Computer Science
North University of China

CS101 Fall 2017

Introduction to Computer Science

Instructor: Qingsong Guo
School of Computer Science & Technology

http://abelgo.cn/cs101.html

Algorithm: The Art of Problem-Solving

Sequential, Conditional, Iterative, and Recursive Structures

Algorithm in nutshell

The concept of algorithm

- A stepwise process
- Algorithm is the art of problem solving

Algorithm representation

- Pseudocode and primitives

Commonly used structures

- Sequential structure
- Conditional structure
- Iterative structures
- Recursive structures (will be discussed later)

Efficiency and correctness

- Measure the efficiency of various algorithms

What is an algorithm?

Computers are capable of performing many complicated functions.

However, a computer must be told exactly what to do

- It does not have free will, in the same way as human beings, animals, insects, etc.
- Computers must be provided a deterministic sequence of instructions

This notion of a sequence of instructions is called an **algorithm.** An **algorithm** is a list of instructions for performing a specific task, or, for solving a particular type of problem.

But this is not the exact definition for algorithm. The definition relies on some other concepts such as Turing machine, Lambda calculus, recursive function, etc.

What is an algorithm? (cont.)

Properties

- An algorithm is an ordered set of unambiguous, executable steps that defines a terminating process.
- Unambiguity gives deterministic results
- An algorithm must be executable and can terminate as to give out the result.

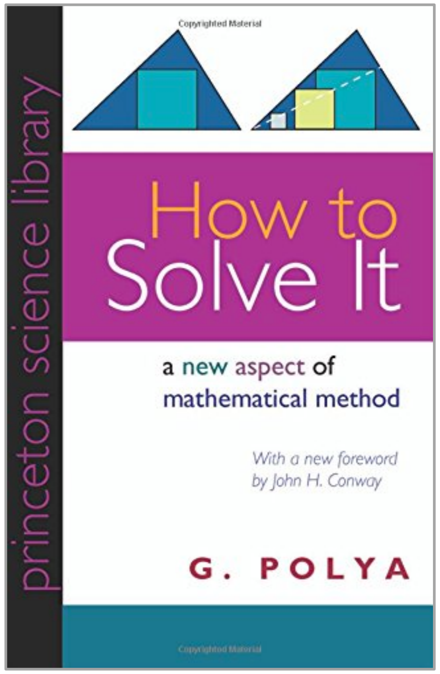
Algorithm representation

- Flowchart (流程图)
- Describe an algorithm with **pseudocode** (伪代码)
- Requires a set of well-defined **primitives** (原语)

G. Polya's problem-solving steps

G. Polya's 4-step process

- 1. Understand the problem.
- 2. Devise a plan for solving the problem.
- 3. Carry out the plan.
- 4. Evaluate the solution


Copyrighted Material
princeton science library
How to Solve It
a new aspect of mathematical method
With a new foreword by John H. Conway
G. POLYA
Copyrighted Material

How to solve it.
《如何解题》

Problem solving philosophy

We have many approaches to carry out Polya’s problem-solving steps.

Top-down methodology (自顶向下)

- Divide and conquer (分而治之): divide the problem into smaller problems which are easier to solve.
- The sub-problems will be divided into even smaller ones until the entire problem has been reduced to a collection of easily solved problems.

Bottom-up methodology (自底向上)

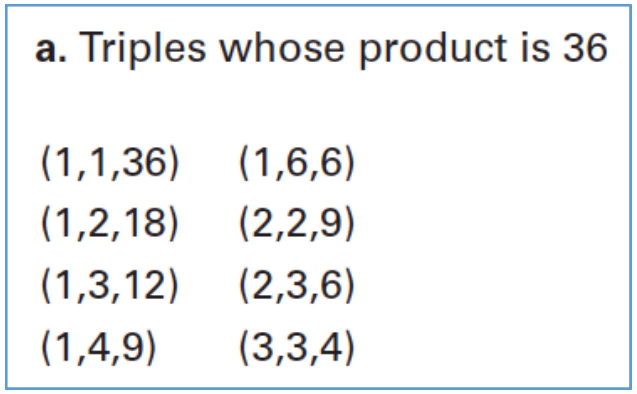
- Solve an easier related problem
- Relax some of the problem constraints
- Solve pieces of the problem first and progress from the specific to general (从特殊到一般)

Problem 1: ages of children problem

Person A is charged with the task of determining the ages of B's three children.

