HT2026 Design & Analysis of Algorithms notes

Remaining TODOs: 1

Relevant reading:

T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*

Contents

1. Program cost and asymptotic notation

Definition 1.1: An algorithm is a finite set of well-defined instructions to accomplish a specific task.

Definition 1.2: An efficient algorithm runs in polynomial time.

Definition 1.3: Insertion sort compares each $(i+1)$the element and compares it with the previously sorted i elements, inserting it in the correct place.

In CRLS-style pseudocode (as used in *Introduction to Algorithms*):

```
Input: An integer array A
Output: Array A sorted in non-decreasing order

for j = 1 to A.length - 1
  key = A[J + 1]
  // insert A[j + 1] into the sorted sequence A[1..j]
  i = j
  while i > 0 and A[i] > key
    A[i + 1] = A[i]
    i = i - 1
  A[i + 1] = key
```

Definition 1.4: The running time of a CLRS program is defined as:

- Line i takes constant time c_i
- When a loop exits normally, the test is executed one more time than the loop body

Remark: The running time of insertion sort as given is $T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_5 \sum_{j=1}^{n-1} t_j + c_6 \sum_{j=1}^{n-1}(t_j - 1) + c_7 \sum_{j=1}^{n-1}(t_j - 1) + c_8(n-1)$ where t_j is the number of times the test of the while loop is executed for a given value of j.

Then in the worst case, $t_j = j + 1$, so $T(n) = an^2 + bn + c$ for some a, b, c. Hence $T(n)$ is quadratic in n.

In the best case, $t_j = 1$ so $T(n)$ is linear.

Definition 1.5: Let $f, g : \mathbb{N} \to \mathbb{R}^+$. Then

$$O(g(n)) \coloneqq \{f : \mathbb{N} \to \mathbb{R}^+ \mid \exists n_0 \in \mathbb{N}^+ . \exists c \in \mathbb{R}^+ . \forall n . n \geq n_0 \implies f(n) \leq c \cdot g(n)\}.$$

If $f \in O(g(n))$ then g is an asymptotic upper bound for f.

Proposition 1.6: The algorithm is correct.

Proof:

By a loop-invariant argument:

- **Initialisation** - Prove the invariant I holds prior to first iteration
- **Maintenance** - Prove that if I holds just before an iteration, then it holds just before the next iteration
- **Termination**: Prove that, when the loop terminates, the invariant I along with the reason the loop terminates imply the correctness of the program

This is similar to mathematical induction, but rather than proving for all numbers, we expect to exit the loop.

Invariant: At the start of the jth iteration, `A[1..j]` is sorted.

When $j = 1$, $A[1..j]$ is a singleton so is trivially sorted.

The outer loop terminates when `j = A.length`. So the loop invariant at termination says that `A[1..A.length] = A` is sorted.

To prove maintenance, we need to prove that, at the end of the `while` loop, the sequence `A[1], ..., A[i], key, A[i+2], ..., A[j+1]` are sorted.

The invariant of the `while` loop is: TODO

□

Remark: Some nice properties of insertion sort:

- It is stable (preserves relative order of equal keys)
- In-place
- Online (can sort the list as it is recieved)

Lemma 1.7: Let $f, g, h : \mathbb{N} \longrightarrow \mathbb{R}^+$. Then:

- $\forall c > 0 \;.\; f \in O(g) \implies cf \in O(g)$
- $\forall c > 0 \;.\; f \in O(g) \implies f \in O(cg)$
- $f_1 \in O(g_1) \land f_2 \in O(g_2) \implies f_1 + f_2 \in O(g_1 + g_2)$
- $f_1 \in O(g_1) \land f_2 \in O(g_2) \implies f_1 + f_2 \in O(\max(g_1, g_2))$
- $f_1 \in O(g_1) \land f_2 \in O(g_2) \implies f_1 \cdot f_2 \in O(g_1 \cdot g_2)$
- $f \in O(g) \land g \in O(h) \implies f \in O(h)$
- $\forall p \in \mathcal{P}_n \;.\; p(n) \in O(n^l)$
- $\forall c > 0 \;.\; \lg(n^c) \in O(\lg(n))$
- $\forall c, d > 0 \;.\; \lg^c(n) \in O(n^d)$
- $\forall c > 0, d > 1 \;.\; n^c \in O(d^n)$
- $\forall 0 \leq c \leq d \;.\; c^n \in O(d^n)$

Definition 1.8: If $f(n) \in \Omega(g(n))$, we say that g is an asymptotic lower bound for f.

Corollary 1.9: $f(n) \in O(g(n)) \iff g(n) \in \Omega(f(n))$.

Definition 1.10: If $f(n) \in \Theta(g(n))$, g is an asymptotic tight bound for f.

$$f(n) \in O(g(n)) \land f(n) \in \Omega(g(n)) \iff f(n) \in \Theta(g(n)).$$

Remark: Big $O/\Omega/\Theta$ are not closed under function composition.

Index