華南理工大學

South China University of Technology

Intro to Computer Science and Software Engineering

Problem Solving and Algorithms

Dr Yubei Lin
yupilin@scut.edu.cn
School of Software Engineering

How to solve problems

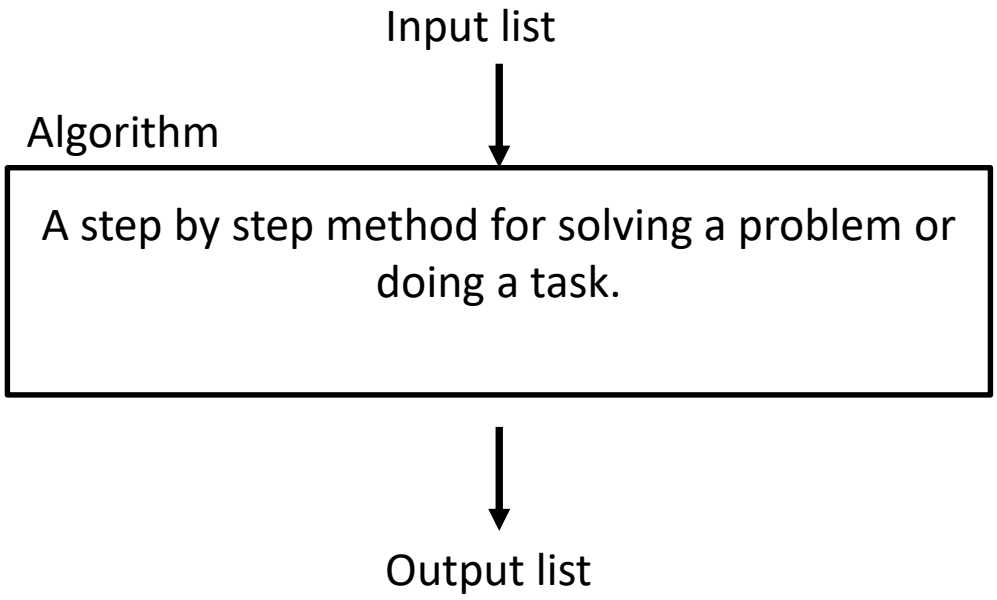
Ask Questions

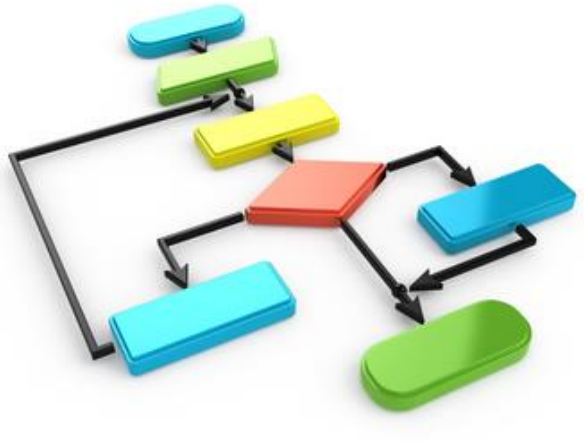
Look for familiar things

Strategies
Divide and Conquer

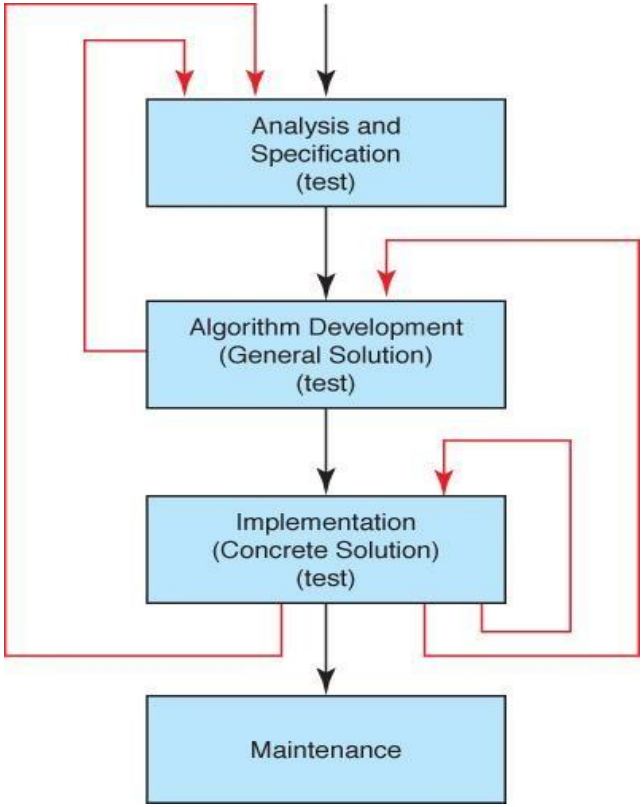
The concept of Algorithm

- Algorithm
  - A step by step method for solving a problem or doing a task.





Computer problem-solving process



Algorithm

A set of unambiguous instructions for solving a problem or subproblem in a finite amount of time using a finite amount of data.

Q: key words?

Abstract Step

An algorithmic step containing unspecified details

Concrete Step

An algorithm step in which all details are specified

