Vishal Neeli

January 24, 2021

1 Recursion

1.1 Design principle

1. Reduce the problem of size n to a size n-1
2. Figure out the base case (usually n=0 or 1)
3. Terminate recursion at the base case. Make sure that every n reaches the base case.

1.2 Parameterization

- It is consumes extra memory if pass a new array (part of old array copied into this) for recursion. Instead we should pass the indices as the parameters.
- In creating recursive methods, it is often useful to define additional functions(or methods) to facilitate recursion.

1.3 Tail Recursion

- Recursion has to maintain function calls on stack that makes it more expensive than iterative methods.
- To reduce this extra usage of resources, we can write a algorithm in a tail recursive way i.e, the function should directly return the function call (without any other operations on the result returned by the function call).

2 Algorithm analysis

We primarily compare running time in this course. To analyse algorithms, runtime should be machine independent for which we use 'RAM' model of computation.

2.1 RAM model

- Generic single processor model
- Computer supports simple constant time instructions
  - Arithmetic $(+, -, \times, /, floor, ..)$
  - Data movement (load, store, copy)
  - Control (branch, function call)
- We assume that the cost (runtime) of all simple instructions is 1
- Sequential execution - No concurrent execution
- Flat memory model and accessing a memory costs 1 unit.

2.2 Asymptotic Notation

- Θ notation: Given functions g, we define

  $$\Theta(g(n)) = \{f(n) : \exists \text{ positive constants } c_1, c_2, n_0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}$$

  g(n) is a asymptotic tight bound for f(n).

- O notation: Given functions g, we define

  $$O(g(n)) = \{f(n) : \exists \text{ positive constants } c, n_0 \text{ such that } f(n) \leq cg(n), \forall n \geq n_0\}$$

  g(n) is a asymptotic upper bound for f(n).

- Ω notation: Given functions g, we define

$$\Omega(g(n)) = \{f(n) : \exists \text{ positive constants } c, n_0 \text{ such that } cg(n) \leq f(n), \forall n \geq n_0\}$$

g(n) is a asymptotic lower bound for f(n).