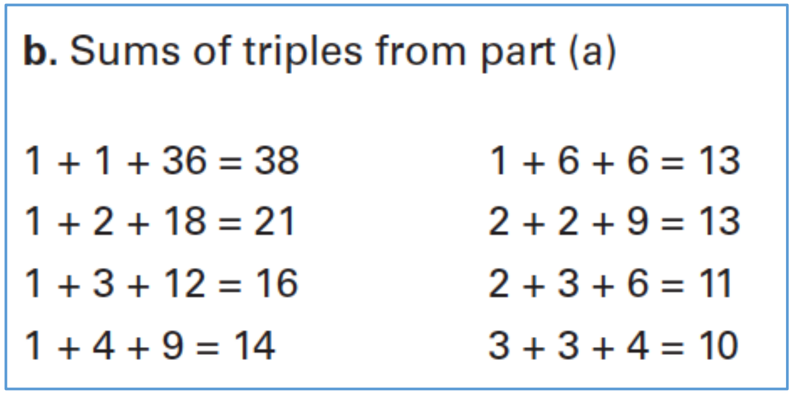
- B tells A that the product (乘积) of the children's ages is 36.
- A replies that another clue is required.
- B tells A the sum of the children's ages.
- A replies that another clue is needed.
- B tells A that the oldest child plays the piano.
- A tells B the ages of the three children.

How old are the three children?



a. Triples whose product is 36

(1,1,36)	(1,6,6)
(1,2,18)	(2,2,9)
(1,3,12)	(2,3,6)
(1,4,9)	(3,3,4)



b. Sums of triples from part (a)

1 + 1 + 36 = 38	1 + 6 + 6 = 13
1 + 2 + 18 = 21	2 + 2 + 9 = 13
1 + 3 + 12 = 16	2 + 3 + 6 = 11
1 + 4 + 9 = 14	3 + 3 + 4 = 10

Problem 2: winner of predictions

A, B, C, and D made the following predictions before a race:

- A predicted that B would win.
- B predicted that D would be last.
- C predicted that A would be third.
- D predicted that A's prediction would be correct.

Only one of these predictions was true, and this was the prediction made by the winner. In what order did A, B, C, and D finish the race?

Solve an easier related problem

- A and D are equivalent, and thus they can't be winners.
- Since A is wrong, B can't be the winner and D must be the last.
- Based on the above analysis, C must be the winner and the order should be either **CBAD** or **CDAB**
- Since B is wrong, D can be last and the order must be CDAB

Example: baking algorithm

Ingredients (原料)

4 1/2 cups all-purpose flour, 2 teaspoons baking soda
2 cups butter, softened
1 1/2 cups packed brown sugar, 1/2 cup white sugar
2 (3.4 ounce) packages instant vanilla pudding mix
4 eggs, 2 teaspoons vanilla extract
4 cups semisweet chocolate chips
2 cups chopped walnuts (optional)

Procedures

1. Preheat oven to 350 degrees F (175 degrees C).
2. Sift together the flour and baking soda, set aside.
3. In a large bowl, cream together the butter, brown sugar, and white sugar.
4. Beat in the instant pudding mix until blended.
5. Stir in the eggs and vanilla.
6. Blend in the flour mixture.
7. Finally, stir in the chocolate chips and nuts.
8. Drop cookies by rounded spoonfuls onto ungreased cookie sheets.
9. Bake for 12 minutes in the preheated oven, until edges are golden brown.

http://allrecipes.com//Recipe/award-winning-soft-chocolate-chip-cookies/Detail.aspx

Aspects of the baking cookie algorithm

Has a **sequence of steps**

- Each step should be performed in order
- Do step 1, then 2, then 3, etc.