The FindLargest example

FindLargest

Input: a list of positive integers

1. Set Largest to 0
2. while (more integers)

   2.1 FindLarger

   End while
3. return Largest

   End

FindLarger

Input: Largest and integer

1. if (integer is greater than Largest)

   then

   1. Set Largest to the value of integer

   end if

End

What about the Abstract Step and Concrete Step?

Three constructs

- Sequence, Decision and Repetition (loop)

do action 1
do action 2
.....

Sequence

If a condition is true,
Then

do a series of actions

Else

do another series of action

Decision

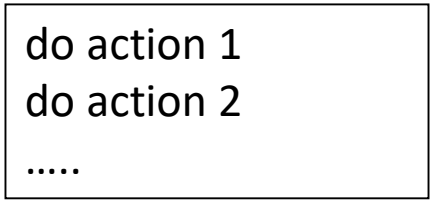
While a condition is true,

do action 1
do action 2
…. ….

Repetition

Variables in constructs

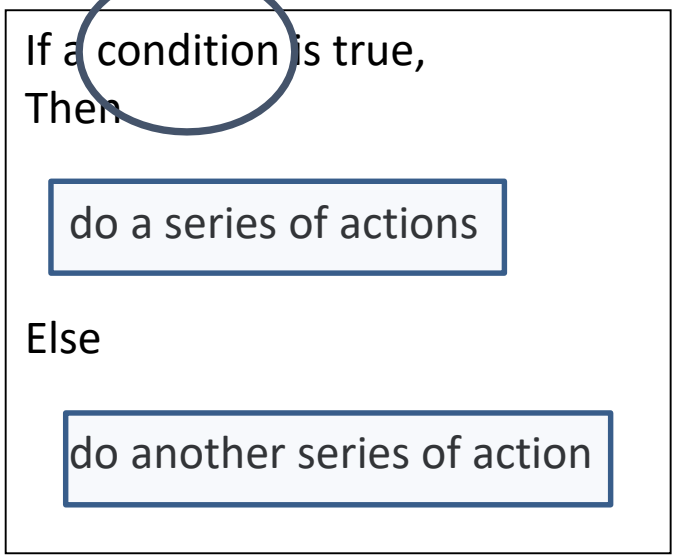
- Sequence, Decision and Repetition (loop)



