CS2002D: Program Design
Lecture-11

INSERTION-SORT(A)

1. **for** j = 2 to A.length c_1

2. key = A[j]; c_2

3. // Insert A[j] into the sorted sequence A[1… j-1]

4. i = j-1 c_3

5. **while** i > 0 and A[i] > key c_4

6. **do** A[i+1] = A[i] c_5

7. i = i-1 c_6

8. A[i+1] = key c_7

While loop within for loop

- **For/while loop** : test is executed one time more than the loop body
- **Let t_j be the number of times** the while loop in line 5 is executed
- Since it is within a for loop, for each j = 2,3,…,n, where n = A.length, **total number of times while loop executed is $\Sigma_{j=2 \text{ to } n} \; t_j$**

INSERTION-SORT(A)	cost	Times
1. for j = 2 to A.length	c_1	n
2. key = A[j];	c_2	n - 1
3. // Insert A[j] into the sorted sequence A[1...j-1]		
4. i = j-1	c_3	n - 1
5. while i > 0 and A[i] > key	c_4	$\Sigma_{j=2 \text{ to } n} t_j$
6. A[i+1] = A[i]	c_5	$\Sigma_{j=2 \text{ to } n} (t_j-1)$
7. i = i -1	c_6	$\Sigma_{j=2 \text{ to } n} (t_j-1)$
8. A[i+1] = key	c_7	n - 1

Running time of an algorithm

- Sum of the running times for each statement executed
  - a statement that takes a cost of $\mathbf{c_i}$ to execute and is executed **n times**, **contribute $\mathbf{c_i}$ * n** to the total running time

T(n): running time of IS : sum of the products of the cost and times

T(n) = ?

What do you think is the best case for IS?

Input:
1,2,3,4,5,6,7,8,9,10

Input:
10,9,8,7,6,5,4,3,2,1

Best case of IS – Already sorted array

- For each j = 2,3,…,n, we know that **A[i] <= key** in line 5, i has its initial value of j – 1

  i.e A[1] <= 2, for j = 2, A[2] <= 3, for j = 3, …..

- **Condition is FALSE and the body of the while loop will not be executed**
- Condition alone will be executed, therefore,

  $t_j = 1$, for j = 2,3,…,n
- Best case running time:

$$T(n) = c_1 n + c_2 (n-1) + c_3 (n-1) + c_4 (n-1) + c_7 (n-1)$$

$$= (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$$

Best case of IS - Linear function

$$T(n) = (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$$

$$= a\,n + b$$, where a and b depend on the statement costs c_i

It's a linear function of n

Worst case of IS - reverse sorted

Input: 10,9,8,7,6,5,4,3,2,1

Compare each element A[j] with each element in the entire sorted subarray A[1… j-1]

i.e for j = 2, A[2] will be compared with A[1]

Resultant array: 9,10,8,7,6,5,4,3,2,1

for j = 3, A[3] will be compared with A[2] and A[1]

Resultant array: 8, 9,10,7,6,5,4,3,2,1

What is the value of tj in the worst case?

5.while i > 0 and A[i] > key		c_4	$\Sigma_{j=2 \text{ to } n} t_j$
6.	**A[i+1] = A[i]**	c_5	$\Sigma_{j=2 \text{ to } n} (t_j-1)$
7.	**i = i -1**	c_6	$\Sigma_{j=2 \text{ to } n} (t_j-1)$

What is t_j for the worst case?

t_j = j, for j = 2, 3, …, n

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

and

$$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

Worst case running time

T(n) = ?

T(n) = a n^2 + b n + c, for constants a, b and c that depends on the statement costs c_i

T(n) quadratic function of n

Worst case running time

- Longest running time for any input of size n
- Upper bound on the running time for any input
- Provides a guarantee that the algorithm will not take more than the specified value
- Worst case occurs fairly often – Searching a database, information is not present

Average case

- As bad as the worst case
- Randomly choose n numbers and apply IS
- How long does it take to insert element A[j] in the sorted subarray A[1…j-1]
- On the average, half the elements are less than A[j] and half the elements are greater than A[j]
- Hence, we check half of the subarray

  A[1… j-1].

  What is the value of t_j ?
- What is the average case running time?

Average case

- Value of tj is j/2

- What is the average case running time?

    - Quadratic function in the size of the input

Ex: What is the expression for the running time in the Average case?

Thank You