Lecture 3: Insertion Sort

Agenda:

- Worst/average/best case run time
- Correctness
  - Loop invariants
  - Insertion sort correctness

Reading:

- Textbook pages 15 – 27

Insertion sort pseudocode (recall)

```
InsertionSort(A) **sort A[1..n] in place

for j ← 2 to n do
    key ← A[j] **insert A[j] into sorted sublist A[1..j − 1]
    i ← j − 1
    while (i > 0 and A[i] > key) do
        A[i + 1] ← A[i]
        i ← i − 1
    A[i + 1] ← key
```

Analysis of insertion sort

`InsertionSort`(A)	*cost*	*times*
for $j \leftarrow 2$ **to** n **do**	c_1	n
$\quad key \leftarrow A[j]$	c_2	$n-1$
$\quad i \leftarrow j-1$	c_3	$n-1$
$\quad$ **while** $(i > 0$ **and** $A[i] > key)$ **do**	c_4	$\sum_{j=2}^{n} t_j$
$\qquad A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^{n} (t_j - 1)$
$\qquad i \leftarrow i-1$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$\quad A[i+1] \leftarrow key$	c_7	$n-1$

t_j — number of times the `while` loop test is executed for j.

$$T(n) = c_1 n + (c_2 + c_3 - c_5 - c_6 + c_7)(n-1) + (c_4 + c_5 + c_6) \sum_{j=2}^{n} t_j$$

Running time — order of growth in input size

- Consider leading term only
  - lower order terms become insignificant for large n
- Further ignore the constant coefficient
  - constant factor is less significant than the rate
- Insertion sort:
  - best case — $\Theta(n)$
  - worst case — $\Theta(n^2)$

Quick analysis of insertion sort

- Assumptions:
  - (Uniform cost) RAM
  - Key comparison (KC) happens: $i > 0$ and $A[i] > key$
    (minors: loop counter increment operation, copy)
  - RAM running time proportional to the number of KC
- Best case (BC)
  - What is the best case?
    already sorted
  - One KC for each j, and so $\sum_{j=2}^{n} 1 = n - 1$
- Worst case (WC)
  - What is the worst case?
    reverse sorted
  - j KC for fixed j, and so $\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$
- Average case (AC)

Quick analysis of insertion sort (AC)

- Average case: always ask "average over what input distribution?"

- Unless stated otherwise, assume each possible input equiprobable

  Uniform distribution

- Here, each of the __ possible inputs equiprobable (why?)

- Key observation: equiprobable inputs imply for each key, rank among keys so far is equiprobable

- *e.g.*, when $j = 4$, expected number of KC is $\frac{1+2+3+4}{4} = 2.5$

- Conclusion: expected # KC to insert key j is $\frac{\sum_{i=1}^{j} i}{j} = \frac{j+1}{2}$

- Conclusion: total expected number of KC is

$$\sum_{j=2}^{n} \frac{j+1}{2} = \frac{1}{2} \sum_{j=3}^{n+1} = \frac{1}{2} \left(\frac{(n+1)(n+2)}{2} - 3 \right) = \frac{n^2 + 3n - 4}{4}$$

- $\Theta(n^2)$

Correctness of insertion sort

Claim:

- At the start of each iteration of the `for` loop, the subarray $A[1..j-1]$ consists of the elements originally in $A[1..j-1]$ and in sorted order.

Proof of claim.

- initialization: $j = 2$
- maintenance: $j \to j+1$
- termination: $j = n+1$

Loop invariant vs. Mathematical induction

- Common points
  - initialization vs. base step
  - maintenance vs. inductive step
- Difference
  - termination vs. infinite

Correctness & Loop invariant

- Why correctness?
  - Always a good idea to verify correctness
  - Becoming more common in industry
  - This course: a simple introduction to correctness proofs
  - When loop is involved, use loop invariant (and induction)
  - When recursion is involved, use induction
- Loop invariant (LI)
  - Initialization: does LI hold 1^{st} time through?
  - Maintenance: if LI holds one time, does LI hold the next?
  - Termination #1: upon completion, LI implies correctness?
  - Termination #2: does loop terminate?
- Insert sort LI:

  At start of `for` loop, keys initially in $A[1..j-1]$ are in $A[1..j-1]$ and sorted.

  - Initialization: $A[1..1]$ is trivially sorted
  - Maintenance: none from $A[1..j]$ moves beyond j; sorted
  - Termination #1: upon completion, $j = n + 1$ and by LI $A[1..n]$ is sorted
  - Termination #2: `for` loop counter j increases by 1 at a time, and no change inside the loop

Sketch of more formal proof of Maintenance

- Assume LI holds when $j = k$ and so $A[1] \leq A[2] \leq \ldots \leq A[k-1]$
- The `for` loop body contains another `while` loop. Use another LI.
- LI2: let $A^*[1..n]$ denote the list at start of `while` loop. Then each time execution reaches start of `while` loop:
  - $A[1..i+1] = A^*[1..i+1]$
  - $A[i+2..j] = A^*[i+1..j-1]$
- Prove LI2 (exercise)
- Using LI2, prove LI

  Hint: when LI2 terminates, either $i = 0$ or $A[i] \leq key$ (the latter implies either $i = j-1$ or $A[i+1] > key$).

Have you understood the lecture contents?

well	ok	not-at-all	topic
□	□	□	why 3 types of analysis
□	□	□	AC analysis requirements
□	□	□	WC/BC/AC for insertion sort
□	□	□	loop invariant
□	□	□	diff between LI & math induction
□	□	□	correctness of insertion sort by LI