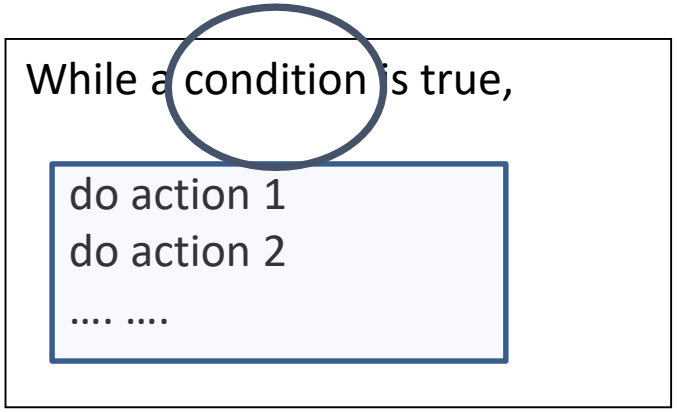
Sequence

A variable must be used to store the value of current condition!

Simple variable is enough, e.g. int count = 0;



Decision



Repetition

Loop constructions

count-controlled loop

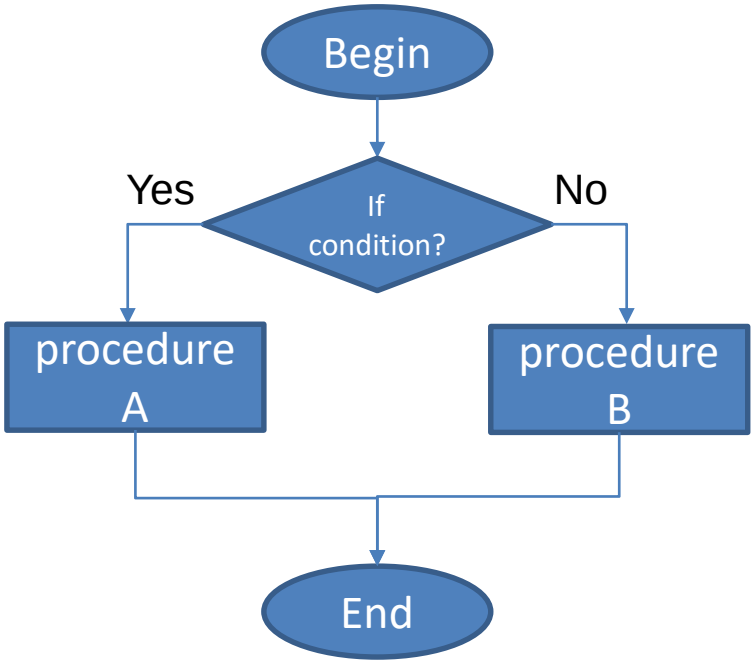
```
Set sum to 0
Set count to 1
While (count <= limit)
        Read number
        Set sum to sum + number
        Increment count
Write "Sum is " + sum
```

event-controlled loop

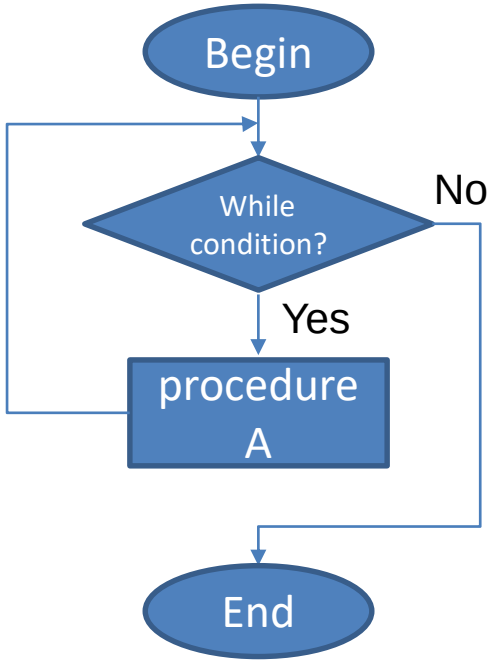
```
Set sum to 0
Set allPositive to true
WHILE (allPositive)
        Read number
        IF (number > 0)
                Set sum to sum + number
        ELSE
                Set allPositive to false  Write
"Sum is " + sum
```

Algorithm representation

- Flowchart
  - a pictorial representation showing how the algorithm flows from beginning to end.



