Introduction to Algorithms

ZHENG Hong
Department of Computer Science and Engineering
East China University of Science and Technology

Welcome to *Introduction to Algorithms,* Fall 2018

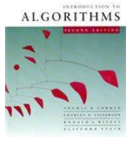
- **Handouts**
- Course Information
- Calendar
- Registration (ECUST students only)
- References
- Objectives and Outcomes

Course information

1.	Staff	6.	Textbook
2.	Prerequisites	7.	Collaboration policy
3.	Lectures	8.	Problem sets
4.	Recitations	9.	Grading policy
5.	Handouts	10.	Collaboration policy

Textbook

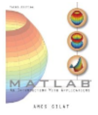
Introduction to Algorithms (2nd)
by Thomas H.Cormen, Charles E.Leiserson, Ronald L.Rivest, Clifford Stein (CLRS) MIT Press



Textbooks

- Intro to Engineering Programming
  - James Holloway, 2004
  - ISBN: 0471202150

- MATLAB: An Introduction with Applications, 3rd (or newer) Edition
  - Amos Gilat, 2007
  - ISBN: 0470108770



Grading policy

- Class Attendence-----5%
- Class Presentation-----25%
  - ●ppt（5%）
  - ●Content: （10%）
  - ●Oral Report（10%）
- Final Exam-----70%

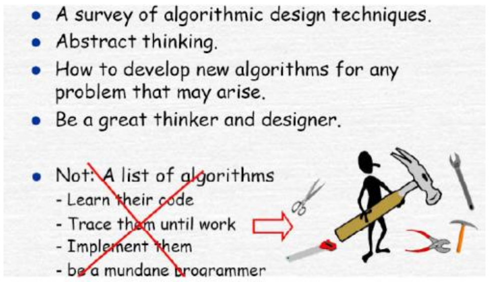
➢Students must choose a topic,and attend a term discussion.
➢Introduction example, about 2-4 students report each time, and each student prepares about 15-20 minutes.

Course Objectives

This course introduces students to the analysis and design of computer algorithms.
Upon completion of this course, students will be able to do the following:
Analyze the asymptotic performance of algorithms.

- Demonstrate a familiarity with major algorithms and data structures.
- Apply important algorithmic design paradigms and methods of analysis.
- Synthesize efficient algorithms in common engineering design situations.

Course Objectives



- A survey of algorithmic design techniques.
- Abstract thinking.
- How to develop new algorithms for any problem that may arise.
- Be a great thinker and designer.

- ~~Not: A list of algorithms~~
  - ~~Learn their code~~
  - ~~Trace them until work~~
  - ~~Implement them~~
  - ~~be a mundane programmer~~

TOPICS

- ***Introduction (Asymptotic notation, Divide and conquer)***
- ***Sorts (Heapsort, Quicksort, Sorting in Linear Time)***
- ***Data Structures***
- ***Advanced Design and Analysis Techniques***
- ***Recurrences***
- ***Divide-and-Conquer***
- ***Dynamic Programming***
- ***Greedy Algorithm***
- ***Graph Searching: Depth-first Search, Topological Sort, DAG Shortest Paths***

What will you get from this course?

- You will learn how to “think like an engineer”
- You will learn about algorithms and how to design/implement them
- You will learn programming in C++ and MATLAB

- You will have fun!

What is an algorithm?

- A list of instructions for accomplishing a task that may be executed by a mechanism.

Algorithm

- A list of instructions that, when executed, transform information from input to output.
- The instructions are a finite set of steps that can be executed, in a definite order, by a deterministic mechanism.
- When these steps are actually executed,
- the execution must terminate after a finite time.

Key Point

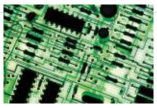
Examples of “famous” algorithms

- Web search: (Page Rank)
- Recommendations: (you want to buy…)
- Mp3 encoding: (and decoding)

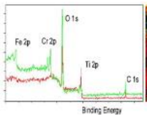
iTunes

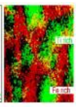
Other uses of algorithms

- Circuit Design



- Materials Analysis





- Air Pollution Modeling

Other uses of algorithms

- Air Traffic Control

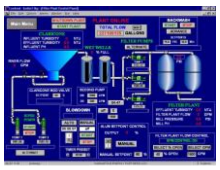
- Magnetic Resonance Imaging

- Computer Numerical Control

Other uses of algorithms

- Chemical Analysis

- Process Control Simulation



Question 1 – Is it an algorithm?

