**Insertion Sort**





The basic operation of the algorithm is the key comparison A[j ]>v.

**Worst Case Time Complexity:**

The number of key comparisons in this algorithm obviously depends on the nature of the input. In the worst case, *A*[*j* ]*> v* is executed the largest number of times, i.e., for every *j* = *i* − 1*, . . . ,* 0.

In other words, the worst-case input is an array of strictly decreasing values. The number of key comparisons for such an input is

= 1+2+3+….+(n-1)

=

= θ(n2)

**Best Case Time Complexity:**

In the best case, the comparison A[j ]> v is executed only once on every iteration of the outer loop. It happens if and only if A[i − 1]≤ A[i] for every i = 1, . . . , n − 1, i.e., if the input array is already sorted in nondecreasing order. For sorted array, the number of key comparison is

**Average Case Time Complexity:**

For a randomly ordered array insertion sort makes on average half as many comparisons as on decreasing arrays. The task is to determine how many key comparisons are done on the average to insert one new element into the array.

Assume keys are distinct,

For an element in positions i, it has i+1 possible locations to be inserted

The probability is given by

=

=

Now adding for n-1 insertions,

= θ(n2)