Decision construction flowchart



Loop construction flowchart

Algorithm representation

- Pseudocode
  - an Englishlike representation

FindLargest
Input: a list of positive integers
1. Set Largest to 0
2. while (more integers)
    2.1 if (the integer is greater than Largest)
        then
            2.1.1 Set Largest to the value of the integer
    End while
3. return Largest
    End

More formal definition

- Algorithm
  - an ordered set of unambiguous steps that produces a result and terminates in a finite time.

Subalgorithm

- An algorithm can be divided into small units called subalgorithm;
- Each subalgorithm is in turn divided into smaller subalgrithms.
- This process continues until the subalgorithm become intrinsic (understood immediately)

Subalgorithm

FindLargest
Input: a list of positive integers
1. Set Largest to 0
2. while (more integers)

   2.1 FindLarger

   End while
3. return Largest
   End

FindLarger
Input: Largest and integer
1. if (integer is greater than Largest)
   then
      1.1 Set Largest to the value of integer
   end if
End

Basic Algorithm

- Summation: of a list of numbers
- Smallest and Largest
  - Try the FindSmallest

Basic Algorithm

- Sorting https://visualgo.net/en/sorting
  - Selection sort
  - Bubble sort
  - Insertion sort
- Other sorting algorithm
  - Quick sort, heap sort, shell sort ….
- Which one is betters, it depends on … …
- Complexity of algorithm!!!

Selection sort

算法思想：选择排序，从头至尾扫描序列，找出最小的一个元素，和第一个元素交换，接着从剩下的元素中继续这种选择和交换方式，最终得到一个有序序列。

算法过程：原始序列：43，65，4，23，6，98，2，65，7，79

第一趟： 2，65，4，23，6，98，43，65，7，79

第二趟： 2，4，65，23，6，98，43，65，7，79

第三趟： 2，4，6，23，65，98，43，65，7，79

第四趟： 2，4，6，7，43，65，98，65，23，79

第五趟： 2，4，6，7，23，65，98，65，43，79

第六趟： 2，4，6，7，23，43，98，65，65，79

第七趟： 2，4，6，7，23，43，65，98，65，79

第八趟： 2，4，6，7，23，43，65，65，98，79

第九趟： 2，4，6，7，23，43，65，65，79，98

选择排序的时间复杂度为：O(n^2)，空间复杂度：O(1)

Bubble sort

算法思想：首先将第一个记录的关键字和第二个记录的关键字进行比较，若为逆序，则将两个记录交换之，然后比较第二个记录的关键字和第三个记录的关键字，依次类推，直至第n-1个记录和第n个记录的关键字进行过比较为止。

