Lec 7

Recursive Correctness

2025-06-02

Review

Recursion Trees

Master Theorem

Let $T(n) = aT(n/b) + f(n)$. Define the following cases based on how the root work compares with the leaf work.

1. **Leaf heavy**. $f(n) = O(n^{\log_b(a)-\epsilon})$ for some constant $\epsilon > 0$.
2. **Balanced**. $f(n) = \Theta(n^{\log_b(a)})$
3. **Root heavy**. $f(n) = \Omega(n^{\log_b(a)+\epsilon})$ for some constant $\epsilon > 0$, and $af(n/b) \leq cf(n)$ for some constant $c < 1$ for all sufficiently large n.

Then,

$$T(n) = \begin{cases} \Theta(n^{\log_b(a)}) & \text{Leaf heavy case} \\ \Theta(f(n)\log(n)) & \text{Balanced case} \\ \Theta(f(n)) & \text{Root heavy case} \end{cases}$$

New Stuff

Correctness

- For any algorithm/function/program, define a precondition and a postcondition.

- The **precondition** is an assertion about the inputs to a program.
- The **postcondition** is an assertion about the end of a program.

- An algorithm is **correct** if the precondition implies the postcondition.

- *“If I gave you valid inputs, your algorithm should give me the expected outputs.”*

Examples

Correctness of Mergesort

Assuming merge is correct

```
1  def merge_sort(l):
2      n = len(l)
3      if n <= 1:
4          return l
5      else:
6          left = merge_sort(l[:n//2])
7          right = merge_sort(l[n//2:])
8          return merge(left, right)
```

Multiplication

Grade School

Multiplication

Divide and Conquer

Karatsuba

```
 1  def karatsuba(x, y):
 2      if x < 10 and y < 10:
 3          return x * y
 4      n = max(len(str(x)), len(str(y)))
 5      m = n//2
 6      x_l = x // 10**m
 7      x_h = x % 10**m
 8      y_l = y // 10**m
 9      y_h = y % 10**m
10      z_0 = karatsuba(x_l, y_l)
11      z_1 = karatsuba(x_l + x_h, y_l + y_h)
12      z_2 = karatsuba(x_u, y_h)
13      return (z_2 * 10**(2*m)) + ((z_1 - z_2 - z_0) * 10**m) + z_0
```

Summary

Binary Search

```
def bin_search(l, t, a, b):
    if b == a:
        return None
    else:
        m = (a + b)//2
        if l[m] == t:
            return m
        elif l[m] < t:
            return bin_search(l, t, m+1, b)
        elif l[m] > t:
            return bin_search(l, t, a, m)
```