1. Request a value for A
2. Request a value for B
3. Request a value for C
4. Assign B×B-4×A×C to D
5. Return D

a) No, it is not an algorithm
b) Yes, it returns the solution to a quadratic equation $Ax^2+Bx+C=0$.
c) Yes, it returns the discriminant for a quadratic equation $Ax^2+Bx+C=0$.

Question 1 – Is it an algorithm?

1. Request a value for A
2. Request a value for B
3. Request a value for C
4. Assign B×B-4×A×C to D
5. Return D

a) No, it is not an algorithm
b) Yes, it returns the solution to a quadratic equation $Ax^2+Bx+C=0$.
c) Yes, it returns the discriminant for a quadratic equation $Ax^2+Bx+C=0$.

Question 2 – Is it an algorithm?

```
1. Request a value for A
2. While A>0
3.     Assign A/2 to A
4. Return A
```

Note: Assume that A is a positive number.

a) No, it is not an algorithm
b) Yes, it returns a value of one-half A.
c) Yes, it returns the smallest even divisor of A.

Question 2 – Is it an algorithm?

```
1. Request a value for A
2. While A>0
3.     Assign A/2 to A
4. Return A
```

The algorithm will never end.

a) ~~No, it is not an algorithm~~
b) ~~Yes, it returns a value of one-half~~ A.
c) Yes, it returns the smallest even divisor of A.

Question 3 – Is it an algorithm?

```
1. Request a value for A
2. Request a value for B
3. While B>A
4.     Request a new value for B
5. Assign B/A to C
6. Return C
```

Note: Assume that A and B are positive numbers and that line 6 will eventually execute.

a) No, it is not an algorithm
b) Yes, it returns a value <= 1.
c) Yes, it returns a value >= 1.
d) Yes, it returns 1

Question 3 – Is it an algorithm?

```
1. Request a value for A
2. Request a value for B
3. While B>A
4.     Request a new value for B
5. Assign B/A to C
6. Return C
```

a) No, it is not an algorithm
b) ~~Yes, it returns a value <= 1~~.
c) ~~Yes, it returns a value >= 1~~.
d) Yes, it returns 1

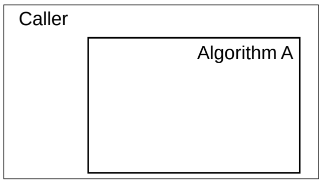
Skills you will develop when mastering algorithms

- Logical thinking
- Ability to analyze a process
- Capability to identify and troubleshoot problems
- Patience and persistence

Administrative Business

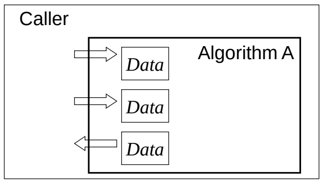
How are algorithms invoked?

- Every algorithm has a caller.
- An algorithm is usually called by another algorithm.



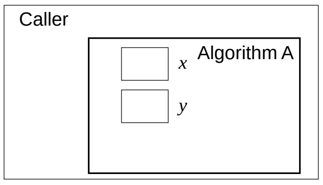
How is data handled?

- Data is exchanged between the algorithm and its caller.
- Since our definition of an algorithm is that it transforms input to output there must be a way to accept input and provide output to the caller.



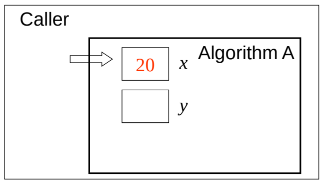
Example: Remainder Algorithm

```
1.  Request a value for $x$
2.  Request a value for $y$
3.  While $x \geq y$ do
4.      Assign $x$-$y$ to $x$
5.  Return $x$
6.  End
```



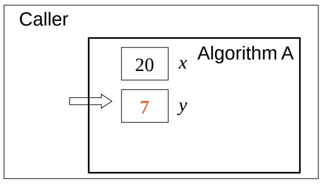
Example: Remainder Algorithm

```
1.  Request a value for $x$
2.  Request a value for $y$
3.  While $x \geq y$ do
4.      Assign $x$-$y$ to $x$
5.  Return $x$
6.  End
```