算法过程：原始序列： 43, 65, 4, 23, 6, 98, 2, 65, 7, 79

第一趟： 43, 4，23，6，65，2，65，7，79，98

第二趟： 4，23，6，43，2，65，7，65，79，98

第三趟： 4，6，23，2，43，7，65，65，79，98

第四趟： 4，6，2，23，7，43，65，65，79，98

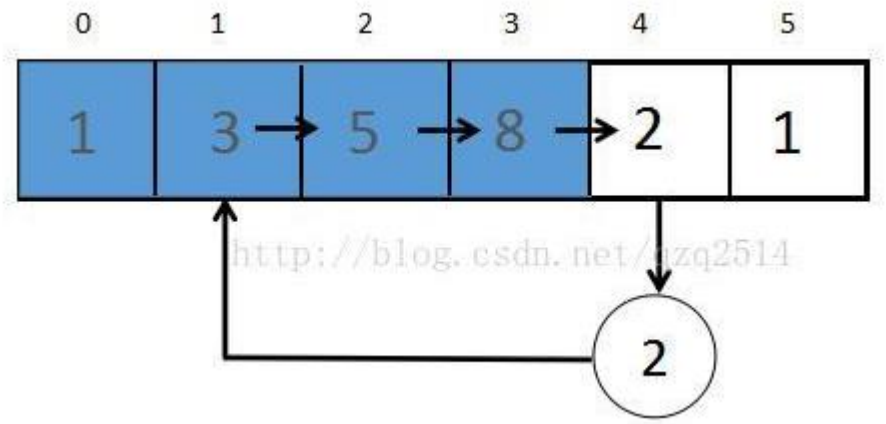
第五趟： 4，2，6，7，23，43，65，65，79，98

第六趟： 2，4，6，7，23，43，65，65，79，98

时间复杂度为：O(n^2)，空间复杂度为O(1)

Insertion sort

算法思想：每步把待排序的记录按其关键码值的大小逐个插入到一个前面已经排好序的有序序列中，直到所有的记录插入完为止，得到一个新的有序序列。

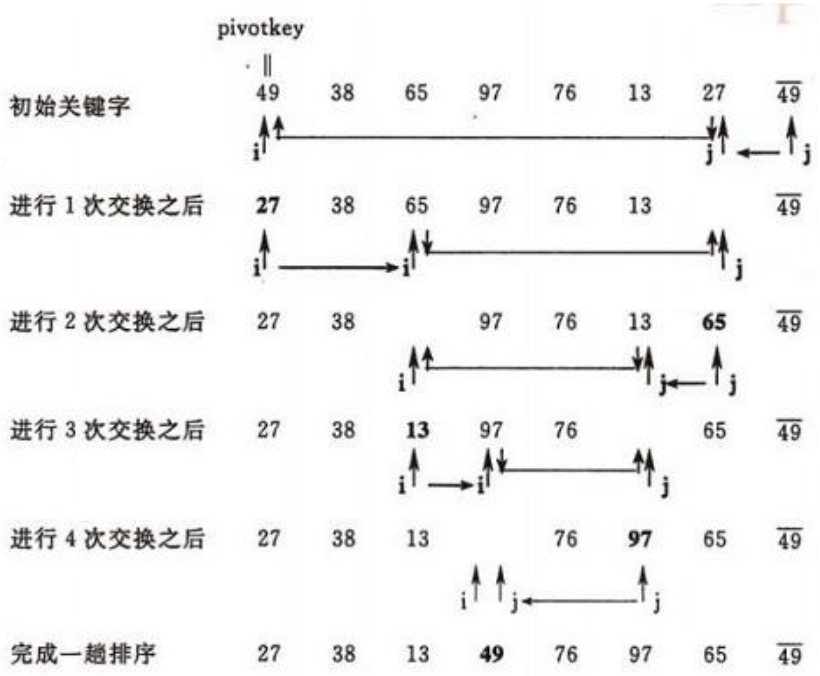


时间复杂度为：O(n^2)，空间复杂度为O(1)

Quick sort

算法思想：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

(a) 一趟排序的过程：



(b) 排序的全过程

初始状态	{49	38	65	97	76	13	27	$\overline{49}$}
一次划分之后	{27	38	13}	**49**	{76	97	65	$\overline{49}$}
分别进行快速排序	{13}	**27**	{38}					
	结束		结束		{49	65}	**76**	{97}
					49	{65}		结束
						结束		
有序序列	{13	27	38	49	$\overline{49}$	65	76	97}

