Insertion Sort

```
InsertionSort (int A[], int N)
{
 for (int P = 1; P < N; ++P)
  {
   int item = A[P];
   for (int j = P;
       (j > 0)&& (A[j-1] > item);
       --j)
     A[j] = A[j-1];
    A[j] = item;
  }
}
```

10	20	31	5	12

10	20	31	5	12

10	20	31	5	12

5	10	20	31	12

5	10	12	20	31

Insertion Sort (2)

- Worst Case: (input is in reverse order)
  - O(N^2)
- Best Case: (input is already sorted)
  - O(N)
- Average Time:
  - O(N^2)
- (from O(N+I) where I = number of inversions)

Comparison

	Worst	Avg.	Best
Selection Sort:	O(N^2)	O(N^2)	O(N^2)
Bubble Sort:	O(N^2)	O(N^2)	O(N)
Insertion Sort:	O(N^2)	O(N^2)	O(N)

Shell Sort

- compares distant items

- increment sequence:
  - h1 = 1, h2, ..., h(k)
  - Shell's:
    - h(t) = N/2, h(k) = h(k+1)/2

- after a phase (using h(k))
  - A[i] <= A[i+hk]

10	31	5	20	12

5	20	10	31	12

5	10	12	20	31

Example

10	31	5	20	*12*	9	16	27	*2*
2	9	5	20	10	31	16	27	12
2	9	5	20	10	27	12	31	16
2	5	9	10	12	16	20	27	31

81	94	11	96	*12*	35	17	95	*28*
12	35	*11*	95	*28*	94	*17*	96	*81*
11	35	12	94	17	95	28	96	81
11	12	17	28	35	81	94	95	96

Shell Sort (2)

```
void
ShellSort (int A[], int N)
{
 int i, j, Increment, Tmp;

 for (Increment = N/2; Increment > 0; Increment /= 2)
  for (i = Increment; i < N; i++)
  {
   Tmp = A[i];
   for (j = i; j >= Increment; j -=  Increment)
    if (Tmp < A[j - Increment])
     A[j] = A[j - Increment];
    else
     break;
   A[j] = Tmp;
  }
}
```

Shell Sort (3)

- Analysis is difficult
  - for Shell's increments: O(N^2)
    - h(t) = N/2, h(k) = h(k+1)/2
  - for Hibbard's increments: O(N^(3/2))
    - 1, 3, 7, ..., 2^k - 1