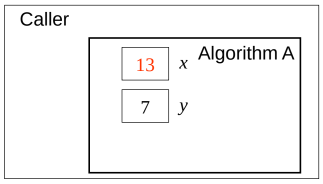
Example: Remainder Algorithm

```
1.  Request a value for $x$
2.  Request a value for $y$
3.  While $x \geq y$ do
4.      Assign $x$-$y$ to $x$
5.  Return $x$
6.  End
```



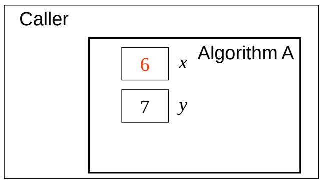
Example: Remainder Algorithm

```
1.  Request a value for $x$
2.  Request a value for $y$
3.  While $x \geq y$ do
4.      Assign $x$-$y$ to $x$
5.  Return $x$
6.  End
```



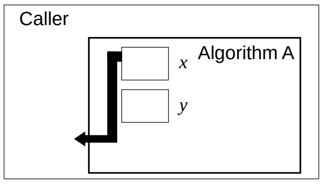
Example: Remainder Algorithm

```
1. Request a value for x
2. Request a value for y
3. While x ≥ y do
4.     Assign x-y to x
5. Return x
6. End
```



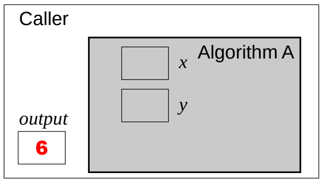
Example: Remainder Algorithm

```
1. Request a value for x
2. Request a value for y
3. While x ≥ y do
4.     Assign x-y to x
5. Return x
6. End
```



Example: Remainder Algorithm

```
1. Request a value for x
2. Request a value for y
3. While x ≥ y do
4.     Assign x-y to x
5. Return x
6. End
```



Tips for success

- Attend lecture and lab regularly
- Get help when you need it
- Make use of office hours
- Sleep/eat regularly
- Be patient and persistent

Next Lecture

- C++ Basics and Input/Output

Analysis of algorithms

- The theoretical study of computer-program performance and resource usage.
- What’s more important than performance?

• modularity	• user-friendliness
• correctness	• programmer time
• maintainability	• simplicity
• functionality	• extensibility
• robustness	• reliability

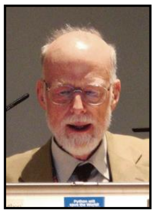
Why study algorithms and performance?

- Algorithms help us to understand scalability.
- Performance often draws the line between what is feasible and what is impossible.
- The lessons of program performance generalize to other computing resources.
- Program = data structure + algorithm

1974年图灵奖获得者：唐纳德.克努特（Donald Ervin Knuth，中文名：高德纳）

29岁提出了算法与数据结构的概念，31岁开始出版他的历史性经典巨著，34岁获得图灵奖，花费10年时间打造一个西方世界普遍使用的排版系统，还免费提供使用。学习算法的同学自然会知道《The Art of Computer Programming》这本书，该书计划共写7卷，仅仅出版三卷之后，就已经震惊世界，此书与牛顿的“自然哲学的数学原理”等一起，被评为“世界历史上最伟大的十种科学著作”之一。
学习编译原理与数据结构的同学，自然也会感叹KMP算法（一种字符串匹配算法）和LR(k)算法有多么不可思议，可是类似这样的算法在那三卷书中比比皆是。

Quicksort算法之父——1980年图灵奖获得者查尔斯•霍尔



快速排序算法的发明者是获得1980年度图灵奖的英国牛津大学计算机科学家查尔斯.霍尔（Charles Hoare），他还发明了CASE和程序设计语言的公理化方法。程序设计语言的公理化定义方法是用一组公理和一组规则描写语言应有的性质，从而使语言与具体实现的机器无关，而且也易于证明程序的正确性。获得图灵奖不仅是因为他发明了CASE和QUICKSORT，而是因为他在编程语言的定义和设计方面的基础性贡献。

http://www.cqvip.com/Read/Read.aspx?id=32099677

用算法解决现实问题

图灵奖获得者惠普高级院士罗伯特•塔扬教授

罗伯特•塔扬（Robert Tarjan）教授是世界知名计算机学家，他的研究领域主要包括图论、算法和数据结构设计。罗伯特教授是许多图论算法的发明者，比如树中最近共同祖先离线算法、Splay trees、Fibonacci heaps、平面性检测（Planarity testing）等。1986年，他与John Hopcroft因为在算法及数据结构的设计和分析中所取得的成果而荣获图灵奖。他于1982年获得首届奈望林纳奖（Nevanlinna Prize），现为美国科学院院士、美国计算机协会（ACM）院士、美国普林斯顿大学教授。