时间复杂度为：$O(n\log_2 n)$

空间复杂度为：$O(\log_2 n)$

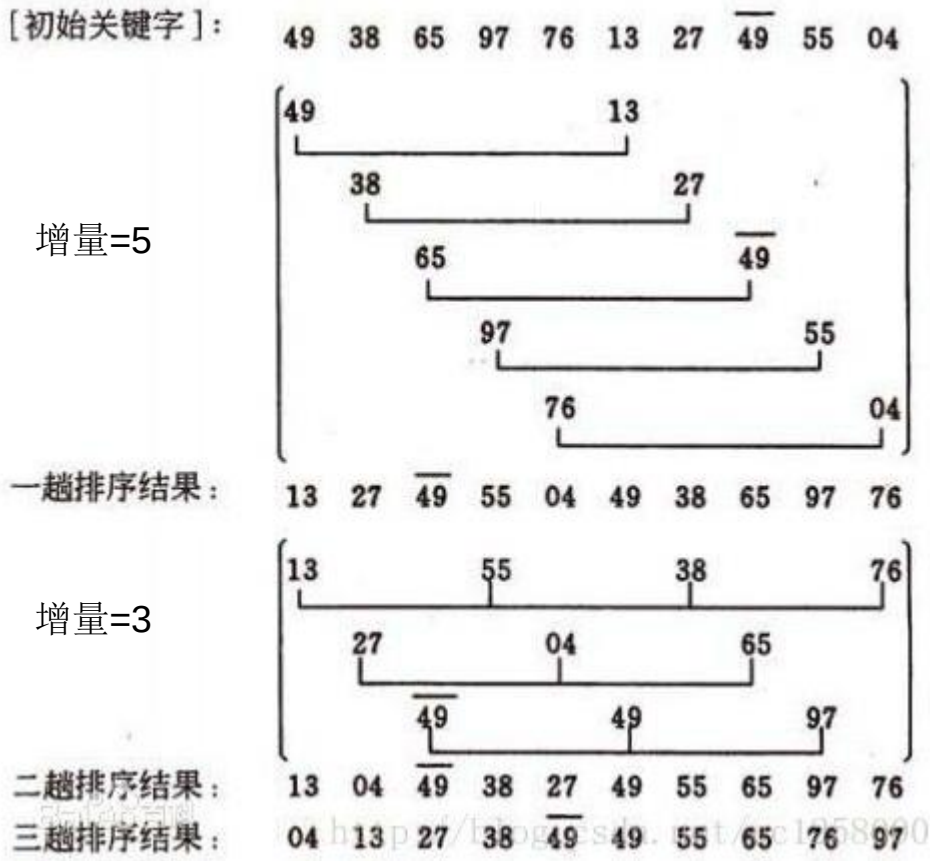
所有图片均来自网络

Shell sort

算法思想：又名缩小增量排序，本质是插入排序，只不过是将待排序的序列按某种规则分成几个子序列，分别对几个子序列进行直接插入排序。这个规则就是增量，增量选取很重要，增量一般选序列长度一半，然后逐半递减，直到最后一个增量为1，为1相当于直接插入排序。

时间复杂度为：O(n^1.25)

空间复杂度为：O(1)



所有图片均来自网络

Basic Algorithm

- Searching
  - Sequential searching (unordered list)
  - Binary searching (ordered list)

