Chapter 8

Algorithms

Objectives 學習目標

After studying this chapter, students should be able to:

- Define an algorithm and relate it to problem solving. 定義演算法
- Define three constructs and describe their use in algorithms. 三種結構使用於演算法
- Describe UML diagrams and pseudocode and how they are used in algorithms. 演算法表示：流程圖與虛擬碼
- List basic algorithms and their applications.
- Describe the concept of sorting and understand the mechanisms behind three primitive sorting algorithms. 排序
- Describe the concept of searching and understand the mechanisms behind two common searching algorithms. 搜尋
- Define subalgorithms and their relations to algorithms. 副程式
- Distinguish between iterative 遍循 and recursive 遞迴 algorithms

Source: Foundations of Computer Science © Cengage Learning
Generalization 一般化

Is it possible to generalize the algorithm? We want to find the largest of \( n \) positive integers, where \( n \) can be 1000, 1,000,000, or more. Of course, we can follow the above method and repeat each step. But if we change the algorithm to a program, then we need to actually type the actions for \( n \) steps!

There is a better way to do this. We can tell the computer to initial the state and repeat the steps \( n \) times. We now include this feature in our pictorial algorithm.

**Generalization of FindLargest**

\[
	ext{FindLargest} \quad \text{// Given: List[0] ~ List[n-1]} \\
\{ \\
\quad \text{Largest} = 99999; \quad \text{初始狀態} \\
\quad \text{for} (i = 0 \text{ to } n-1) \quad \text{重複n次} \\
\quad \{ \\
\quad \quad \text{if} (\text{List} [i] > \text{Largest}) \quad \text{Largest} = \text{List}[i]; \\
\quad \} \\
\quad \text{return} \ \text{Largest}; \\
\}
\]
Now that we have discussed the concept of an algorithm and shown its representation, here is a more formal definition:

**An algorithm is an ordered set of unambiguous steps that produces a result and terminates in a finite time.** 一演算法由一组有序而非模棱两可的步骤组成，能在有限时间内完成，並產生正確的結果

**Sorting Algorithms 排序演算法**

One of the most common applications in computer science is sorting, which is the process by which data is arranged according to its values  將資料依照其值而排列. People are surrounded by data. If the data was not ordered, it would take hours and hours to find a single piece of information. 如資料為排序，尋找某一資料很困難. Imagine the difficulty of finding someone’s telephone number in a telephone book that is not ordered.

In this section, we introduce **three basic sorting algorithms** 三個基本排序方法: selection sort選擇排序法, bubble sort氣泡排序法 and insertion sort插入排序法. These three sorting algorithms are the foundation of faster sorting algorithms used in computer science today.
SelectionSort (List[n]) // Given: List[0] ~ List[n-1]
{
    for (i = 0 to n-1)
    {
        smallest = List[i];  k_min = i;
        for (j = i+1 to n-1)
        {
            if (List[j] < smallest)
            {
                smallest = List[j];  k_min = j;
            }
        }
        temp = List[i];
        List[i] = List[k_min];  List[k_min] = temp;
    }
}

Searching Algorithms 搜尋演算法

Another common algorithm in computer science is searching, which is the process of finding the location of a target among a list of objects. In the case of a list, searching means that given a value, we want to find the location of the first element in the list that contains that value. 搜尋乃找一筆資料位在哪裡

There are two basic searches for lists: sequential search 順序搜尋 and binary search 二元搜尋. Sequential search can be used to locate an item in any list, whereas binary search requires the list first to be sorted.
Binary Search for a given sorted 1D-array A[19]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>19</td>
<td>27</td>
<td>33</td>
<td>38</td>
<td>42</td>
<td>47</td>
<td>54</td>
<td>62</td>
<td>65</td>
<td>69</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>81</td>
<td>85</td>
<td>91</td>
<td>97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Sequential Search** (A[n], 91)

```plaintext
{   count = 0;
    for (i=0 to n-1)
    {   count = count + 1;
        if (A[i]==91)  return count;
    }
}
```

**Binary Search** (A[n], 91)

```plaintext
{   low=0; high=n-1; count=0;
    while (low <= high)
    {   mid = (low+high) / 2; count = count+1;
        if (A[i]==91)  return count;
        else if (A[i]<=91)  low = mid+1;
            else                 high = mid-1;
    }
}
```

Recursive Solution to Factorial Problem

The recursive solution does not need a loop, as the recursion concept itself involves repetition.

**Algorithm 8.7** Pseudocode for recursive solution of factorial problem:

```
Algorithm: Factorial (n)

Purpose: Find the factorial of a number using recursion

Pre: Given: n
Post: None
Return: n!

{   Factorial (n)
    {   if (n == 0)  return (1);
        else             return (n * Factorial (n-1));
    }
}

if (n = 0)     return 1
else             return n * Factorial (n-1)
```
Review Questions

➢ What is the **formal definition** of **algorithm**?

➢ Please show **two kinds of algorithm representation**.

➢ Please list the detailed steps of **selection sort** for the numbers, 56, 78, 25, 39, 99, 67, and 15.

➢ Please list the detailed steps of **bubble sort** for the numbers, 56, 78, 25, 39, 99, 67, and 15.

➢ Please list the detailed steps of **insertion sort** for the numbers, 56, 78, 25, 39, 99, 67, and 15.

➢ Please give the **algorithm** of **binary search**.

➢ Please give the **iterative algorithm** of **n factorial**.

➢ Please give the **recursive algorithm** of **n factorial**.