http://blog.sina.com.cn/s/blog_5e13f6110102dy7s.html?tj=1

http://www.ccf.org.cn/resources/1190201776262/2012/05/18/14.pdf

图灵奖获得者John E. Hopcroft教授

约翰·E·霍普克洛夫特（John E. Hopcroft），美国康奈尔大学智能机器人实验室主任、计算机科学系工程与应用数学的IBM教授，世界计算机科学最高奖图灵奖获得者，美国国家科学院与工程院院士。1961年在西雅图大学获得电气工程学士学位。研究方向主要是计算机科学理论，为评价算法可观的判断标准提出了算法最坏情况下的鉴定算法。他的深入算法是计算机科学的经典教材，也因此被誉为算法大师。1986年因为在数据结构和算法设计与分析中的重要的基础性的贡献而获得图灵奖。2005年获得IEEE哈里·古德纪念奖。2007年获得计算机研究协会的杰出贡献奖。著作有《算法设计与分析基础》、《数据结构与算法》、《自动机理论、语言和计算导论》等。

http://www.cs.cornell.edu/jeh/

图灵奖得主——姚期智院士

姚期智，祖藉湖北孝感，1946年出生于上海，幼年随父母移居台湾。1967年毕业于台湾大学，之后赴美国深造。1972年获哈佛大学物理学博士学位，1975年获伊利诺大学香槟分校计算机科学博士学位。曾先后在麻省理工学院、斯坦福大学、加州大学伯克利分校等美国高等学府从事教学和研究，1986年至2004年6月任普林斯顿大学计算机科学系教授。2004年9月正式加盟清华大学高等研究中心任全职教授。他是美国国家科学院院士、中国科学院外籍院士、美国人文及科学院院士。 因为对计算理论的诸多贡献，2000年，美国计算机学会把该年度的图灵奖授予他，使他成为自图灵奖创立以来首位获奖的华裔学者。

图灵奖获得者John E. Hopcroft教授

▪**John E. Hopcroft** is the IBM Professor of Engineering and Applied Mathematics in Computer Science at Cornell University.

▪Hopcroft's research centers on theoretical aspects of computing, especially analysis of algorithms, automata theory, and graph algorithms. He has coauthored four books on formal languages and algorithms with Jeffrey D. Ullman and Alfred V. Aho.

▪His most recent work is on the study of information capture and access.

▪He was honored with the A. M. Turing Award in 1986.

http://www.cs.cornell.edu/jeh/

Why to study the Algorithm

- **Performance often draws the line between what is feasible and what is infeasible.**
- **Algorithmic mathematics provides a language for talking about program behavior.**
- **Performance is the currency of computing.**
- **Speed is fun!**

Give an algorithm

➢You will often be called upon to “give an algorithm” to solve a certain problem.

➢Your write-up should take the form of a short essay.

➢A topic paragraph should summarize the problem you are solving and what your results are.

The Algorithm

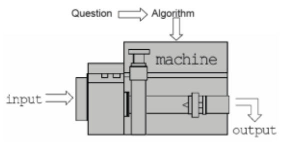
The body of the essay should provide the following:

① A description of the algorithm in English and, if helpful, pseudo-code.
② At least one worked example or diagram to show more precisely how your algorithm works.
③ A proof (or indication) of the correctness of the algorithm.
④ An analysis of the running time of the algorithm.

□Remember, your goal is to communicate.
□Full credit will be given only to correct solutions which are described clearly.
□Convoluted and obtuse descriptions will receive low marks.

2-What’s an Alrogithm?

- **Algorithm： A well-defined computational procedure that takes some value,or set of values, as input and produces some value,or set of values, as output.**



The theoretical study of computer-program performance and resource usage.

- **What’s more important than performance?**
- **– modularity**
- **– correctness**
- **– maintainability**
- **– robustness**
- **– user-friendliness**
- **– programmer time**
- **– simplicity**
- **– extensibility**
- **– reliability**

- **issues: correctness, efficiency(amount of work done and space used), storage(simplicity, clarity), optimality.etc.**

The theoretical study of computer-program performance and resource usage.

- What's more important than performance?

