We can use an ordered dictionary to sort a set of values:

1. Insert the elements one by one with a series of `insert` operations.
2. Remove the elements in sorted order with a series of `smallest()` and `remove()` operations.

The running time of this sorting method depends on the ordered dictionary implementation.

**Algorithm OrdDict-Sort(S)**

- **Input** set of values $S$
- **Output** $S$ sorted increasingly

```
P ← empty ordered dictionary

while $S$ is not empty {
    e ← first value in $S$
    P.insert(e)
}

while P is not Empty {
    e ← P.smallest()
    P.remove(e)
    Print(e)
}
```
Sequence-based Ordered Dictionary

- **Implementation with an unsorted sequence**
  - Store the items of the dictionary in a list-based sequence, in arbitrary order
  - **Performance:**
    - **insert** takes $O(1)$ time since we can insert the item at the beginning or end of the sequence
    - **smallest** and **remove** take $O(n)$ time since we have to traverse the entire sequence to find the smallest key and remove it

- **Implementation with a sorted sequence**
  - Store the items of the dictionary in a sequence, sorted by key
  - **Performance:**
    - **insert** takes $O(n)$ time since we have to find the place where to insert the item
    - **smallest** and **remove** take $O(1)$ time since the smallest key is at the beginning of the sequence
Selection-Sort

- Selection-sort is the variation of OrdDict-sort where the ordered dictionary is implemented with an unsorted sequence.

- Running time of Selection-sort:
  1. Inserting the elements into the ordered dictionary with \( n \) insert operations takes \( O(n) \) time.
  2. Removing the elements in sorted order from the dictionary with \( n \) smallest and remove operations takes \( O(n^2) \) time.
Insertion-Sort

- Insertion-sort is the variation of OrdDict-sort where the ordered dictionary is implemented with a sorted sequence.

Running time of Insertion-sort:

1. Inserting the elements into the dictionary with \( n \) insert operations takes time \( O(n) \) per insertion, so the total time is \( O(n^2) \).

2. Removing the elements in sorted order from the dictionary with a series of \( n \) smallest and remove operations takes \( O(n) \) time.

- Insertion-sort runs in \( O(n^2) \) time.
We can implement selection-sort and insertion-sort without an external data structure. Here is insertion-sort:

**Algorithm** insertion-sort \((A, n)\)

**In:** Array \(A\) storing \(n\) values

**Out:** Array \(A\) sorted in increasing value.

**for** \(i = 1\) **to** \(n-1\) **do** {

\(t = A[i]\)

\(j = i-1\)

**while** \((j >= 0)\) **and** \((A[j] > t)\) **do** {


\(j = j-1\)

}

\(A[j+1] = t\)

**return** \(A\)