Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1: [B, C, D, A, E, H, G, F]

At each stage “insert” $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

```
i = 1:   [B,  C,  D,  A,  E,  H,  G,  F]
i = 2:   [B,  C,  D,  A,  E,  H,  G,  F]
```

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1:	[B, C, D, A, E, H, G, F]
i = 2:	[B, C, D, A, E, H, G, F]
i = 3:	[B, C, D, A, E, H, G, F]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1: [B, C, D, A, E, H, G, F]

i = 2: [B, C, D, A, E, H, G, F]

i = 3: [B, C, D, A, E, H, G, F]

(j = 2) [B, C, D, A, E, H, G, F]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

```
i = 1:    [B,  C,  D,  A,  E,  H,  G,  F]
i = 2:    [B,  C,  D,  A,  E,  H,  G,  F]
i = 3:    [B,  C,  D,  A,  E,  H,  G,  F]
(j = 1)   [B,  C,   ,  D,  E,  H,  G,  F]
```

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1: [B, C, D, A, E, H, G, F]

i = 2: [B, C, D, A, E, H, G, F]

i = 3: [B, C, D, A, E, H, G, F]

(j = 0) [B, , C, D, E, H, G, F]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1:	[B, C, D, A, E, H, G, F]
i = 2:	[B, C, D, A, E, H, G, F]
i = 3:	[B, C, D, A, E, H, G, F]
(j = -1)	[A, B, C, D, E, H, G, F]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

```
i = 1:   [B, C, D, A, E, H, G, F]
i = 2:   [B, C, D, A, E, H, G, F]
i = 3:   [B, C, D, A, E, H, G, F]
i = 4:   [A, B, C, D, E, H, G, F]
```

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1:	[B,	C,	D,	A,	E,	H,	G,	F]
i = 2:	[B,	C,	D,	A,	E,	H,	G,	F]
i = 3:	[B,	C,	D,	A,	E,	H,	G,	F]
i = 4:	[A,	B,	C,	D,	E,	H,	G,	F]
i = 5:	[A,	B,	C,	D,	E,	H,	G,	F]

At each stage “insert” $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1: [B, C, D, A, E, H, G, F]
i = 2: [B, C, D, A, E, H, G, F]
i = 3: [B, C, D, A, E, H, G, F]
i = 4: [A, B, C, D, E, H, G, F]
i = 5: [A, B, C, D, E, H, G, F]
i = 6: [A, B, C, D, E, H, G, F]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1:	[B, C, D, A, E, H, G, F]
i = 2:	[B, C, D, A, E, H, G, F]
i = 3:	[B, C, D, A, E, H, G, F]
i = 4:	[A, B, C, D, E, H, G, F]
i = 5:	[A, B, C, D, E, H, G, F]
i = 6:	[A, B, C, D, E, H, G, F]
(j = 5)	[A, B, C, D, E, H, G, F]

At each stage “insert” $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1:	[B, C, D, A, E, H, G, F]
i = 2:	[B, C, D, A, E, H, G, F]
i = 3:	[B, C, D, A, E, H, G, F]
i = 4:	[A, B, C, D, E, H, G, F]
i = 5:	[A, B, C, D, E, H, G, F]
i = 6:	[A, B, C, D, E, H, G, F]
(j = 4)	[A, B, C, D, E, G, H, F]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

```
i = 1:   [B,  C,  D,  A,  E,  H,  G,  F]
i = 2:   [B,  C,  D,  A,  E,  H,  G,  F]
i = 3:   [B,  C,  D,  A,  E,  H,  G,  F]
i = 4:   [A,  B,  C,  D,  E,  H,  G,  F]
i = 5:   [A,  B,  C,  D,  E,  H,  G,  F]
i = 6:   [A,  B,  C,  D,  E,  H,  G,  F]
i = 7:   [A,  B,  C,  D,  E,  G,  H,  F]
```

