Algorithm Analysis

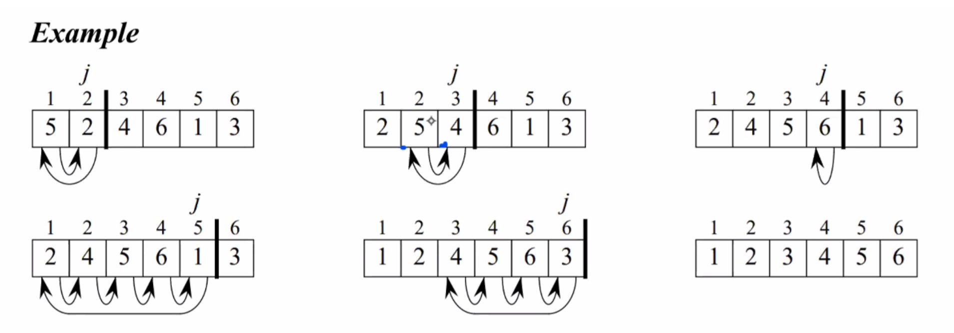
Reagan Shirk

September 1, 2020

Sorting

Insertion Sort

- Reordering, not adding or removing anything (at least for this class, I don't think that's true overall)



- On the quiz, we'll have two problems. The first will be an insertion sort and it is intended to be solved just like a computer would solve it- just run the algorithm by hand.
  - From my experience in this class, the quiz will always be two questions.
    - One question will be running an algorithm, one question will be more theory based
    - We will be told what to expect for at least one of the questions (the first question on Thursdays quiz will be like the example pictured above)
    - The quizzes **are always indexed at 1, not 0**

Efficiency of Insertion Sort

- We don't want to use hardware dependent measurements (such as time) to determine the efficiency of an algorithm
- The right way to analyze time efficiency/complexity is to count the steps
  - This is because an algorithm is a sequence of simple steps, so if we're counting steps we don't need to worry about what machine the algorithm is running on
- Efficiency is a mathematical question

INSERTION-SORT(A)	*cost*	*times*
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1 \,..\, j-1]$.	0	$n-1$
$i = j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n-1$

- You take the cost for each step and the number of times that each step is performed
- The best case for insertion sort is for it to already be sorted in non-decreasing order, the number of steps here is n
- The worst case for insertion sort is for it to be sorted in decreasing order, the number of steps here is n^2
- What is the average case for insertion sort, will it be closer to the best case or the worst case?
  - It's not far away from the worst case **this is something we need to remember**
- **For most algorithms** the average case will be closer to the worst case. There are a couple exceptions, but it is true for most algorithms. That's why we worry about worst case in this class

Correctness

- Loop invariants are used to help understand why an algorithm gives a correct answer. The loop invariant for an insertion sort:
  - At the start of each iteration of the "outer" for loop, the loop indexed by j - the subarray A[1…j-1] consists of the elements originally in A[1…j-1] but in sorted order
- Ya girl missed the part on initialization, maintenance, and termination but I can go back and read the book. It's pretty simple to understand. Basically in a loop you have to have an initialized index, anything that is true before the loop needs to be true inside the loop, and the loop eventually needs to end

Quiz

- This info is burried in my notes, might as well make it easily accessible
- Two questions, last 15 minutes of class
- Cameras need to be on
- First question will be like the insertion sort example above
- Second question will be analyzing time complexity of some given insertion sort
  - The constant isn't important, he just wants the exponent for the n
  - If the first half of the array is sorted in decreasing order and the second half of the array is sorted in increasing order, what's the complexity? n^2