Analysis of Insertion Sort
Tune-in exercise

• What is algorithm analysis?
  – Means to predict the resources needed for an algorithm execution.

• What resources are we concerned with?
  – Running time and memory

• Why do we need such resource prediction?
  – To be able to compare algorithms
  – To be able to provision resources for algorithm execution
Example of **insertion sort** on an instance

- Which is the algorithm?
- Which is the input?
- Which is the output?
- What is the instance?

The algorithm:

The input:

The output:

Instance:

\(<5, 2, 4, 6, 1, 3>\)
Example of *insertion sort* on an instance

Take a minute to think on your own of what is happening at each step.
Insertion sort (pseudo code)

**INSERTION-SORT(A)**

1. for $j = 2$ to $A.length$
2. \[ key = A[j] \]
4. \[ i = j - 1 \]
5. \[ while \ i > 0 \ and \ A[i] > key \]
7. \[ i = i - 1 \]
8. \[ A[i + 1] = key \]

This step can be reached when $i=0$ or if $A[i] \leq key$. In both cases, $key$ is placed s.t. $A[1 \ldots i]$ is sorted.

Input array is 1-based

j indexes the whole array

i indexes the sorted sequence
Insertion sort -- analysis

- Recall that each primitive operation takes constant time
- Assume there are $n$ numbers in the input

```plaintext
INSERTION-SORT(A)
1  for $j = 2$ to $A.length$
2     key = $A[j]$
4     $i = j - 1$
5     while $i > 0$ and $A[i] > key$
6         $A[i+1] = A[i]$
7         $i = i - 1$
8         $A[i+1] = key$
```

- $c_1$, $c_2$, and $c_3$ are constants and do not depend on $n$
**Insertion sort -- analysis**

- Assume there are $n$ numbers in the input

```plaintext
\text{INSERTION-SORT}(A)
\begin{align*}
1 & \text{for } j = 2 \text{ to } A.\text{length} \\
2 & \quad \text{key} = A[j] \\
3 & \quad \text{// Insert } A[j] \text{ into the sorted sequence } A[1 \ldots j - 1]. \\
4 & \quad i = j - 1 \\
5 & \quad \textbf{while } i > 0 \text{ and } A[i] > \text{key} \\
6 & \quad \quad A[i + 1] = A[i] \\
7 & \quad \quad i = i - 1 \\
8 & \quad A[i + 1] = \text{key}
\end{align*}
```

What is the time needed for the algorithm execution?
**Insertion sort -- analysis**

- Assume there are **n** numbers in the input

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2     key = A[j]
3     // Insert A[j] into the sorted sequence A[1..j - 1].
4     i = j - 1
5       while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i - 1
8     A[i + 1] = key
```

- **While** loop is executed at most **j-1** times for a given **j**, so time spent in loop is at most 
  \((j-1)c_2\)
- Any iteration of the outer **For** loop takes at most 
  \(c_1 + (j-1)c_2 + c_3\)
- The overall running time of insertion sort is 
  \[\sum_{j=2}^{n}[c_1 + (j-1)c_2 + c_3]= d_1n^2 + d_2n + d_3\]
Was our analysis too pessimistic?

• We just performed a worst-case analysis of insertion sort, which gave us an upper bound of the running time.

• Was our analysis too pessimistic? In other words, are there instances that will cause the algorithm to run with quadratic time in n?
  – The worst-case instance is a reverse-sorted sequence $a_1, a_2, \ldots, a_n$ such that $a_1 > a_2 > \ldots > a_n$

• Since worst-case sequence exists, we say that our analysis is “tight” and ”not pessimistic”.
Insertion sort growth rate

• Consider insertion sort’s running time as the function $d_1n^2 + d_2n + d_3$
  – The dominant part of this function is $n^2$ (i.e. as $n$ becomes large, $n^2$ grows much faster than $n$)
  – Thus, the growth rate of the running time is determined by the $n^2$ term
  – We express this as $O(n^2)$ (a.k.a. “big-oh” notation*)
  – We compare algorithms in terms of their running time

* To be formally defined later
Algorithm comparison

• Which algorithm is better?
  – We answer this question by comparing algorithms’ \( O() \) running times.

• Example: Compare algorithm A and B. Which one is better?
  – Algorithm A: \( O(n^2) \)
  – Algorithm B: \( O(n \log_2(n)) \)

• B is more efficient.
• Intuitively \( n^2 \) grows faster
• We might be wrong for small instances but when \( n \) is large B will be faster
• Large sizes come about very often (Facebook has 100s of millions of users)

By Cmglee - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=50321072
Announcements

• Read through Chapters 1 and 2 in the book
• Homework 1 posted, Due on Sep 7