Lecturer: Andrew Parkes
http://www.cs.nott.ac.uk/~pszajp/

G52ACE 2017-18
The "Master Theorem"

Recap: Recurrence Relations

- A recurrence relation is a sort of recursively defined function
  - But, generally, applied to the case when the function is some measure of resources …
  - and so we might only want the big-Oh family properties of the solution

- Suppose that the runtime of a program is T(n), then a recurrence relation will express T(n) in terms of its values at other (smaller) values of n.

Recap: Example: Merge Sort

- Suppose the runtime of merge-sort of an array of n integers is $T(n)$. Then

$$T(n) = 2\,T(n/2) + b + a\,n$$

  - "$2\,T(n/2)$" is due to having to sort the two sub-arrays each of size $n/2$
  - "b" is the cost of doing the split
  - "$a\,n$" is the cost of doing the merge (and any copying to/from the workspace, etc.)

Recap: Solving Recurrence

- General pattern
  1. Starting from the base case, use the recurrence to work out many cases, by directly substituting and working upwards in values of n
  2. Inspect the results, look for a pattern and make a hypothesis for the general results
  3. Attempt to prove the hypothesis – typically using some form of induction

  Often then extract the large n behavior using big-Oh family

  Long and tedious, but many cases are covered by a general rule with the name of "Master theorem"

Master Theorem (MT)

Consider recurrence relations of the form

$$T(n) = a\,T(n/b) + f(n)$$

- Designed for "divide and conquer" in which problems are divided into 'a' instances of a problem of size n/b.
- Aim is to be able to quickly express the "big-Oh family" behavior of T(n) for various cases of the values of a and b, and the scaling behavior of f(n).

It does not cover all cases, but does cover many useful cases.

Master Theorem

- No proof needed in this module.
- Just learn it, and how to apply it!
- Suggest to generate and try many examples

Motivations

- Consider the special case that $f(n) = 0$
- $T(n) = a\,T(n/b)$ with $T(1) = 1$
  - $T(b) = a$
  - $T(b^2) = a^2$
  - $T(b^3) = a^3$
- So $T(b^k) = a^k$
- Exercise (offline): prove by induction

Motivations

- Consider the special case that $f(n) = 0$
  - $T(n) = a\,T(n/b)$ with $T(1) = 1$
  - Gives $T(b^k) = a^k$
- But now suppose $f(n) = n^c$ for some c
- We can ask which term dominates; the recurrence or the f(n)?
- Put $n = b^k$ then compare
  - "Recurrence term": $a^k = (b^{log\_b(a)})^k$.
    - Note: used the identity that $a == b^{log\_b(a)}$
  - "f term" $n^c = (b^k)^c = (b^c)^k$
- So need to compare c and $\log_b(a)$

Master Theorem (MT): Cases

T(n) = a T(n/b) + f(n)

Results are split into 3 cases, according to comparing the growth rate of f(n) to $n\hat{}(\log_b(a))$

- Case 1: f(n) "grows slower". Recurrence term dominates. "Solution ignores f"
- Case 2: f(n) grows same – up to log factors – "mix of recurrence with a,b, and also the f(n) term"
- Case 3: f(n) grows faster. "Solution ignores recurrence terms and a,b"

MT: Case 1

$T(n) = a\,T(n/b) + f(n)$

$f(n)$ is $O(n^c)$ with $c < \log_b a$

Note: it is "<" not "<=" and it is a "big-Oh"

Then $T(n)$ is $\Theta(n \wedge (\log_b a))$

That is, $T(b^k)$ grows as $b^{k \log\_b a} = a^k$,

as expected from earlier

MT: Case 1: Example

$T(n) = 2\ T(n/2) + d$

$a = 2$, $b=2$ so $\log_b(a) = \log_2(2) = 1$

$f(n)$ is $O(1)$ which is $O(n^c)$ with $c = 0$
and note that $c < \log_b(a)$

Then $T(n)$ is $\Theta(n)$

MT: Case 2

T(n) = a T(n/b) + f(n)

if

f(n) is $\Theta(n^c(\log n)^\wedge k)$

with c = $\log_b$ a and $k \geq 0$

(Note: it is "c =" and Big-Theta)

Then T(n) is $\Theta(n^c\,(\log n)^\wedge(k+1)\,)$

Note the growth depends on both the recurrence, a,b, and also depends on f via k.

MT: Case 2: Example

$T(n) = 2\,T(n/2) + 3\,n$

$f(n)$ is $\Theta(\,n\,(\log n)^k\,)$

with

$c = \log_2 2 = 1,$

and

$k=0$

Then $T(n)$ is $\Theta(n \log n)$

(Same as merge sort of previous lecture)

MT: Case 3

T(n) = a T(n/b) + f(n)

f(n) is $\Omega(n^c)$ with c > $\log_b$ a

Notice: it is "c > .." and big-Omega!

And f(n) satisfies the "regularity condition"

a f(n/b) <= k f(n) for some k < 1

Then T(n) is $\Theta(f(n))$.

Growth is dominated by f(n) and so a,b of the recurrence are not used.

MT: Case 3 : Example

$T(n) = 2\,T(n/2) + n^2$

$f(n)$ is $\Omega(n^c)$ with $c = 2 > \log_b a = \log_2 2 = 1$

Also, $f(n)$ satisfies the "regularity condition"

$$2\,f(n/2) = 2\,(n/2)^2 <= k\,f(n) \quad \text{with } k=1/2$$

Then $T(n)$ is $\Theta(n^2)$

Regularity Condition

- a f(n/b) <= k f(n)   for some k < 1
- Suppose f(n) = d n^c   then we need
  - a d (n/b)^c <= k d n^c   for some k < 1
  - a / b^c <= k   for some k < 1
  - a / b^c < 1
  - a < b^c
  - Now take log_b of both sides
  - Need log_b(a) < c
  - Which is already satisfied for case 3 to apply.
  - So is not a new condition in this case (needed for more complex cases)

MT Example

- $T(n) = 4\,T(n/2) + d\,n$ with $T(1)=1$
- $a=4$, $b=2$
- so $\log_b a = \log_2 4 = \log_2 2^2 = 2$

- $f(n)$ is $O(n^c)$ with $c=1 < 2$
- So is case 1.

- Hence is $\Theta(\, n^2 \,)$
  - Matches the exact solution in previous lecture.

Expectations

- Know and understand the Master Theorem (MT)
    - Be able to apply it to examples
    - (Do not need to be able to prove the MT itself!)
- May well be asked to solve a recurrence relation exactly, and then also to solve it using the Master Theorem