

1 Data Structure

An Abstract Data Type (ADT) is a set of objects together with operations that can be performed on these objects.

Example 1. Stack

- object: ordered list.
- operations: PUSH, POP, ISEMPY.

A Data Structure is an implementation of an ADT i.e., a concrete way to store the objects together with algorithm for the operations.

Example 2.

- Implementing stacks using an array with a counter for the top;
- Linked Lists implementation.

In general, an ADT is a way to describe *what* the data is, and what you can do with it, while a Data Structure is a way to describe *how* the data is stored and *how* the operations are performed. In this course, we will show you many ADT's, and many Data Structures for these ADTs.

2 Algorithm Analysis

What is the “complexity” of an algorithm? Conceptually, it is the amount of resources, measured as a function of the size of the input. Why analyze the complexity of algorithms? In order to be able to compare them, so that we can choose between different implementations of an ADT.

Complexity usually means “running time”, but it could also mean

- space (the amount of memory used by the algorithm);
- # of logic gates in a circuit, # of bits of communication in a network, # of nodes and edges in a graph, etc.

Generally running time is measured at a high-level also, by simply counting the number of “steps” taken by the algorithm, where the definition of “step” will depend on the algorithm, e.g., $T(N) = 2n^2 + 3$ means a given algorithm takes $T(N)$ steps for a given input size n .

3 Mathematical Proof

A proof is a sequence of statements ending in a state (to be proved), which is either given to be true, or an axiom, or following from previous statements using the rule of logics. Three commonly used proof techniques are direct proof, proof by contradiction, or by mathematical induction.

3.1 Direct Proof

Example 3. For all $n \in \mathbb{N}$, $n \geq 3$, $n^2 \geq 2n + 1$ (Lemma 1¹)

pf.

$$\begin{aligned} n &\geq 3 \\ n \times n &\geq 3 \times n \\ &= 2n + n \\ &\geq 2n + 1 \end{aligned}$$

Example 4. At least one of Alice, Bob, or Carol is guilty (1), if Alice is guilty, then so is Bob (2), if Alice is not guilty, then neither is Carol (3). To prove: Bob is Guilty.

pf. Consider cases:

- Alice is guilty, then so is Bob (2);
- If Bob is guilty, then he is guilty (axiom);
- If Carol is guilty, then Alice must also be guilty (from 3), consequently so is Bob (from 2 again).

But we have in addition (1), hence, Bob must be guilty.

3.2 Proof by Contraction

Example 5. Same example as above, Alice, Bob, and Carol. To prove Bob is guilty by contraction.

pf. Assume that Bob is not guilty, then Alice is not guilty (from 2), then Carol is not guilty (from 3). So no one would be guilty, but this is in contradiction to (1). Bob must be guilty.

3.3 Mathematical Induction

Mathematical induction is widely used in proving properties about all natural numbers \mathbb{N} (or certain number bigger than 1). The basic idea of mathematical induction is that, let $S(n)$ be some statement we care about (e.g., $S(n)$ is “ $2^n \geq n^2$ ”) to prove that for all $n \in \mathbb{N}$, $n \geq c$, where c is 1 or some number greater, $S(n)$ holds, we prove that

- (*base step*) $S(c)$ holds,
- (*induction step*) for all $n \in \mathbb{N}$, $n \geq c$, if $S(n)$ holds then $S(n + 1)$ holds also.

Example 6 (Example 2.11 in the textbook). For all $n \in \mathbb{N}$, $\sum_{i=1}^n i = \frac{n(n+1)}{2} = S(n)$.

pf. Base step: $1 = S(1) = \frac{1(1+1)}{2} = 1$.

¹The lemma is to be used in Example 7.

Induction Step: assume that for all $n \in \mathbb{N}$, $n \geq 1$ $S(n)$ holds and $\frac{n(n+1)}{2} = S(n)$, prove that $S(n+1)$ holds, where $S(n+1) = \frac{(n+1)(n+2)}{2}$.

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

■

Example 7. For all $n \in \mathbb{N}$, $n \geq 4$, $2^n \geq n^2$.

pf. Base step: $S(4)$ holds: $2^4 \geq 4^3$. Induction Step: assume that for all $n \in \mathbb{N}$, $n \geq 4$, $S(n)$, which is “ $2^n \geq n^2$ ”, holds (\star) , prove that $S(n+1)$, which is “ $2^{(n+1)} \geq (n+1)^2$ ”, also holds.

$$\begin{aligned} 2^{(n+1)} &= 2 \cdot 2^n \\ &\geq 2 \cdot n^2 && (\text{by } \star) \\ &= n^2 + n^2 \\ &\geq n^2 + (2n+1) && (\text{Lemma 1}) \\ &= (n+1)^2 && (\text{Since } n \geq 3) \end{aligned}$$

Conclusion: it follows by induction that for all $n \in \mathbb{N}$, $n \geq 4$, $2^n \geq n^2$. ■

4 Asymptotic Analysis

Time Complexity: for each algorithm, we want to find $T : \mathbb{N} \rightarrow \mathbb{N}$ such that $T(n)$ is (maximum) number of steps required to execute the algorithm on an input of size n , e.g. $T(n) = 2n^2 + 3$.

