Insertion Sort & running time analysis

Basic operation: insert an element into a sorted list so the final list is still sorted.

\[ j = 2 \quad 0 \]
\[ j = 3 \quad 0 \]
\[ j = 4 \quad 0 \]
\[ \vdots \]
\[ j = n \]

inner loop: comparison

while \( i > 0 \) & \( A[i] \geq \) key

\[ n \]

correctness

proof:

loop invariant

Chap. 2
InsertionSort (A)  // (length [A]=n)
1. for j = 2 to length [A]
   C1
2. do key ← A[j]
   C2
   n - 1
   0
4. i ← j - 1
   C4
   n - 1
5. while i > 0 and A[i] > key
   C5
   \[ \sum_{j=2}^{n} t_j \]
   C6
7. i ← i - 1
   C7
   \[ \sum_{j=2}^{n} (t_j - 1) \] same as above
8. A[i+1] ← key
   C8
   n - 1
The loop head is executed one more time than the loop body.

Total running time

\[ T(n) = C_1 \cdot n + C_2 \cdot (n-1) + C_4 \cdot (n-1) + C_5 \sum_{j=2}^{n} t_j + C_6 \sum_{j=2}^{n} (t_j - 1) \]

\[ + C_7 \sum_{j=2}^{n} (t_j - 1) + C_8 \cdot (n-1) \]

- minimum value \( T(n) \)

- best case: \( t_j = 1 \)

\[ T(n) = C_1 \cdot n + C_2 \cdot (n-1) + C_4 \cdot (n-1) + C_5 \sum_{j=2}^{n} 1 \]

\[ = (C_1 + C_2 + C_4 + C_5 + C_8) \cdot n - (C_2 + C_4 + C_5 + C_8) \]

best case input:

sorted array: 1, 2, 10, 20
- maximum value of $T(n)$

\[
\text{worst case } \quad t_j = j \quad \sum_{j=2}^{n} t_j = \sum_{j=2}^{n} j = 2 + 3 + \ldots + n = \frac{n(n+1)}{2} - 1
\]

\[
T(n) = C_1 n + C_2 (n-1) + C_4 (n-3) + C_5 \left( \frac{n(n+1)}{2} - 1 \right) + C_6 \left( \frac{n(n-1)}{2} \right) + C_7 + \ldots + C_8 (n-1)
\]

\[
= \left( \frac{C_5 + C_6 + C_7}{2} \right) n^2 + \left( C_1 + C_2 + C_4 + \frac{C_5}{2} - \frac{C_6}{2} - C_7 + C_8 \right) n
\]

\[
- (C_2 + C_4 + C_5 + C_8)
\]

\[
= a' n^2 + b' n + c'
\]

worst case input: reversely sorted array 5, 3, 2, 1, 0

- Average case analysis \( t_j = \frac{j}{2} \)
\[ C_5 \cdot \sum_{j=2}^{n} t_j = C_5 \cdot \sum_{j=2}^{n} \frac{1}{2} = \frac{C_5}{2} \sum_{j=2}^{n} j \]

\[ T(n) = \ldots \]

Summary:
worst-case & average case analysis

1. The longest running time for any input size \( n \): the worst case

2. The upperbound of the running time for any input

3. The worst case occurs often: e.g. database search = failed.

4. The average case is often roughly as bad as the worst case.
   roughly half elements \( \leq \) key \( \frac{j}{2} \)
Running time analysis using random-access machine (RAM) model.

RAM: - a generic one-processor

- instructions are executed one after another; no concurrent operations

- Each “simple” operation (+, -, =, if, else, =/<=) take exactly 1 step

- loops & subroutines are * not * simple operations but depend on the size of input.

  e.g. sort() matrix multiplication data & the contents of

- Each memory access takes 1 step
In-class exercise

1. for i = 1 to n
2. T ← i
3. for j = 1 to i^2
4. if j % 2 = = 0 // even or odd?
5. T ← T + j
6. print T

Simplifications/approximations

- Coefficient & lower-order terms can be ignored

<table>
<thead>
<tr>
<th>n</th>
<th>( \frac{3}{2} n^2 )</th>
<th>( \frac{3}{2} n^2 + \frac{3}{2} n - 4 )</th>
<th>% difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>150</td>
<td>181</td>
<td>17%</td>
</tr>
<tr>
<td>100</td>
<td>15,000</td>
<td>15,496</td>
<td>23%</td>
</tr>
<tr>
<td>500</td>
<td>875,000</td>
<td>876,746</td>
<td>0.5%</td>
</tr>
</tbody>
</table>
Asymptotic notations \( O \), \( \Omega \), \( \Theta \), \( o(n) \)

\[ f(n) = O(g(n)) \text{ iff. } \exists c > 0, n_0 \text{ s.t. for } n > n_0, f(n) \leq c \cdot g(n) \]
Goal: you need to know how to prove \( f(n) = O(g(n)) \). Specifically, provide your choices of \( c \) & \( n_0 \).

Examples: prove \( n^2 + 7n + 5 = O(n^2) \).

\[ \text{proof: } \quad f(n) = n^2 + 7n + 5 \]
\[ \leq n^2 + 7n^2 + 5n^2 = 13n^2 \quad \text{when } n \geq 1 \]

Thus, let \( c = 13 \), \( n_0 = 1 \) we have \( f(n) \leq 13n^2 \) when \( n \geq 1 \).
Method: \( f(n) = O(g(n)) \iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \) when \( n \to \infty \)
\( c \cdot g(n) - f(n) = 0 \)

Proof: Let \( c = 2 \) and \( n_0 = 8 \), consider that
\[
\frac{n^2 + 7n + 5}{2n^2} = 0 \quad \text{when} \quad n \approx 7.65
\]
Check that the derivative of difference is negative when \( n \geq 8 \), so \( n^2 + 7n + 5 \leq 2n^2 \) for \( n \geq 8 \)

In-class exercise:
prove: 1. \( 3n^2 + 2n + 5 = O(n^2) \)
2. \( 3n^2 + 2n + 5 = O(n^2 + \log n) \)
3. True or false
if \( f_1(n) = O(f_2(n)) \) and \( f_2(n) = O(f_3(n)) \)
then \( f_1(n) = O(f_3(n)) \)

\[
f(n) = O(g(n)) \text{ iff. the exist a constant } c > 0 \text{ and a value } n_0 \text{ s.t. for any } n > n_0, \quad f(n) \leq c \cdot g(n)
\]