SFU

Insertion Sort

CMPT 125
Mo Chen
SFU Computing Science
27/1/2020

Lecture 10

Today

- Insertion Sort

Insertion Sort Algorithm

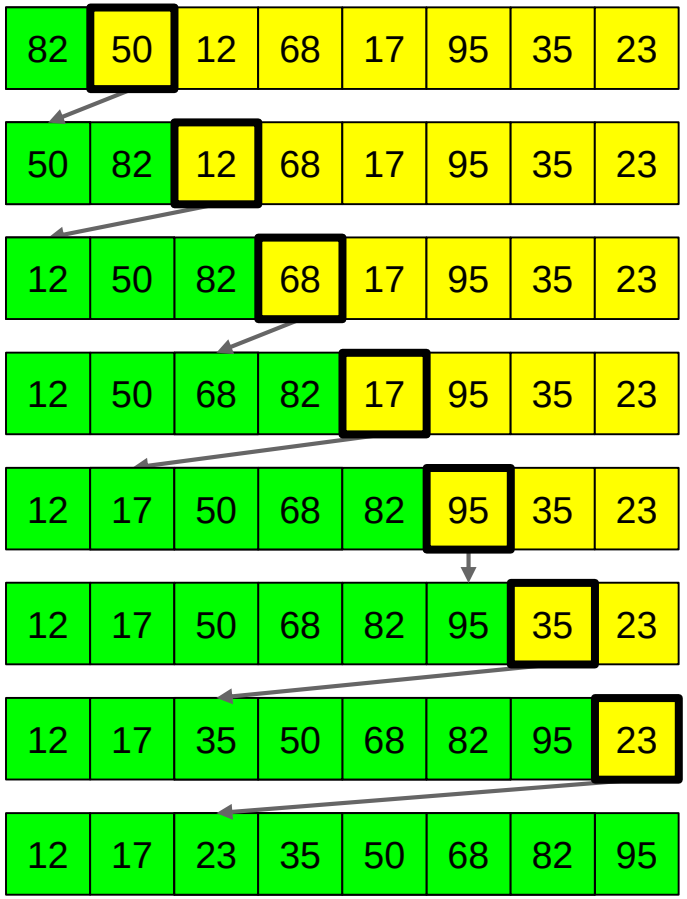
Strategy:

- Insert one element at a time into a sorted list
  - Locate the insertion point
  - Slide array elements to make space
    While new element < array element

- Array divided into two parts: sorted and unsorted (like Selection Sort)
- Sorted part grows one at a time (like Selection Sort)

Insertion Sort Demo

Sort this array using Insertion Sort:



create insertion point in 0 slides

create insertion point in 1 slide

create insertion point in 2 slides

create insertion point in 1 slide

create insertion point in 3 slides

create insertion point in 0 slides

create insertion point in 4 slides

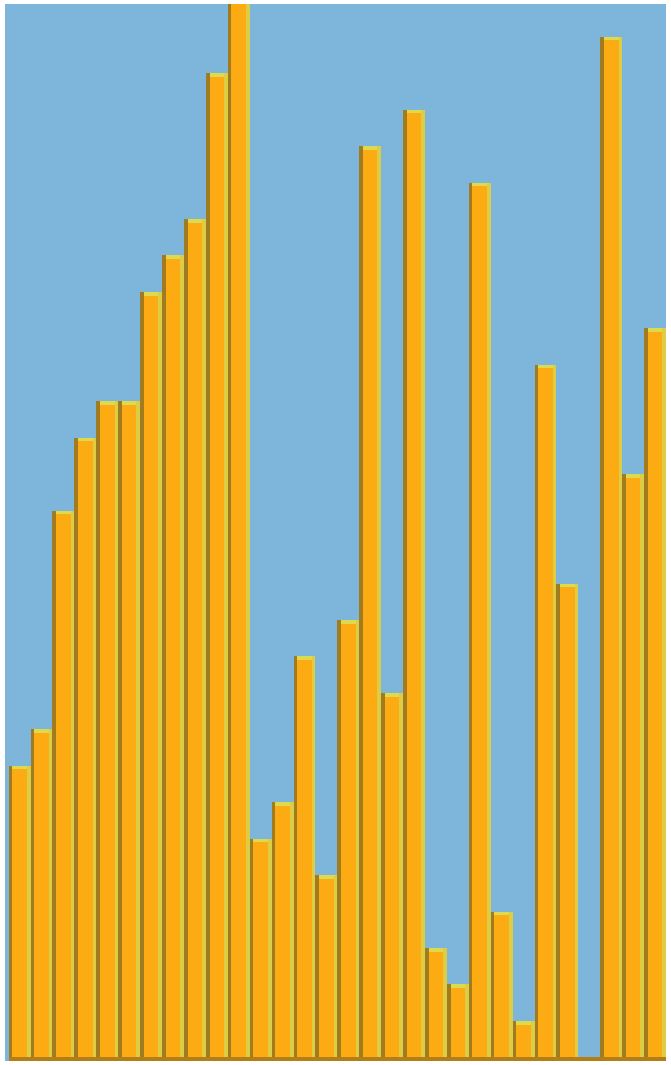
create insertion point in 5 slides

Total number of slides depends on the initial order of the input.

What's the worst case for array of length N?

What's the best case?

