CSE 101 – Sep 30, 2019 (Week 1)

Notes provided by Ben Sihota bsihota@ucsc.edu

Cost of Insertion Sort

Define c_i = cost of step i in insertion sort $(1 \leq i \leq 7)$

Goal: Find runtime $T(n)$ as a function of the length of the input array

$$A[1...n]$$

~ *Unit of the "cost" is not given, but it's usually runtime*

3 Cases

1. Best Case
   - ~ *You can arrange a bookshelf of* n *elements in* $n!$ *ways*
2. Worst Case
   - Take the max $n!$
   - ~ *Focus on the most*
3. Average Case
   - ~ *Most difficult to compute*

Pseudocode

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

$$c_5 * \sum_{j=2}^{n}(t_j - 1) + c_6 * \sum_{j=2}^{n}(t_j - 1) + c_7(n-1)$$

~ c_2 *and* c_3 *is* $n-1$ *since it's in the while loop*

Notation

Let t_j = # of tests of while loop repitition condition on the j^{th} iteration of the outer for loop

1. **Best Case:** $t_j = 1$

$$\therefore \sum_{j=2}^{n} t_j = \sum_{j=2}^{n} 1 = (n-1)$$

$$\therefore T(n) = c_1 n + (c_2 + c_3 + c_4 + c_7)(n-1)$$

$$= (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)$$

2. **Worst Case:** $t_j = j$

$$\sum_{j=2}^{n} t_j = \sum_{j=2}^{n} j = (\sum_{j=1}^{n} j) - 1 = \frac{n(n+1)}{2} - 1$$

$$\sum_{j=2}^{n} (t_j - 1) = \sum_{j=2}^{n} (i-1) = \sum_{j=1}^{n-1}$$

(truncated :/)

So:

$$T(n) = (\frac{1}{2}c_4 + \frac{1}{2}c_5 + \frac{1}{2}c_6)n^2 + \\ (c_1 + c_2 + ...)n \\ +(-c_2 - c_3 ...)$$

3. **Average Case:** $t_j = \frac{i}{2}$

So

$$T(n) = (...)n^2 + (...)n + (...)$$

Summary:

Best: $\frac{T(n)}{an+b}$

$\Theta(n)$

Worst: $cn^2 + dn + e$

$\Theta(n^2)$

Average: $fn^2 + gn + h$

$\Theta(n^2)$

Notation to be defined:

$\Theta, O, \Omega, o, \omega$

Informal definition of Θ

- Drop lower order terms
- Replace leading coefficient by 1

Example

Given algorithm A, B, C, D – all solving same problem

Results:

Algorithm	$T(n)$	Asymp Runtime
A	n^2	$\Theta(n^2)$
B	$10n^2$	$\Theta(n^2)$
C	$10n^2 + 2n - 100$	$\Theta(n^2)$
D	$1000n + 10,000$	$\Theta(n)$

Notice

- $$\frac{C}{B} = (1 + \frac{1}{5n} - \frac{10}{n^2}) \to 1$$
- **Comparing A to B**: A is 10 times better
  - We can equalize by running B on a faster machine
- **Comparing A to D**: *No matter how steep the line or shallow the parabola, there will be a crossover point.* After crossover, D is the better algorithm
  - D is superior

Procedure:

Pick a basic operation and count the number of times it is executed in best, worst, and average case

In sorting algorithms, basic operation is the comparison of array elements

$$A_i < A_j$$

For insertion sort

Algorithm	$T(n)$	Asymp Runtime
Best	$n - 1$	$\Theta(n)$
Worst	$\frac{n(n-1)}{2}$	$\Theta(n^2)$
Average	$\frac{n(n-1)}{4}$	$\Theta(n^2)$