Has a **loop** for baking

- Do: keep cookies in oven
- Until:
  - Cookies have golden brown edges and at least 10 minutes have passed since cookies were placed in the oven
  - This is a condition for ending the loop

High-level operations

- Stirring, mixing, beating, creaming, sifting, baking, etc.
- Requires a some cooking knowledge to understand them

Building blocks of algorithms

Operations (基本操作)

- Individual steps to perform

Sequences structure (顺序结构)

- These are sequences of steps, one after another

Condition and branching (条件和分枝)

- A particular state of the of the world while the algorithm is operating
- Example: “edge of cookies is golden brown”

Iteration (迭代)

- Repeating a sequence of steps until some condition holds

Recursion (递归)

- Repeating a sequence of steps by re-starting the same sequence of steps with new input conditions, repeatedly

Pseudocode primitives

Assignment (赋值)

- *name* ← **expression**
- char Sex = ’f’;

Conditional branching (条件分枝)

- **if** (*condition*) activity A **else** activity B
- if(Sex==‘f’) printf(%c/n, “female”);
  else printf(%c/n, “male”);

Repeated execution (重复执行)

- Iteration
- **while** *condition* **do** activity
- while((count--)!=0) printf(%c/n, Sex);

Procedure (子过程)

- **procedure** *name* (generic names)

Pseudocode for procedure Greetings

Use pseudocode to describe the procedure Greetings for making greetings to the world

```
**procedure** Greetings
        Message← "hello world!";
        Count ← 100;
        while (Count > 0) do
                print the Message;
                Count← Count-1;
```

Operations

Operations are the basic building blocks of algorithms

In the baking algorithm

- Stirring, mixing, beating, creaming, sifting, baking, etc.
- These are relatively high level operations

For computers being instructed via a programming language, operations are simple:

- Assignment
  - ► Giving a variable a value, *Temperature = 49.5*
- Performing an simple computation
  - ► Addition, multiplication, division, log, square root, etc.
  - ► Area = 3.14159 * radius^2
- Moving data, allocating memory, etc.
- Read item of input, produce item of output, etc.

Sequences of operations

A sequence of operations is

- A set of basic operations performed one after another, for which each operation is performed and completed before next one begins.
- Operations need to be performed in the order they are written
  - ► Each operation potentially modifies the state of the situation and may depend on the current state at the start of the operation
  - ► Example: for chocolate chip cookies, adding in the chips changes the batter, by adding the chips
  - ► If this is done too early, some of the dry ingredients will stick to the chips
  - ► The sequence of the operations matters
- This is the von Neumann fetch-execute cycle all over again

Condition and branching (分枝)

Most algorithms change their behavior based on the current state of the situation

- For computers, a change based on the current values of variables inside the computer

A condition takes the form of a comparison

- Oven temperature **is equal to** 350 degrees
- The color of cookie edges **is equal to** golden brown

Conditions are used to determine how an algorithm should branch

- That is, how the sequence of steps to be performed should be changed

Iterative structures

Iteration

- perform a sequence of actions repeatedly until a condition becomes true
- Condition is also known as the termination condition (终止条件)

For loop

- **for** (until *condition*)

  Body

While loop (pretest loop)

- **while** (*condition*)

  Body

Repeat loop (Posttest loop)

- **do** Body

  while *condition*

The while loop structures

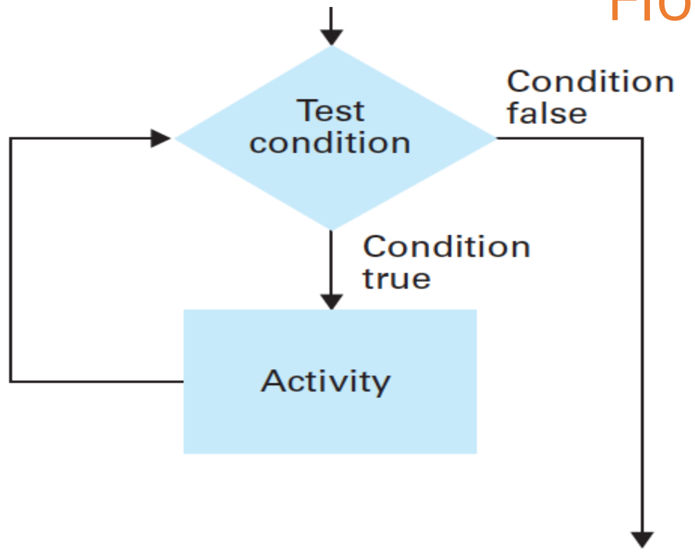
While loop:

