Data Structures

(Feodor F. Dragan)
Department of Mathematics and Computer Science
Kent State University

Fall, 2000

CHAPTER 6
Algorithm Analysis

Contents

• What is Algorithm Analysis?
• A Few Examples
• Maximum Contiguous Subsequence Sum
• General Big-Oh Rules
• The Logarithm
• Static Searching

What is Algorithm Analysis?

• Correctness

• Complexity → time, space

1. worst case
2. best case
3. average case

• How to estimate running time for an algorithm?
• What techniques drastically reduce running time?
• What mathematical framework can be used?

Sorting

- Rearrange a[0], a[1], …, a[n-1] into ascending order. When done, a[0] <= a[1] <= … <= a[n-1]
- 8, 6, 9, 4, 3 => 3, 4, 6, 8, 9

Sort Methods

- Insertion Sort
- Bubble Sort
- Selection Sort
- Count Sort
- Shaker Sort
- Shell Sort
- Heap Sort
- Merge Sort
- Quick Sort

Insert An Element

- Given a sorted list/sequence, insert a new element
- Given 3, 6, 9, 14
- Insert 5
- Result 3, 5, 6, 9, 14

Insert an Element

- 3, 6, 9, 14 insert 5
- Compare new element (5) and last one (14)
- Shift 14 right to get 3, 6, 9, , 14
- Shift 9 right to get 3, 6, , 9, 14
- Shift 6 right to get 3, , 6, 9, 14
- Insert 5 to get 3, 5, 6, 9, 14

Insertion Sort

- Start with a sequence of size 1
- Repeatedly insert remaining elements

Insertion Sort

- Sort 7, 3, 5, 6, 1
- Start with 7 and insert 3 => 3, 7
- Insert 5 => 3, 5, 7
- Insert 6 => 3, 5, 6, 7
- Insert 1 => 1, 3, 5, 6, 7

Insertion Sort

```
for (int i = 1; i < a.length( ); i++)
{// insert a[i] into a[0:i-1]
   // code to insert comes here
}
```


Insertion Sort

```
for (int i = 1; i < a.length( ); i++)
{// insert a[i] into a[0:i-1]
  int t = a[i];
  int j;
  for (j = i - 1; j >= 0 && t < a[j]; j--)
     a[j + 1] = a[j];
  a[j + 1] = t;
}
```


Complexity

- Space/Memory
- Time
- Count a particular operation
- Count number of steps
- Asymptotic complexity

Comparison Count

```
for (int i = 1; i < a.length( ); i++)
{// insert a[i] into a[0:i-1]
  int t = a[i];
  int j;
  for (j = i - 1; j >= 0 && t < a[j]; j--)
      a[j + 1] = a[j];
  a[j + 1] = t;
}
```


Comparison Count

- Pick an instance characteristic … n, n = a.length( ) for insertion sort
- Determine count as a function of this instance characteristic.

Comparison Count

```
for (j = i - 1; j >= 0 && t < a[j]; j--)
    a[j + 1] = a[j];
```

How many comparisons are made?

Comparison Count

```
for (j = i - 1; j >= 0 && t < a[j]; j--)
    a[j + 1] = a[j];
```

number of compares depends on
a[]s and t as well as on n

Comparison Count

- ➔ Worst case count = maximum count
- ➔ Best case count = minimum count
- ➔ Average count

Worst-Case Comparison Count

for (j = i - 1; j >= 0 && **t < a[j];** j--)
    a[j + 1] = a[j];

a = [1, 2, 3, 4] and t = 0 => 4 compares
a = [1,2,3,…,i] and t = 0 => i compares

Worst-Case Comparison Count

for (int i = 1; i < n; i++)
for (j = i - 1; j >= 0 && **t < a[j];** j--)
    a[j + 1] = a[j];

total compares = 1 + 2 + 3 + … + (n-1)
                        = (n-1)n/2

Asymptotic Complexity of Insertion Sort

- O(n^2)
- What does this mean?
  - Time or number of operations does not exceed c.n^2 on any input of size n

Complexity of Insertion Sort

- Is O(n^2) too much time?
- Is the algorithm practical?

Practical Complexities

10^9 instructions/second

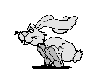
1000	1mic	10mic	1milli	1sec
10000	10mic	130mic	100milli	17min
10^6	1milli	20milli	17min	32years

Impractical Complexities

10^9 instructions/second

n	n^4	n^{10}	2^n
1000	17min	3.2×10^{13} years	3.2×10^{283} years
10000	116 days	???	???
10^6	3 x 10^7 years	??????	??????

Faster Computer Vs Better Algorithm



Algorithmic improvement more useful than hardware improvement.

E.g. 2^n to n^3

More Examples

- Finding the minimum element in an array
  - Sequential scan ➔ O(N)
- Finding the closest points in the plane
  - distance between each pair, how many pairs? N(N-1)/2➔O(N^2)
- Finding Colinear points in the plane
  - Useful in GUIs
  - Brute force solution, how many triples? N(N-1)(N-2)/6➔O(N^3)
  - A clever solution takes only O(N^2)