$$
$$\ldots.$$
$$8^{k-1}T(2) = 8^kT(1)+ 2^{k-1}n^2$$

So, $T(n) = 8^k+ n^2(1+2+2^2\ldots+2^{k-1})=O(n^3)$.

Strassen Algorithm

Define

$P1 = (A_{11}+A_{22})(B_{11}+B_{22})$

$P_2 = (A_{11}+A_{22})B_{11}$

$P_3 = A_{11}\ (B_{11}-B_{22})$

$P_4 = A_{22}\ (-B_{11}+B_{22})$

$P_5 = (A_{11}+A_{12})B_{22}$

$P_6 = (-A_{11}+A_{21})(B_{11}+B_{12})$

$P_7 = (A_{12}- A_{22})(B_{21}+B_{22})$

Then

$C_{11}=P_1+P_4-P_5+P_7,\ C_{12}=P_3+P_5$

$C_{21}=P_2+P_4,\ C_{22}=P_1+P_3-P_2+P_6$

Volker Strassen (1936-)

Complexity analysis

Total: 7 multiplications (subproblems), and 18 additions.

$$T(1)=1,$$
$$T(n) = 7T(\lceil n/2 \rceil)+cn^2.$$

Remark: we have traded multiplications for additions.

Assume $n=2^k$, we have

$$T(n) = 7T(n/2)+ cn^2$$

$$7T(n/2) = 7^2T(n/4)+ 7cn^2/4$$

$$7^2T(n/4) = 7^3T(n/8)+ 7^2cn^2/4^2$$

$$\ldots.$$

$$7^{k-1}T(2) = 7^kT(1)+ 7^{k-1}c2^2$$

So,

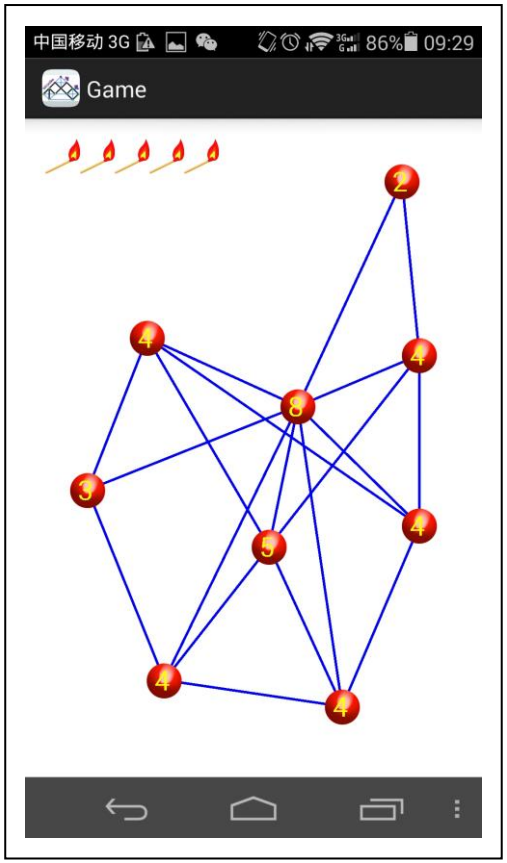
$$T(n) = 7^k+ cn^2(1+7/4+7^2/4^2+\ldots+7^{k-1}/4^{k-1})$$