A Visualization from Wikipedia



Insertion Sort in C

```
void InsertionSort(int arr[], int len) {
```

- Repeat for all `i` from `1` to `len-1`
  - Slide elements to the right to make a space to insert the new element, `arr[i]`
  - Algorithm:
    - Make a temporary copy of `arr[i]`
    - Linear scan from right to left
    - Slide `while` new element < array element
  - Place new element into position

```
}
```

Insertion Sort in C

```
void InsertionSort(int arr[], int len) {
```

- Repeat for all `i` from `1` to `len-1`

```
int newElement = arr[i];
int j = i;
while (newElement < arr[j-1]) {
        arr[j] = arr[j-1];
        j--;
}
```

- Place new element into position

```
}
```

Insertion Sort in C

```
void InsertionSort(int arr[], int len) {
```

- Repeat for all `i` from `1` to `len-1`

```
    int newElement = arr[i];
    int j = i;
    while (newElement < arr[j-1]) {
        arr[j] = arr[j-1];
        j--;
    }
    arr[j] = newElement;
```

```
}
```

Insertion Sort in C

```
void InsertionSort(int arr[], int len) {
    for (int i = 1; i < len; i++) {

        int newElement = arr[i];
        int j = i;
        while (newElement < arr[j-1]) {
                arr[j] = arr[j-1];
                j--;
        }
        arr[j] = newElement;
    }
}
```

What's the bug?

Array bounds error when `j == 0`

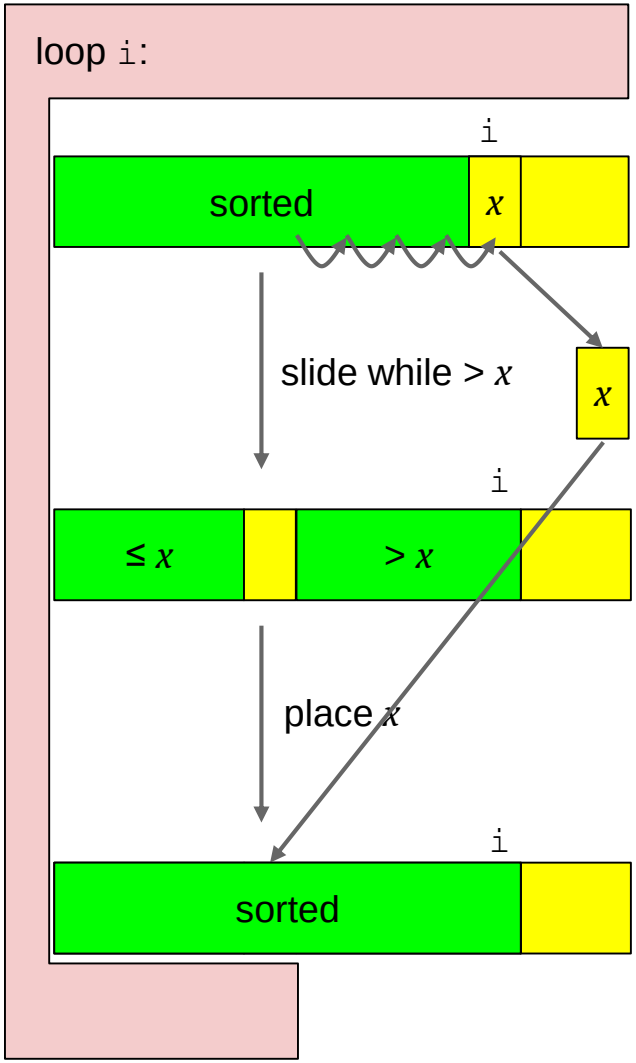
Insertion Sort in C

```
void InsertionSort(int arr[], int len) {
    for (int i = 1; i < len; i++) {
        // Assertion:  At the start of this iteration,
        // arr[0..i-1] are in sorted order
        int newElement = arr[i];
        int j = i;
        while (j > 0 && newElement < arr[j-1]) {
               arr[j] = arr[j-1];
               j--;
        }
        arr[j] = newElement;
    }
}
```

Short circuit eval:
If first part is false, then don't evaluate second part

Assertion Analysis

```
void InsertionSort(int arr[], int len) {
    for (int i = 1; i < len; i++) {
        // At the start of this iteration
        // arr[0..i-1] are in sorted order
        int newElement = arr[i];
        int j = i;
        while (j > 0 && newElement < arr[j-1]) {
            arr[j] = arr[j-1];
            j--;
        }
        arr[j] = newElement;
    }
}
```



Analysis of Insertion Sort

What's the worst case behaviour on an array of length N?

OR . . .

What's the barometer instruction?

Inner loop could be executed `i` times

- `i` slides per loop $\Rightarrow$ $O(N^2)$ total slides (in the worst case)

What sort of input leads to the worst case?

- when input array is reverse sorted

Analysis of Insertion Sort

What's the *best* case?

- When the input array is sorted
- Inner loop executed 0 times $\Rightarrow$ 0 slides

Does this mean a running time of $O(0)$?

- `while` condition is entry condition

  (always performed at least once)

So, $O(N)$ comparisons in the best case

- to verify the array is indeed sorted

Conclusions

- Insertion Sort algorithm varies greatly with nature of input
  - Worst case $O(N^2)$ vastly differs from best case $O(N)$
  - Which case carries more meaning?
- Selection Sort vs Insertion Sort
  - are incremental sorts
  - have same asymptotic running times
- Best sorting algorithms run in $O(N \log N)$
  - New paradigm: Divide & Conquer