- – modularity
- – correctness
- – maintainability
- – robustness
- – user-friendliness
- – programmer time
- – simplicity
- – extensibility
- – reliability

Example 1——百鸡问题

公元5世纪末，我国古代数学家张丘建在他所撰写的《算经》中，提出了这样一个问题：“鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、母、雏各几何？”意思是公鸡每只5元、母鸡每只3元、小鸡3只1元，用100元钱买100只鸡，求公鸡、母鸡、小鸡的只数。

案例一——百鸡问题

令*a*为公鸡只数，*b*为母鸡只数，*c*为小鸡只数。列出约束方程：

$a+b+c=100$　　（1）

$5a+3b+c/3=100$　　（2）

$c\%3=0$　　（3）

分析: *a*、*b*、*c*的可能取值范围为0~100，对*a*、*b*、*c*的所有组合进行测试，满足约束方程的组合是问题的解。把问题转化为用 *n* 元钱买 *n* 只鸡，则上式变为：

$a+b+c=n$　　（1'）

$5a+3b+c/3=n$　　（2'）

算法1 百鸡问题

```
void chicken_question(int n,int &k,int g[],int m[],int s[])
{
   int  a,b,c;
   k = 0;
   for (a=0;a<=n;a++){
      for (b=0;b<=n;b++){
         for (c=0;c<=n;c++) {
            if ((a+b+c==n)&&(5*a+3*b+c/3==n)&&(c%3==0)) {
               g[k] = a;
               m[k] = b;
               s[k] = c;
               k++;
            }
         }
      }
   }
}
```

方案一：用三层循环，内循环100万余次
101*101*101＝1030301
☹

算法2 改进的百鸡问题

```
void chicken_problem(int n,int &k,int g[],int m[],int s[])
{int  i,j,a,b,c; k = 0; i = n/5; j = n/3;
for (a=0;a<=i;a++){
      for (b=0;b<=j;b++) {
          c = n–a–b;
          if ((5*a+3*b+c/3==n)&&(c%3==0)) {
              g[k] = a;
              m[k] = b;
              s[k] = c;
              k++;
          }
      }
   }
}
```

方案二：公鸡最多20只数，b为母鸡只数最多最多34只，优化后内循环525次。
🙂

Example 2--Greatest common divisor.

- What is the greatest common divisor of positive integers m and n ?
- **gcd(m,n),** which are not all zero, is the largest positive integer that divides each of the integers.

- Jump to searchIn mathematics, the **greatest common divisor** (**gcd**) of two or more integers, which are not all zero, is the largest positive integer that divides each of the integers.
- For example, the gcd of 8 and 12 is 4.
- The greatest common divisor is also known as the **greatest common factor** (**gcf**) or **highest common divisor**。

54

Example 2--Greatest common divisor.

- What is the greatest common divisor of 54 and 24?
- The number 54 can be expressed as a product of two integers in several different ways:
- 54 × 1 = 27 × 2 = 18 × 3 = 9 × 6
- Thus the divisors of 54 are:
- 1 , 2 , 3 , 6 , 9 , 18 , 27 , 54.
- Similarly, the divisors of 24 are:
- 1 , 2 , 3 , 4 , 6 , 8 , 12 , 24.
- The numbers that these two lists share in common are the common divisors of 54 and 24:
- 1 , 2 , 3 , 6.

The greatest of these is 6. That is, the greatest common divisor of 54 and 24. One writes:

- gcd (54 , 24) = 6.

55

Example 2.

- What is the greatest common divisor of positive integers m and n ?
- **gcd(m,n),** which are not all zero, is the largest positive integer that divides each of the integers.
- **m mod n=0**
  - gcd(m, n) = gcd(n, m mod n)

The greatest common divisor GCD(m,n) of two positive integers m and n, sometimes written (m,n),is the largest divisor common to m and n, For example,(1,2)=1,(12,18)=6. (m,n) can be easily found by the Euclidean algorithm.