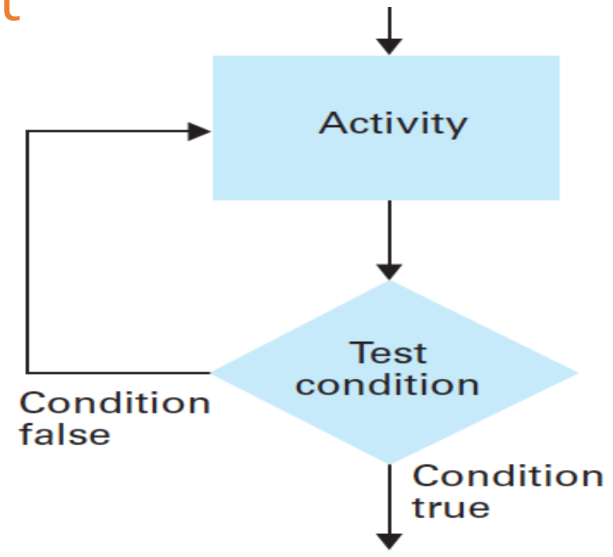
```
while(condition)
    do(Activity)
```

Repeat loop:

```
do(Activity)
    while(condition)
```

Flowchart





Components of loop structures

Formally speaking, it involves three steps in the loop structures

Initialize

- Establish an initial state that will be modified toward the termination condition

Test

- Compare the current state to the termination condition and terminate the repetition if equal

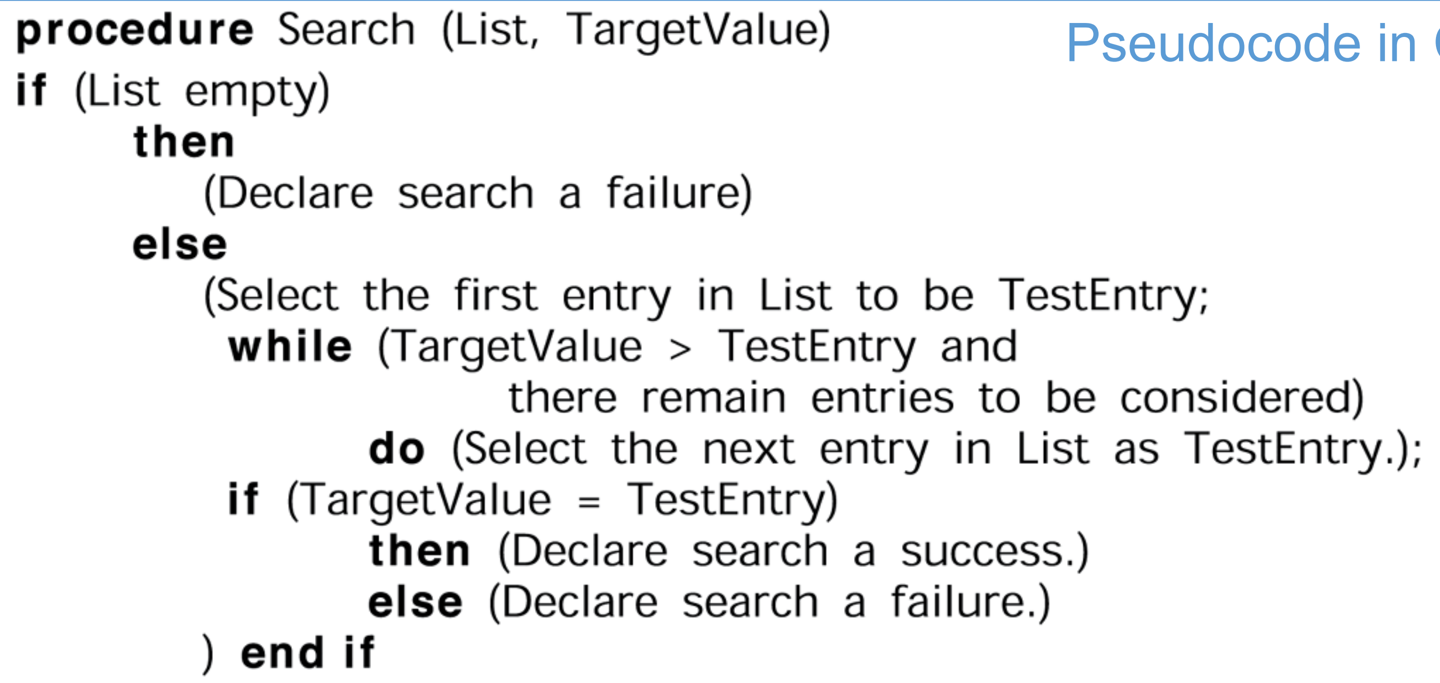
Modify

- Change the state in such a way that it moves toward the termination condition

Sequential search algorithm

Problem setting

- Search a list for the occurrence of a particular target, where the list is keep in alphabetical order(字典序)
- Example: we want to know whether Alex is a student of ME class 2017 or not