Composite Data Types

Simple variables: atomic, only one place to store a value

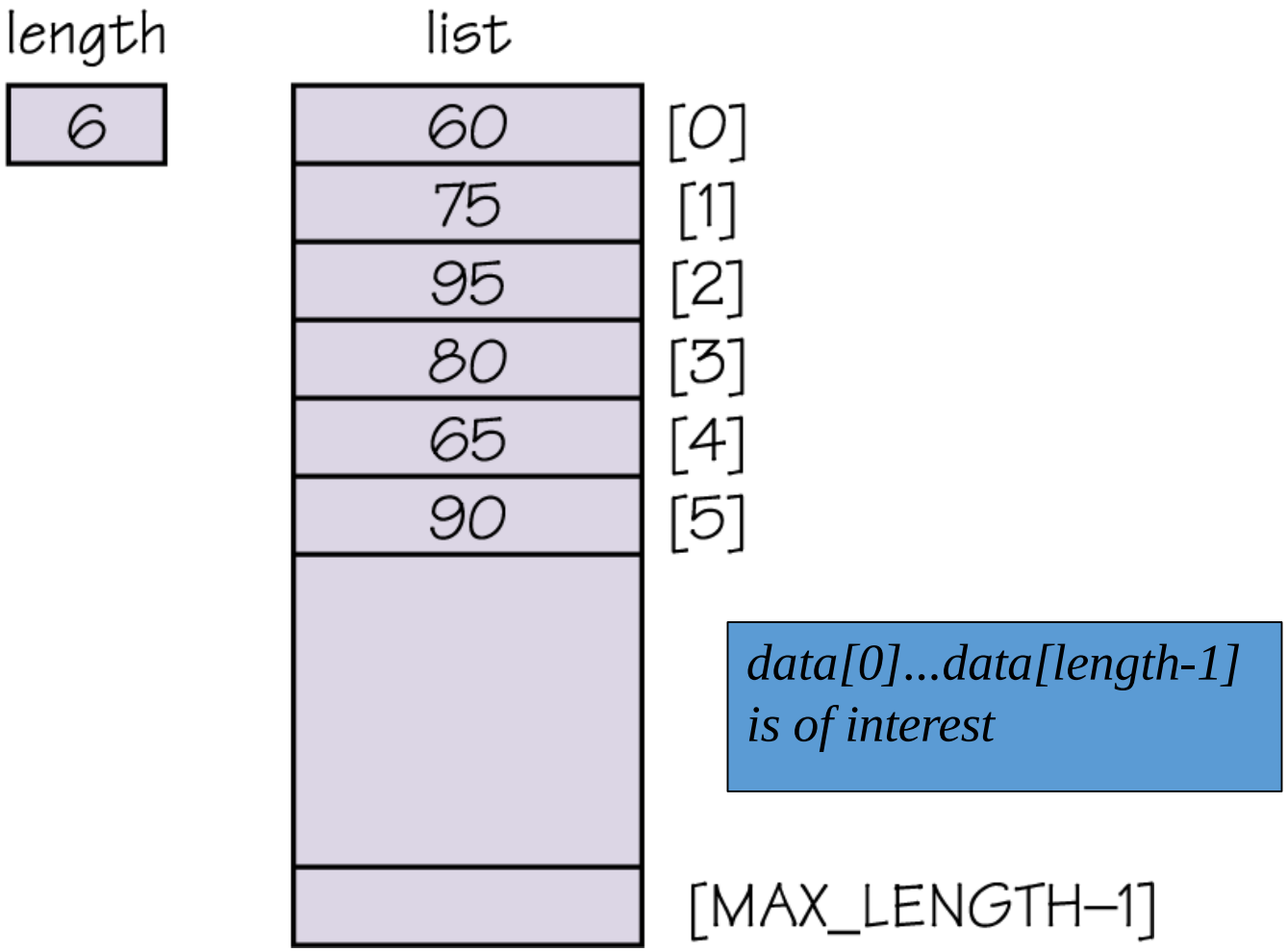
Records

A named heterogeneous collection of items in which individual items are accessed by name. (P. 212)

Arrays

A named homogeneous collection of items in which an individual item is accessed by its position (index) within the collection. (P. 210)

Unsorted array



Sequential Search of an Unsorted Array

A sequential search examines each item in turn and compares it to the one we are searching.

If it matches, we have found the item. If not, we look at the next item in the array.

We stop either when we have found the item or when we have looked at all the items and not found a match.

Thus, a loop with two ending conditions.