Euclid(m,n)

```
//gcd(m,n)
//Input：两个不全为0的非负整数m,n
//Output： the greatest common divisor of positive integers m and n

While n!=0 do
    r←m mod n
    m←n
    n←r
Return m
```

56

Example

gcd(m,n)的连续整数检测算法

第一步：将min{m,n}的值赋给t。
第二步：m除以t，如果余数为0，则进入第三步；否则，进入第四步。
第三步：n除以t，如果余数为0 ，则返回t值作为结果；否则，进入第四步。
第四步：把t的值减1。返回第二步。

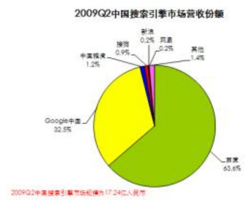
同一个问题有不同的解决方法

gcd(m,n)的中学计算算法

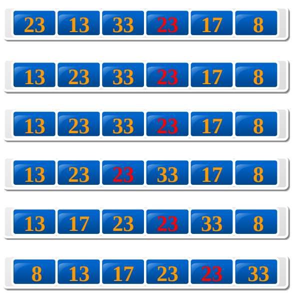
第一步：找到m的所有质因数。
第二步：找到n的所有质因数
第三步：从第一步和第二步中求得的质因数分解式找出所有的公因数
第四步：将第三步中找的质因数相乘，其结果作为给定数字的最大公因数。

57

Example 3——Search



Example 3——Sort



Example 4.——Sort

```
void  insertSort (int  r[ ], int n) {
   for (i=2; i<=n; i++) {
         r[0]=r[i];
              j=i-1;
              for (j=i-1;r[0]<r[j];j--) {
                  r[j+1]=r[j];
                  j=j-1;
              }
             r[j+1]=r[0];
   }
}
```

Example 4.——Graph

Example 5——Combination

最小乘车费用

假　设	1	2	3	4	5	6	7	8	9	10
费　用	12	21	31	40	49	58	69	79	90	101

而任意一辆汽车从不行驶超过10公里。某人想行驶n公里，假设他可以任意次换车，请你帮他找到一种乘车方案，使得总费用最小。

3-What will you get from this course?

- You will learn how to “think like an engineer”
- You will learn about algorithms and how to design/implement them
- You will learn programming in C++ and MATLAB

- You will have fun!

Problem-1

- Problem-1
- The 3 teachers predicted a student competition. Their predictions are as follows:
- A said: student A is the first, student B is the third.
- B said: Student C was the first and student D was the fourth.
- C said: Student D is the second, student A is the third.
- The results show that they are half right, and half wrong with no ranking .
- Try to program the input a, b, c, d their respective rankings .

问题-1

- 3位老师对某次学生竞赛进行了预测，他们的预测如下：
- 甲说：学生A得第一名，学生B得第三名。
- 乙说：学生C得第一名，学生D得第四名。
- 丙说：学生D得第二名，学生A得第三名。
- 竞赛结果表明，它们各说对了一半，说错了一半，并且无名次并列，试编程输入a,b,c,d各自的名次。

Problem-2

A polygon is ***convex*** if all of its internal angles are less than 180° (and none of the edges cross each other). Figure 1 shows an example. We represent a convex polygon as an array $V[1..n]$ where each element of the array represents a vertex of the polygon in the form of a coordinate pair (x, y). We are told that $V[1]$ is the vertex with the minimum x coordinate and that the vertices $V[1..n]$ are ordered counterclockwise, as in the figure. You may also assume that the x coordinates of the vertices are all distinct, as are the y coordinates of the vertices.

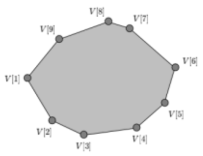


Figure 1: An example of a convex polygon represented by the array $V[1..9]$. $V[1]$ is the vertex with the minimum x-coordinate, and $V[1..9]$ are ordered counterclockwise.

(b) Give an algorithm to find the vertex with the maximum x coordinate in $O(\lg n)$ time.
(c) Give an algorithm to find the vertex with the maximum y coordinate in $O(\lg n)$ time.

A convex polygon

- A convex polygon is a simple polygon (not self-intersecting) in which no line segment between two points on the boundary ever goes outside the polygon. Equivalently, it is a simple polygon whose interior is a convex set.[1]
- In a convex polygon, all interior angles are less than or equal to 180 degrees, while in a strictly convex polygon all interior angles are strictly less than 180 degrees.

The properties of a simple polygon

- The following properties of a simple polygon are all equivalent to convexity:

① Every internal angle is less than or equal to 180 degrees.
② Every point on every line segment between two points inside or on the boundary of the polygon remains inside or on the boundary.
③ The polygon is entirely contained in a closed half-plane defined by each of its edges.
④ For each edge, the interior points are all on the same side of the line that the edge defines.
⑤ The angle at each vertex contains all other vertices in its edges and interior.
⑥ The polygon is the convex hull of its edges.