Let $g : \mathbb{N} \rightarrow \mathbb{N}$, $O(g)$ is defined by the following: $O(g) = \{f : \mathbb{N} \rightarrow \mathbb{N} \mid \text{there exists a constant } c \in \mathbb{R}^+ \text{ and an } n_0 \in \mathbb{N} \text{ such that for all } n \in \mathbb{N} \text{ and } n \geq n_0, f(n) \leq c \cdot g(n)\}$.

Let $g : \mathbb{N} \rightarrow \mathbb{N}$, $\Omega(g)$ is defined by the following: $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{N} \mid \text{there exists a constant } d \in \mathbb{R}^+ \text{ and an } m_0 \in \mathbb{N} \text{ such that for all } n \in \mathbb{N} \text{ and } n \geq m_0, f(n) \geq d \cdot g(n)\}$.

Let $g : \mathbb{N} \rightarrow \mathbb{N}$, $\Theta(g)$ is defined as the union of $O(g)$ and $\Omega(g)$.

To show that $f \in O(g)$, we can either

- use first principles by find a $c \in \mathbb{R}^+$ and an $n_0 \in \mathbb{N}$ to prove that for all $n \in \mathbb{N}$ and $n \geq n_0$, $f(n) \leq c \cdot g(n)$; or
- use the following four handy rules

1. for all f , $f \in O(f)$;
2. if f is a constant, then $f \in O(1)$;
3. if $f_1 \in O(g_1)$ and $f_2 \in O(g_2)$ then $(f_1 \times f_2) \in (O(g_1) \times O(g_2))$;
4. if $f_1 \in O(g_1)$ and $f_2 \in O(g_2)$ then $(f_1 + f_2) \in (\max[g_1, g_2])$.

Remember:

$n \in O(n), n \in O(n^2), n \in O(n^3), \dots$
 $n^2 \notin O(n), n^2 \in O(n^2), n^2 \in O(n^3), \dots$
 $n^3 \notin O(n), n^3 \notin O(n^2), n^3 \in O(n^3), \dots$

$n \in \Omega(n), n \notin \Omega(n^2), n \notin \Omega(n^3), \dots$
 $n^2 \in \Omega(n), n^2 \in \Omega(n^2), n^2 \notin \Omega(n^3), \dots$
 $n^3 \in \Omega(n), n^3 \in \Omega(n^2), n^3 \in \Omega(n^3), \dots$

Example 8. $f_1(n) = n^2 + 5 \in O(n^2)$.

pf. Let $n_0 = 3$ and $c = 2$, we claim that for all $n \in \mathbb{N}$ and $n \geq 3$, $n^2 + 5 \leq c \cdot g(n)$: from $3 \leq n$, we have $5 \leq n^2$, so $n^2 + 5 \leq n^2 + n^2 = 2 \cdot n^2$. ■

Example 9. $f_2(n) = 3n^2 + 4n - 2 \in O(n^2)$.

pf. Let $n_0 = 4$ and $c = 4$, we claim that for all $n \in \mathbb{N}$ and $n \geq 4$, $3n^2 + 4n - 2 \leq c \cdot g(n)$: from $4 \leq n$, we have $4 - \frac{2}{n} \leq n$, so $4n - 2 \leq n^2$, $3n^2 + 4n - 2 \leq 3n^2 + n^2 = 4 \cdot n^2$. ■

Example 10. $f_1(n) = n^2 + 5 \notin O(n)$.

pf. We need to show $n^2 + 5 \notin O(n)$. Suppose that we have some c , we show that for sufficiently large n , it is the case $n^2 + 5 > c \cdot n$. Now choose $n > c$, then $n^2 > c \cdot n$, then $n^2 + 5 > cn$. ■

Example 11. $f_2(n) = 37n + 14 \in O(n)$.

textbfpf. Let $n_0 = 14$ and $c = 38$, we claim that for all $n \in \mathbb{N}$ and $n \geq 14$, $37n + 14 \leq 38n$: since $14 \leq n$, it holds $37n + 14 \leq 38n$. ■

Example 12. $3n^2 + 4n \in O(n^2)$.

pf. $3 \in O(1)$, $n^2 \in O(n^2)$, hence $3n^2 \in O(n^2)$. $4 \in O(1)$, $n \in O(n)$, hence $4n \in O(n)$. Hence, $3n^2 + 4n \in O(n^2)$. ■

Example 13. $5n^2 \log 2^n + n^3 \in O(n^3)$

pf. $5 \in O(1)$, $n^2 \in O(n^2)$, and $\log 2^n \in O(\log 2^n)$, hence $5 \cdot n^2 \cdot \log 2^n \in O(n^2 \cdot \log 2^n)$. Hence $5 \cdot n^2 \cdot \log 2^n + n^3 \in O(\max[n^2 \cdot \log 2^n, n^3]) = O(n^3)$. ■.