CSC265 Notes

Max Xu

'25 Fall

Contents

§1 Day 1: Review and ADTs (Sept 3, 2025)

§1.1 ADTs

Abstract data types (ADTs) are a mathematical object, which has a set of operations. Examples include a set, a sequence, or a graph.

A *data structure* that implements an ADT provides a representation for the object in memory and algorithms for each operation. There may exist many different data structures that implement the same ADT, with varying running time for each operation.

Example 1.1 (Dictionary ADT)

The ADT consists of:

- **Object**: a set of elements, with each having a unique key from totally ordered universe U
- **Operations**:

  Insert(S, x) adds an element with key x to the set S if S does not contain an element with key x

  Delete(S, x) removes an element with key x from S if it exists

  Search(S, x) returns a pointer to the element in S with key x, or nil if such an element doesn't exist

Here we are not worried about key-value pairs, although a specific implementation of this ADT might. We could implement this as a set of keys, with no associated element for each key.

An example implementation of dictionary ADT using singly linked list (unsorted) can have the following properties:

§1.2 Review

Let $t(x)$ be the number of steps taken by an algorithm A on input x.
Let the worst case step complexity of A be $T(n)$, where

$$T(n) = \max\{t(x) \mid x \text{ is an input of size } n\}$$

Typically, this is very hard to determine exactly, which is why asymptotic notation is used instead, as it still captures how quickly $T(n)$ grows with respect to n.

$T(n) \in O(f(n))$ if there exists a constant $c, n_0 \in \mathbb{N}$, for all $n \in \mathbb{N}$ such that when $n > n_0$, $T(n) \leq cf(n)$. $T(n) \in \Omega(f(n))$ if there exists a constant $c, n_0 \in \mathbb{N}$, for all $n \in \mathbb{N}$ such that when $n > n_0$, $cf(n) \leq T(n)$. $T(n) \in \Theta(f(n))$ if $T(n) \in O(f(n))$ and $T(n) \in \Omega(f(n))$.

For upper bound, show that there exists positive constant c and for all n large enough, for every input of size n, the algorithm takes at most $cf(n)$ steps. We write $T(n) \in O(n)$.

For lower bound, show that there exists positive constant c and for all n large enough, there exists some input of size n that makes the algorithm take at least $cf(n)$ steps. We write $T(n) \in \Omega(n)$.

BubbleSort($A[1...n]$**)**

```
1: last = n
2: sorted = False
3: while not sorted do
4:     sorted = True
5:     for j = 1 to last − 1 do
6:         if A[j] > A[j + 1] then
7:             swap A[j] and A[j + 1]
8:             sorted = False
9:         end if
10:    end for
11:    last = last − 1
12: end while
```

Upper Bound

The outer loop can occur at most n times, as last starts at n and decrements by 1 every time the outer loop runs. The inner loop goes from 1 to last $-$ 1, meaning there are at most last $-$ 1 iterations of the inner loop. This can be written as the sum

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

meaning for this algorithm $T(n) \in O(n^2)$.

Lower Bound

We can pick the list $A = [n, n-1, \ldots, 2, 1]$. This list has the property that the first element is the largest number, barring the sorted block of size $i-1$ present at the end of the array at the start of the i-th iteration.[1] This means that it will take $n-i$ swaps, which means it takes

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

steps total, meaning for this algorithm $T(n) \in \Omega(n^2)$.

Theta Bound

As $T(n) \in \Omega(n^2)$ and $T(n) \in O(n^2)$, we have $T(n) \in \Theta(n^2)$.

[1] i think iterations start at 1 in this course?