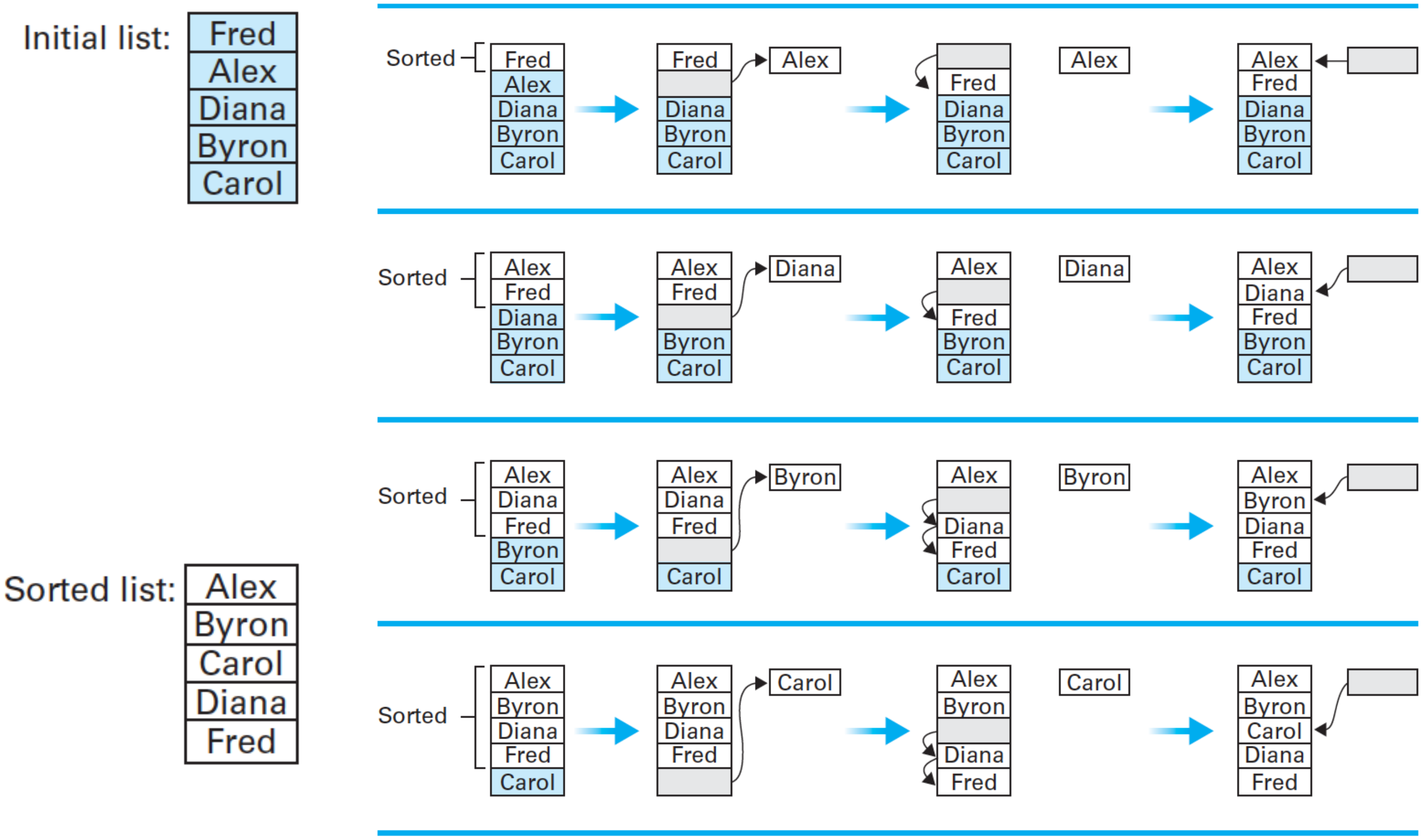


Pseudocode in C style

```
procedure Search (List, TargetValue)
if (List empty)
      then
         (Declare search a failure)
      else
         (Select the first entry in List to be TestEntry;
          while (TargetValue > TestEntry and
                           there remain entries to be considered)
                do (Select the next entry in List as TestEntry.);
          if (TargetValue = TestEntry)
                then (Declare search a success.)
                else (Declare search a failure.)
         ) end if
```