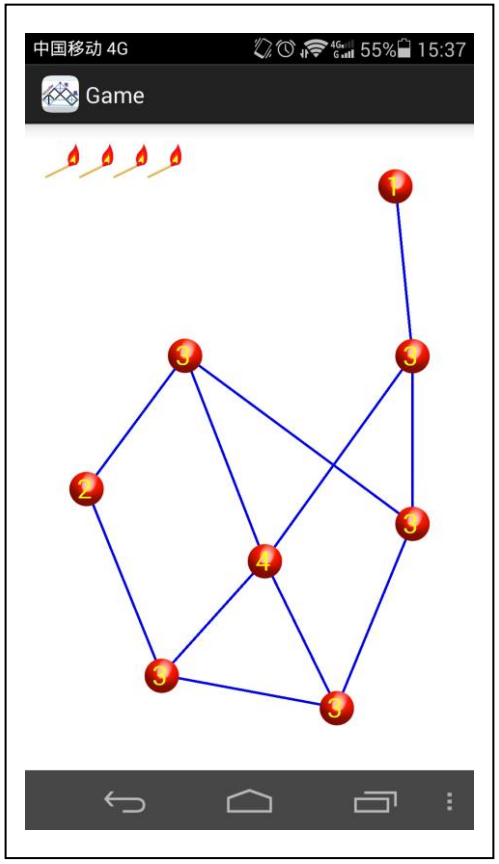
$$=O(7^k)+c4^kO(7^k/4^k)$$

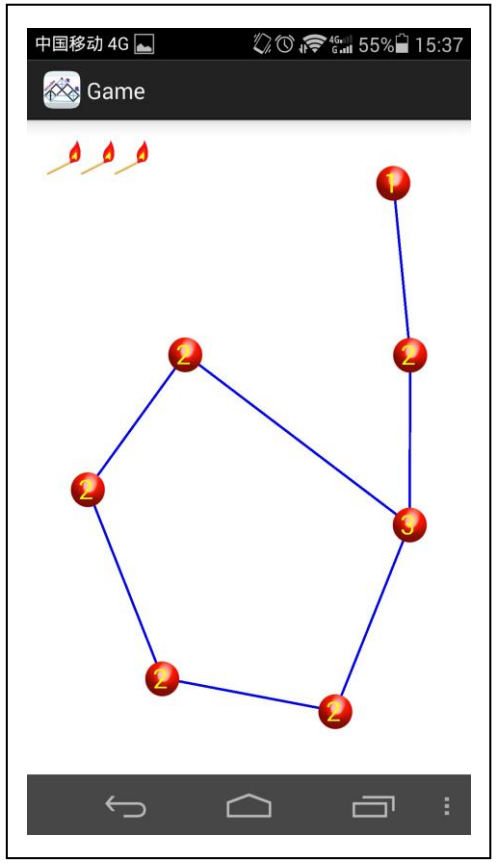
$$= O(7^{\log_2 n}) = O(7^{\log_7 n \log_2 7})= O(n^{\log_2 7})=O(n^{2.81}).$$

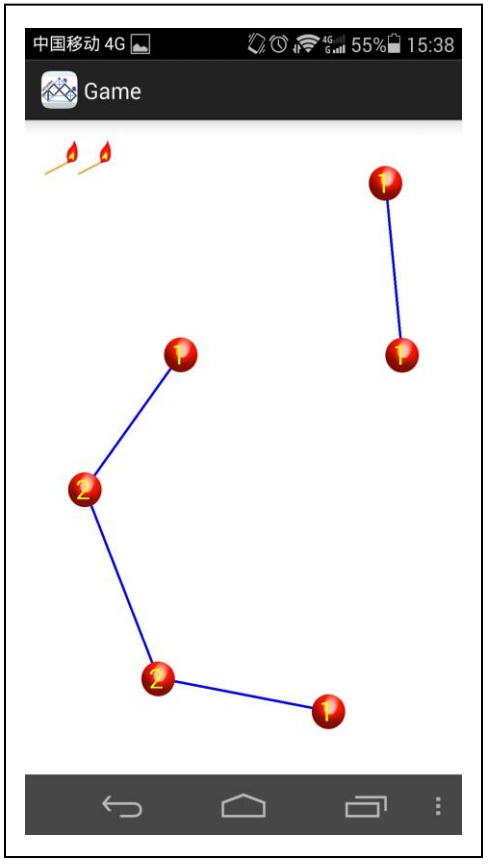
Remark: $1+a+a^2+\ldots+a^{k-1}=(a^k-1)/(a-1)$