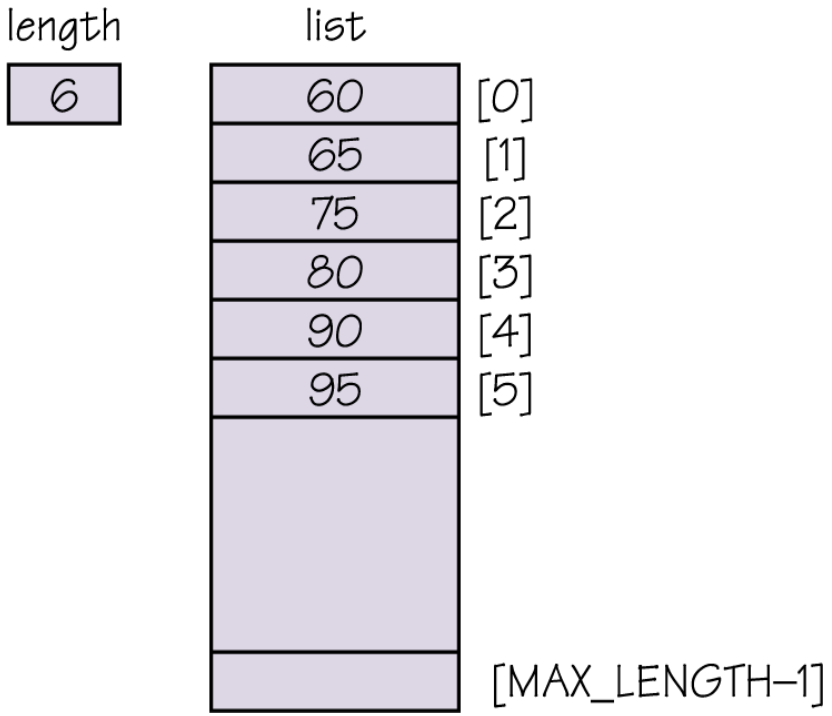
P. 212

Sorted array

The values stored in an array have unique keys of a type for which the relational operators are defined.

Sorting rearranges the elements into either ascending or descending order within the array.

A sorted array is one in which the elements are in order.



A sorted array of integers

Sequential Search in a Sorted Array

If items in an array are sorted, we can stop looking when we pass the place where the item would be it were present in the array.

Is it better?

Binary Search

Binary search (list must be sorted)
Search begins at the middle and finds the item or eliminates half of the unexamined items; process is repeated on the half where the item might be.

Binary Search

Binary search (list must be sorted)

Search begins at the middle and finds the item or eliminates half of the unexamined items; process is repeated on the half where the item might be.

[0]	ant
[1]	cat
[2]	chicken
[3]	cow
[4]	deer
[5]	dog
[6]	fish
[7]	goat
[8]	horse
[9]	camel
[10]	snake

Searching for cat

First	Last	Middle	Comparison	
0	10	5	cat < dog	
0	4	2	cat < chicken	
0	1	0	cat < ant	
1	1	1	cat = cat	**Return: true**

Searching for fish

First	Last	Middle	Comparison	
0	10	5	fish > dog	
6	10	8	fish < horse	
6	7	6	fish = fish	**Return: true**

Searching for zebra

First	Last	Middle	Comparison	
0	10	5	zebra > dog	
6	10	8	zebra > horse	
9	10	9	zebra > rat	
10	10	10	zebra > snake	
11	10		first > last	**Return: false**

Review

- Algorithm
  - an ordered set of unambiguous steps that produces a result and terminates in a finite time.
- Basic algorithm
  - Sorting
  - Searching

Two approaches

- Two approaches to write algorithm for solving a problem
  - Iterations (迭代)
  - Recursion (递归): a process in which an algorithm called itself.

Factorial example（阶乘）

- Iterative solution

Factorial
Input: a positive integer num
1. Set FactN to 1
2. Set i to 1
3. while (i less than or equal to num)
    3.1 Set FactN to FactN * i
    3.2 Increment i
   End while
3. return FactN
   End

Factorial example

- Recursion solution

Factorial
Input: a positive integer num
1. if (num is equal to 0)
then
    1.1 return 1
else
    1.2 return num * Factorial (num-1)
End if
End

Iteration vs. Recursion

Iteration
within program

Subprogram
Call another subprogram

Recursion
Call the program itself

Parameter passing
Value or Reference

SWAP(item1, item2)

Set temp to item1
Set item2 to item1
Set item1 to temp

SWAP(data1, data2)