NPTEL MOOC,JAN-FEB 2015
Week 2, Module 4

DESIGN AND ANALYSIS OF ALGORITHMS

Insertion Sort

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Sorting

- Searching for a value
  - Unsorted array — linear scan, O(n)
  - Sorted array — binary search, O(log n)
- Other advantages of sorting
  - Finding median value: midpoint of sorted list
  - Checking for duplicates
  - Building a frequency table of values

How to sort?

- You are a Teaching Assistant for a course
- The instructor gives you a stack of exam answer papers with marks, ordered randomly
- Your task is to arrange them in descending order

Strategy 2

- First paper: put in a new stack
- Second paper:
  - Lower marks than first? Place below first paper
    Higher marks than first? Place above first paper
- Third paper
  - Insert into the correct position with respect to first two papers
- Do this for each subsequent paper:
  insert into correct position in new sorted stack

Strategy 2 …

74 32 89 55 21 64

Strategy 2 …

~~74~~ 32 89 55 21 64

74

Strategy 2 …

~~74~~ ~~32~~ 89 55 21 64

32 74

Strategy 2 …

~~74~~ ~~32~~ ~~89~~ 55 21 64

32 74 89

Strategy 2 …

~~74~~ ~~32~~ ~~89~~ ~~55~~ 21 64

32 55 74 89

Strategy 2 …

~~74~~ ~~32~~ ~~89~~ ~~55~~ ~~21~~ 64

21 32 55 74 89

Strategy 2 …

~~74~~ ~~32~~ ~~89~~ ~~55~~ ~~21~~ ~~64~~

21 32 55 64 74 89

Strategy 2 …

Insertion Sort

- Start building a sorted sequence with one element
- Pick up next unsorted element and insert it into its correct place in the already sorted sequence

Insertion Sort

```
InsertionSort(A,n) // Sort A of size n

for (pos = 1; pos < n; pos++)
    // Build longer and longer sorted segments
    // In each iteration A[0]..A[pos-1] is already sorted

    // Move first element after sorted segment left
    // till it is in the correct place
    nextpos = pos
    while (nextpos > 0 &&
              A[nextpos] < A[nextpos-1])
        swap(A,nextpos,nextpos-1)
        nextpos = nextpos-1
```

Insertion Sort

74 32 89 55 21 64

Insertion Sort

74 32 89 55 21 64

Insertion Sort

32 74 89 55 21 64

Insertion Sort

32 74 89 55 21 64

Insertion Sort

32 74 55 89 21 64

Insertion Sort

32 55 74 89 21 64

Insertion Sort

32 55 74 21 89 64

Insertion Sort

32 55 21 74 89 64

Insertion Sort

32 21 55 74 89 64

Insertion Sort

21 32 55 74 89 64

Insertion Sort

21 32 55 74 64 89

Insertion Sort

21 32 55 64 74 89

Analysis of Insertion Sort

- Inserting a new value in sorted segment of length k requires upto k steps in the worst case

- In each iteration, sorted segment in which to insert increased by 1

- $t(n) = 1 + 2 + \ldots + n\text{-}1 = n(n\text{-}1)/2 = O(n^2)$

Recursive formulation

- To sort A[0..n-1]
  - Recursively sort A[0..n-2]
  - Insert A[n-1] into A[0..n-2]
- Base case: n = 1

Insertion Sort, recursive

```
InsertionSort(A,k) // Sort A[0..k-1]

  if (k == 1)
    return;

  InsertionSort(A,k-1);
  Insert(A,k-1);
  return;

Insert(A,j) // Insert A[j] into A[0..j-1]

  pos = j;
  while (pos > 0 && A[pos] < A[pos-1])
    swap(A,pos,pos-1);
    pos = pos-1;
```

Recurrence

- t(n), time to run insertion sort on length n
  - Time t(n-1) to sort segment A[0] to A[n-2]
  - n-1 steps to insert A[n-1] in sorted segment
- Recurrence
  - $t(n) = n-1 + t(n-1)$
    $t(1) = 1$
  - $t(n) = n-1 + t(n-1) = n-1 + ((n-2) + t(n-2)) = \ldots =$
    $(n-1) + (n-2) + \ldots + 1 = n(n-1)/2 = O(n^2)$

$O(n^2)$ sorting algorithms

- Selection sort and insertion sort are both $O(n^2)$
- So is bubble sort, which we will not discuss here
- $O(n^2)$ sorting is infeasible for n over 10000
- Among $O(n^2)$ sorts, insertion sort is usually better than selection sort and both are better than bubble sort
  - What happens when we apply insertion sort to an already sorted list?