Insertion sort algorithm

Problem: sort the list Fred, Alex, Diana, Byron, and Carol alphabetically



Pseudocode for insertion sort

Pseudocode in Python style

```
def Sort (List)
    N ← 2;
    while (the value of N does not exceed the length of List) do
        (Select the Nth entry in List as the pivot entry;
        Move the pivot entry to a temporary location leaving a
        hole in List;
        while (there is a name above the hole and that name is
                greater than the pivot)
            do (move the name above the hole down into the hole
                leaving a hole above the name)
        Move the pivot entry into the hole in List;
        N ← N + 1
        )
```

Recursive structures (discussed later)

When creating algorithms that operate on trees

- It can be more convenient to describe what the algorithm does at each level
- Then repeat the algorithm at each level

Example:

- Are you related to George Washington?

General approach

- Is_Related_to_Washington(Person P)
    - ► Determine the parents of P. These are Parent1, parent 2, … parent N
- Is any parent George Washington?
    - ► Yes! Person is related, print this out and stop
    - ► No. Repeat check for all parents of parents
- That is:

```
Is_Related_to_Washington(Parent 1)
Is_Related_to_Washington(Parent 2)
…
Is_Related_to_Washington(Parent N)
```

The description of Is_Related_to_Washington depends on itself!

- That is, it is **recursive**

Algorithm efficiency

Algorithm complexity

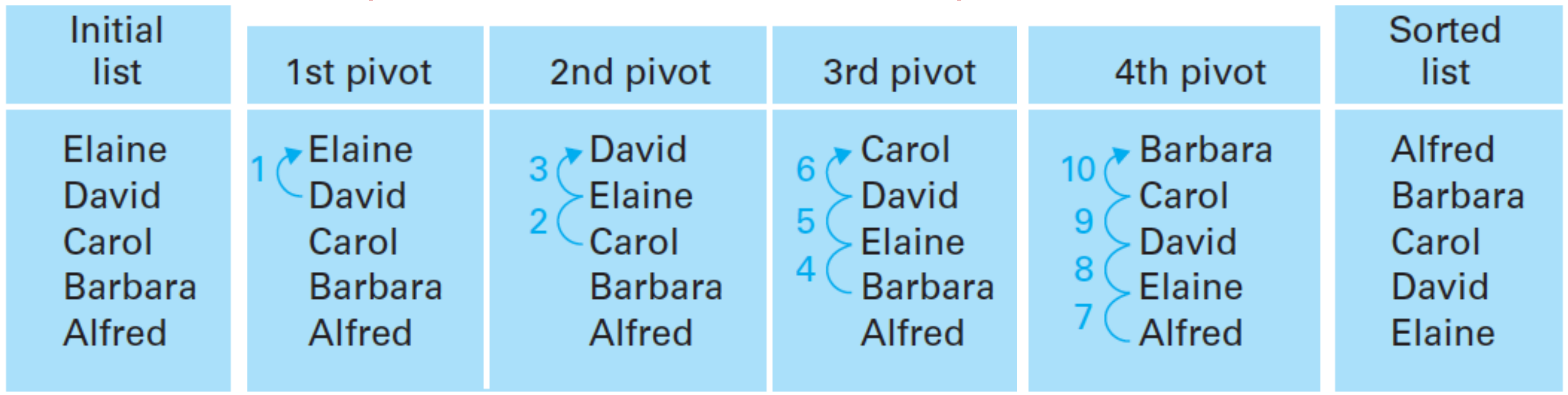
- The number of primitive operations are involved in an algorithm
- Best, worst, and average case analysis
- Big-O notation ***O*** for worst-case analysis
- Big-Omega notation ***Ω*** for the best-case analysis
- Big-theta notation Θ define a range for the complexity

Big-theta notation

- Measured as number of instructions executed
- Used to represent efficiency classes
- Example: Insertion sort is in $\Theta(n^2)$