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1:	[B, C, D, A, E, H, G, F]
i = 2:	[B, C, D, A, E, H, G, F]
i = 3:	[B, C, D, A, E, H, G, F]
i = 4:	[A, B, C, D, E, H, G, F]
i = 5:	[A, B, C, D, E, H, G, F]
i = 6:	[A, B, C, D, E, H, G, F]
i = 7:	[A, B, C, D, E, G, H, F]
(j = 6)	[A, B, C, D, E, G, H, F]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1:	[B,	C,	D,	A,	E,	H,	G,	F]
i = 2:	[B,	C,	D,	A,	E,	H,	G,	F]
i = 3:	[B,	C,	D,	A,	E,	H,	G,	F]
i = 4:	[A,	B,	C,	D,	E,	H,	G,	F]
i = 5:	[A,	B,	C,	D,	E,	H,	G,	F]
i = 6:	[A,	B,	C,	D,	E,	H,	G,	F]
i = 7:	[A,	B,	C,	D,	E,	G,	H,	F]
(j = 5)	[A,	B,	C,	D,	E,	G,	,	H]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

i = 1:	[B, C, D, A, E, H, G, F]
i = 2:	[B, C, D, A, E, H, G, F]
i = 3:	[B, C, D, A, E, H, G, F]
i = 4:	[A, B, C, D, E, H, G, F]
i = 5:	[A, B, C, D, E, H, G, F]
i = 6:	[A, B, C, D, E, H, G, F]
i = 7:	[A, B, C, D, E, G, H, F]
(j = 4)	[A, B, C, D, E, F, G, H]

At each stage "insert" $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

Maintain a prefix of array ($A[0..i-1]$, shown in green) in sorted order; incrementally increase length of this prefix until it encompasses entire array

```
i = 1:   [B,  C,  D,  A,  E,  H,  G,  F]
i = 2:   [B,  C,  D,  A,  E,  H,  G,  F]
i = 3:   [B,  C,  D,  A,  E,  H,  G,  F]
i = 4:   [A,  B,  C,  D,  E,  H,  G,  F]
i = 5:   [A,  B,  C,  D,  E,  H,  G,  F]
i = 6:   [A,  B,  C,  D,  E,  H,  G,  F]
i = 7:   [A,  B,  C,  D,  E,  G,  H,  F]
i = 8:   [A,  B,  C,  D,  E,  F,  G,  H]
```

At each stage “insert” $A[i]$ (shown in red) among existing $A[0..i-1]$ values while maintaining sortedness

Insertion Sort

```
Algorithm IS(A, n):
  for i $\leftarrow$1 to n$-$1 do
    # Place A[i] in its correct position among A[0]..A[i$-$1]
    curr $\leftarrow$A[i]
    j $\leftarrow$i $-$ 1
    while j $\geq$0 and A[j] $>$ curr do
      A[j+1] $\leftarrow$A[j]
      j $\leftarrow$j$-$1
    A[j+1] $\leftarrow$curr
```

Insertion Sort

Suppose following statement true at beginning of iteration i:

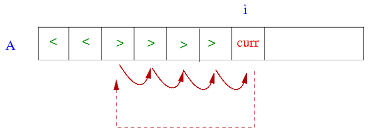
Invarient

Values in $A[0..i-1]$ are in increasing order

[1]

```
Algorithm IS(A, n):
  for i ←1 to n−1 do
     # Place A[i] in its correct
     # position among A[0]..A[i−1]
     curr ←A[i]
     j ←i − 1
     while j ≥0 and A[j] > curr do
        A[j+1] ←A[j]
        j ←j−1
     A[j+1] ←curr
```

- Note inner loop applies only to j where $A[j] > A[i]$ (aka curr) and it shifts each such value one slot rightwards
- Values smaller than $A[i]$ are not moved



- Invariant true at end of iteration i (and hence beginning of iteration $i+1$

[1] Assume values in A distinct (argument generalizes)

Insertion Sort Analysis

```
Algorithm IS(A, n):
  for i ←1 to n−1 do
    # Place A[i] in its correct
    # position among A[0]..A[i−1]
    curr ←A[i]
    j ←i − 1
    while j ≥0 and A[j] > curr do
      A[j+1] ←A[j]
      j ←j−1
    A[j+1] ←curr
```

Inner loop

$$\begin{aligned} \#\text{iterations} &\le i \\ \#\text{comparisons} &\le 2(i+1) \end{aligned}$$

Outer loop

$$\begin{aligned} \#\text{iterations} &\le n-1 \\ \#\text{comparisons} &\le n \\ &\quad +\text{contrib. of inner loop} \end{aligned}$$

$$\#\text{comps (total)} \le n + \sum_{i=1}^{n-1} 2(i+1) = \cdots = n^2 + 2n - 2$$