Recurrences

Alexandre David

- A **recurrence** is an equation or inequality that describes a function in terms of its value on smaller inputs.
- Methods for solving recurrences:
  - Substitution method
  - Recursion-tree method
  - Master method

- Two steps:
  - Guess the form of the solution
  - Use induction to find the constants and prove that the solution works
- Problem: come up with a good guess
  - Use recursion trees
  - Correct the guess

- Each node represents the cost of a single subproblem.
- Useful when it describes the running time of a divide-and-conquer algorithm.
- Used to generate a good guess or as a direct proof of a solution to a recurrence.
- Example: $T(n)=3T(n/4)+\theta(n^2)$
  - Construct recursion tree to obtain a guess
  - Use the substitution method for the proof

The Master Method

- For recurrences of the form *T(n)=aT(n/b)+f(n)* where *a≥1* and *b>1* are constants and *f(n)* is asymptotically positive.
- Theorem: *T(n)* can be bounded as follows
  - if $f(n)=O(n^{\log_b a-\epsilon})$ for some ε>0 then $T(n)=\Theta(n^{\log_b a})$
  - if $f(n)=\Theta(n^{\log_b a})$ then $T(n)=\Theta(n^{\log_b a}\lg n)=\Theta(f(n)\lg n)$
  - if $f(n)=\Omega(n^{\log_b a+\epsilon})$ for some ε>0 and if *af(n/b)≤cf(n)* for some *c<1* then $T(n)=\Theta(f(n))$
- Intuition: compare f(n) to $n^{\log_b a}$, the larger is the solution. Smaller: polynomially smaller by a factor of n^{ϵ}. Larger: polynomially larger + "regularity" condition.

Master Method: Examples

- *T(n)=9T(n/3)+n*

a=9, b=3, f(n)=n $n^{\log_b a}=n^{\log_3 9}=n^2$

Case 1 with ε=1, we conclude *T(n)=Θ(n²)*.

- *T(n)=T(2n/3)+1*

a=1, b=2/3, f(n)=1 $n^{\log_b a}=n^{\log_{3/2} 1}=n^0=1$

Case 2, we conclude *T(n)=Θ(*lg *n).*

- *T(n)=3T(n/4)+n*lg*n* $n^{\log_b a}=n^{\log_4 3}=O(n^{0.793})$

*a=3, b=4, f(n)=n*lg*n* $f(n)=\Omega(n^{\log_b a+\epsilon})$

Case 3, check regularity:

*af(n/b)=3(n/4)*lg*(n/4)≤(3/4)n*lg*n=cf(n).*

We conclude *T(n)=Θ(n*lg*n).*

Master Method: Example

- *T(n)=2T(n/2)+n*lg*n*

*a=2, b=2, f(n)=n*lg*n,* $n^{\log_b a}=n$

Case 3?

*f(n)=n*lg*n* is not polynomially larger than *n*: no ε>0 such that *n*lg*n*=Ω(n^{1+ε}).

We cannot apply the master theorem.

Master Theorem: Proof Idea

- Proof for a sub-domain: *n=1,b,b²,...*
  - compute the cost with a recursion tree (lemma 4.2)
    leaves: $\Theta(n^{\log_b a})$
    +tree: $\sum_{j=0}^{\log_b n-1} a^j f(n/b^j)$
  - bound the 2nd term with 3 cases (as in the theorem) (lemma 4.3)
  - evaluate the sum (asymptotically) using lemma 4.3.
- Extend the proof for any *n*.