The worst-case analysis of the insertion sort algorithm

Comparisons made for each pivot



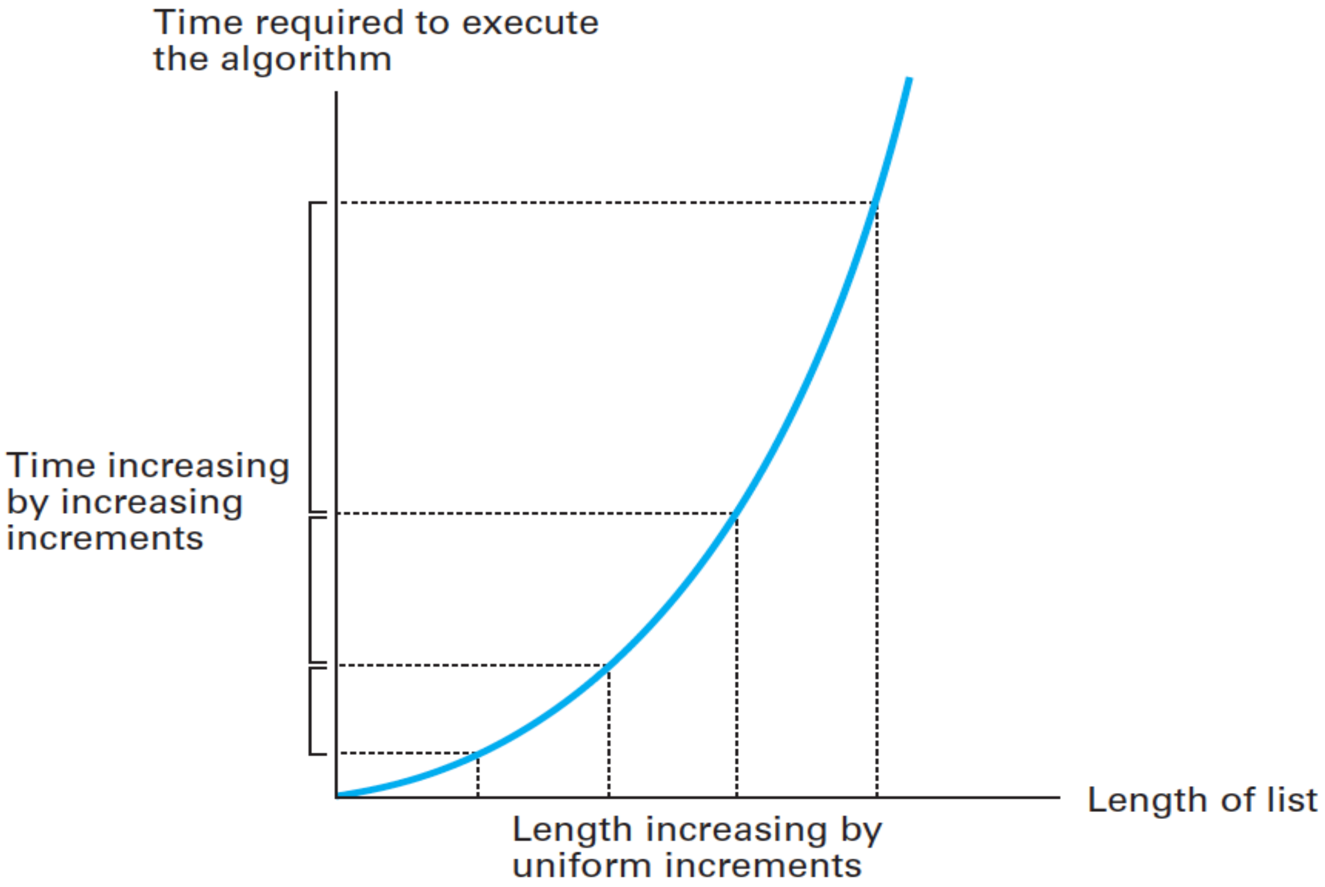
Initial list	1st pivot	2nd pivot	3rd pivot	4th pivot	Sorted list
Elaine	1 Elaine	3 David	6 Carol	10 Barbara	Alfred
David	David	2 Elaine	5 David	9 Carol	Barbara
Carol	Carol	Carol	4 Elaine	8 David	Carol
Barbara	Barbara	Barbara	Barbara	7 Elaine	David
Alfred	Alfred	Alfred	Alfred	Alfred	Elaine

Suppose the length of the list is n

- It involves n-1 pivots
- For the k-th pivot there are k comparisons
- There are 1+2+…+(n-1) = $(n^2-n)/2$ comparisons in total
- $O(n^2)$