本学期大作业


中国移动 3G 86% 09:29
Game


中国移动 4G 55% 15:37
Game


中国移动 4G 55% 15:37
Game

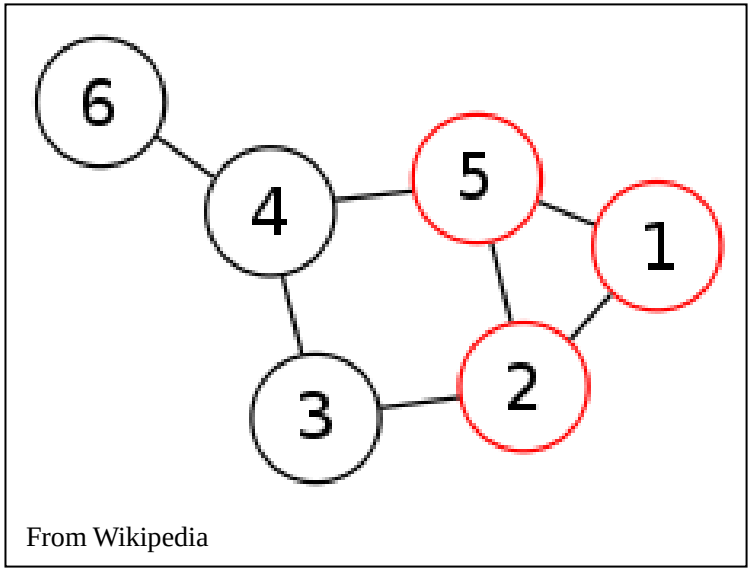

中国移动 4G 55% 15:38
Game

Maximum Clique

A *clique* of a graph is a set of nodes such that every pair of nodes is connected by an edge.

A *maximum clique* of a graph is a clique with the largest number of nodes.

The *maximum clique problem* is to find a maximum clique in a given graph.



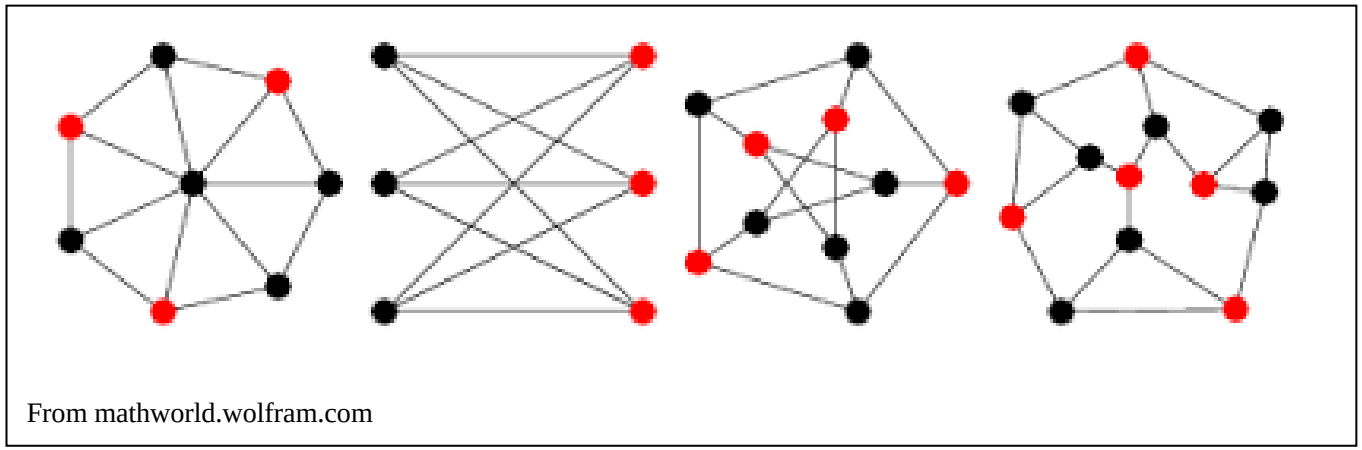
From Wikipedia

Maximum Independent Set

An *independent set* of a graph is a set of nodes such that every pair of nodes is not connected by an edge.

A *maximum independent set* of a graph is an independent set with the largest number of nodes.

The *maximum independent set problem* is to find a maximum independent set in a given graph.



From mathworld.wolfram.com

下载：

手机游戏：

http://www.nlsde.buaa.edu.cn/~kexu/games/RBGame_G.apk

https://github.com/notbad/RBGame

测试实例：

http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm

http://iridia.ulb.ac.be/~fmascia/maximum_clique/BHOSLIB-benchmark

要求：

- 分组（1 人-3 人）
- 集中授课结束后分班讨论，每组需提交一个 ppt 报告并指派一人在课堂上做报告。
- 如果报告由多人合作完成，需注明每人贡献的百分比。

Homework 4

令 A[1..n] 是一个由 n 个数所组成的数组，定义 $d = \min_{1 \le i \ne j \le n} |\mathrm{A}(i) - \mathrm{A}(j)|$ 为数组 A 的**差值**，其中 $|a|$ 表示 a 的绝对值。

设计一个求数组**差值**的算法（用伪码描述）并分析算法的时间复杂性。