Basic Concepts

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 6, 2012

What We Have Done (Chapter 1 and Supplementary Materials)

- C++ (in class): pointers, references, template, STL
- C++ (in reading): everything else in Chapter 1
- DSA (in class): historical notes, programming vs coding, “definition”, motivation

What We Will Do

- more about algorithms (supplementary)
- arrays
- linked lists
- very difficult homework 2 (2141461162 bytes of data, 1.5 times more than last year)

Five Criteria of 食譜

阿基師的蕃茄炒蛋食譜

蕃茄3顆、蛋3顆、蔥2支、薑1小塊、太白粉、鹽、糖

1. 切蔥花備用；薑末備用。
2. 蕃茄去蒂畫十字刀，下鍋汆燙。
3. 撈起蕃茄，放入冷水中除外皮。
4. 蛋液打勻，加入少許鹽巴；將蕃茄切適當大小。
5. 起油鍋，爆香薑末，加入蕃茄，倒入適量的水，加入少許鹽、糖。
6. 加入少許太白粉勾芡。
7. 加入蛋液，輕輕翻炒。
8. 起鍋前撒上蔥花。

- Input: 食材
- Output: 菜色
- Definiteness: 清楚的步驟
- Finiteness: 一定可以做完
- Effectiveness: 煮菜的人做得到

Five Criteria of Algorithm

SMALLEST-NUM-INDEX-FINDING
(integer array *list*, integer size *n*)

1. set *min* to 0
2. set *i* as $1, 2, \cdots, n-1$
   - if *list*[*i*] is smaller than *list*[*min*], then set *min* as *i*
3. return *min*

- Input: external supplies
- Output: desired output
- Definiteness: clear steps
- Finiteness: will terminate
- Effectiveness: can be done by computers

~~Insertion~~ Sort

Selection

SEL-SORT(integer array *list*, integer size *n*)
outputs an in-place sorted list

- for i from 0 to $n-1$
  1. let *min* be the index of the smallest number from $list[i]$ to $list[n-1]$
  2. interchange $list[i]$ and $list[min]$

- Input list n
- Output none
- Definiteness
- Finiteness
- Effectiveness

```
unsigned int i;
for(i=10;i>=0;i--) { printf("%d", i); }
```

- step one: can be done by the computer with a simple modification of SMALLEST-NUM-INDEX-FINDING
- step two: can be done by the computer easily (How?)

Correctness of Selection Sort

SEL-SORT(integer array *list*, integer size *n*)
outputs an in-place sorted list

Given: integer array *list* with integer size n

- for i from 0 to $n-1$
  1. let *min* be the index of the smallest num. from $list[i]$ to $list[n-1]$
  2. interchange $list[i]$ and $list[min]$

Theorem

After the loop of $i = q$, for any $j > q$,

$$arr[0] \leq arr[1] \leq \cdots \leq arr[q] \leq arr[j].$$

Proof by Mathematical Induction:

- When $i = 0$, statement true (why?).
- Assume statement true when $i = t$;
  then when $i = t + 1$, (what happens?)

will see more about sorting and other algorithms in this class

Basic Algorithms: Sequential and Binary Search

- Input: a **sorted** integer array *list* with size *n*, an integer *searchnum*
- Output: if *searchnum* is within *list*, its index; otherwise −1

SEQ-SEARCH $(list, n, searchnum)$

```
for i ← 0 to n − 1 do
  if list[i] == searchnum
    return i
  end if
end for
return −1
```

BIN-SEARCH $(list, n, searchnum)$

```
left ← 0, right ← n − 1
while left ≤ right do
  middle ← floor((left + right)/2)
  if list[middle] > searchnum
    right ← middle − 1
  else if list[middle] < searchnum
    left ← middle + 1
  else        /* list[middle] == searchnum */
    return middle
  end if
end while
return −1
```

Sequential Search: Eliminate One Element Each Time

SEQ-SEARCH(*list*, *n*, *searchnum*)

for $i \leftarrow 0$ to $n - 1$ **do**
  if $list[i] == searchnum$
    return i
  end if
end for
return -1

$l[0]$	$l[1]$	$l[2]$	$l[3]$	$l[4]$	$l[5]$	$l[6]$
1	3	4	9	9	10	13

- search for 9
- search for 15 (worst case?)

Binary Search: Eliminate at Least Half Each Time

BIN-SEARCH(*list*, *n*, *searchnum*)

```
left ← 0, right ← n − 1
while left ≤ right do
    middle ← floor((left + right)/2)
    if list[middle] > searchnum
        right ← middle − 1
    else if
    list[middle] < searchnum
        left ← middle + 1
    else
        return middle
    end if
end while
return −1
```

$l[0]$	$l[1]$	$l[2]$	$l[3]$	$l[4]$	$l[5]$	$l[6]$
1	3	4	9	9	10	13

- search for 9

1 step, done

3 steps

- search for 15 (worst case?)

15 > 9   L ← M+1

10 > 9   L ← M+1

• search for 3   2 steps

3 < 9   3 = 3