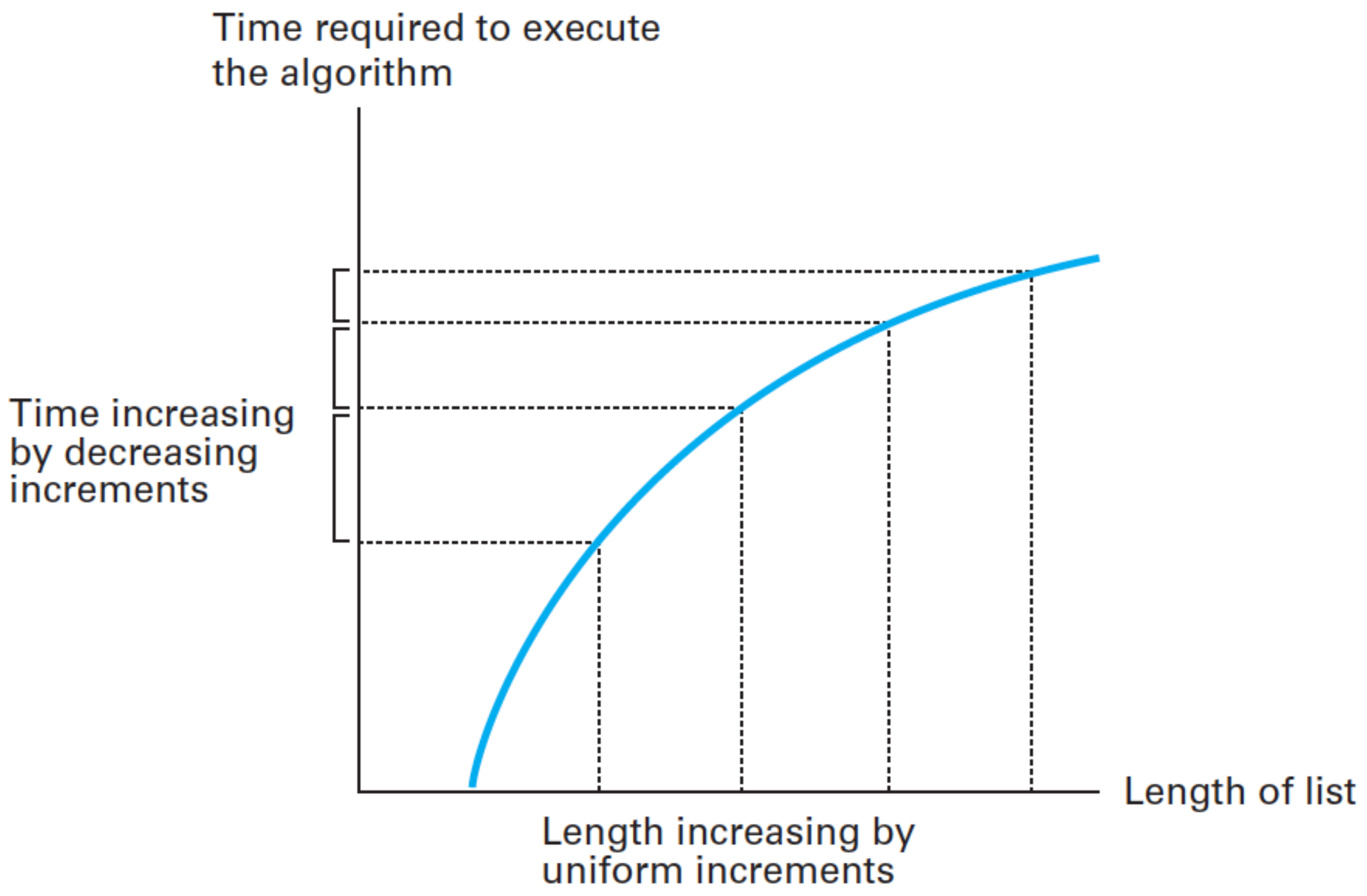
Graph of the worst-case analysis of the insertion sort algorithm

Algorithm complexity is in Θ(n^2)



Graph of the worst-case analysis of the binary search algorithm

Algorithm complexity is in Θ(log n)



Information

Course site

- http://abelgo.cn/cs101.html

Office hours

- You should appoint it in advance
- Email: qingsongg@gmail

Course management system

- All course stuff could be found on Piazza