CMSC351-2023-09-27

Wednesday, September 27, 2023 8:11 AM

```
function binarysearch(A,TARGET)
    L = 0
    R = n-1
    while L <= R
        C = floor((L+R)/2)
        if A[C] == TARGET
            return C
        elif TARGET < A[C]
            R = C-1
        elif TARGET > A[C]
            L = C+1
        end
    end while
    return FAIL
end
```

```
function mcs(A,L,R)
    if L == R:
        return A[L]
    else:
        C = (L+R) // 2
        Lmax = mcs(A,L,C)
        Rmax = mcs(A,C+1,R)
        Smax = maximum straddling mcs
        return max(Lmax,Smax,Rmax)
    end if
end function
```

A → A Just for Reference! No need to copy!

not all today! ☺

RECURRENCE RELATIONS, DIGGING DOWN, RECURRENCE TREES, THE MASTER THEOREM

① Intro: Goal is to develop some new tools for analyzing time complexity. (and really, other things, too)

② Prep: Consider binary search and MCS-Divide + Conquer

Binary Search: At a high (abstract) level we:
Check the middle (constant time)
Then check a list half the size

MCS-DAC: At a high (abstract) level we:
Calc straddling sum (linear time)
Then check two lists half the size

We can think of these as:

Binary Search: $T(n) = T(\frac{n}{2}) + c$

- $T(n)$: time for list of length n
- $T(\frac{n}{2})$: for list of length $n/2$
- c: some constant time

MCS-DAC: $T(n) = 2T(\frac{n}{2}) + Cn$

- $2T(\frac{n}{2})$: two lists of length $n/2$
- Cn: some linear time

③ Recurrence Relations

defn. A recurrence reln is an eqn which gives the value

defn. A recurrence reln is an eqn which gives the value of a fctn in terms of earlier values.
Typically we'll have one (or more) base cases.
Sometimes we'll have floor and ceiling fctns, often not.

ex $T(n) = T(\frac{n}{2}) + c$ with $T(1) = D$ c, D = constants

ex $T(n) = T(\lceil n/2 \rceil) + 3$ with $T(1) = 5$

ex $T(n) = 2T(n/3) + n+1$ with $T(1) = 7$

ex $T(n) = 3T(n/4) + 4T(n/3) + n^2 + n + 2$ with $T(1) = 0$

④ Solving for specific value

We can often solve for specific values!

ex $T(n) = T(\lceil n/2 \rceil) + n + 1$ w/ $T(1) = 2$

then $T(2) = T(\lceil 2/2 \rceil) + 2 + 1 = T(1) + 2 + 1 = 2 + 2 + 1 = 5$

$T(3) = T(\lceil 3/2 \rceil) + 3 + 1 = T(2) + 3 + 1 = 5 + 3 + 1 = 9$

$T(4) = T(\lceil 4/2 \rceil) + 4 + 1 = T(2) + 4 + 1 = 5 + 4 + 1 = 10$

$T(5) = T(\lceil 5/2 \rceil) + 5 + 1 = T(3) + 5 + 1 = 9 + 5 + 1 = 15$

⋮

Fine, but can we find $T(n) = \Theta(???)$?

⑤ Note 1: When computing values like in ④ we need $\lceil\ \rceil$ or $\lfloor\ \rfloor$ to create integers.

Note 2: For a more comprehensive analysis, we want.

⑥ Digging Down:

ex Sps we have $T(n) = T(\frac{n}{2}) + 3$ w/ $T(1) = 2$ ⊛

obs: $T(n) = T(\frac{n}{2}) + 3$ ↤ but $T(\frac{n}{2}) = T(\frac{n/2}{2}) + 3$ ⊛

$= [T(\frac{n}{2^2}) + 3] + 3 = T(\frac{n}{2^2}) + 2(3)$ ↤ but $T(\frac{n}{2^2}) = T(\frac{n/2^2}{2}) + 3$ ⊛

$= T(\frac{n}{2^3}) + 3(3)$

$= T(\frac{n}{2^4}) + 4(3)$

$= T(\frac{n}{2^5}) + 5(3)$

⋮ ↤ pattern?

$= T(\frac{n}{2^k}) + k(3)$ ↤ in general!

Stops when $n/2^k = 1$ b/c $T(1) = 2$, so, note $n = 2^k$ so $k = \lg n$

Stops when $n/2^k = 1$ b/c $T(1) = 2$, so, note $n = 2^k$ so $k = \lg n$

$= T(1) + (\lg n)(3)$

$T(n) = 2 + 3\lg n = \Theta(\lg n)$

Process: Use the R.R to keep rewriting $T(n)$
until we can see a general form w/ a k in it.
Then set inside of $T(*)$ equal to the base value, typically 1.

Solve for k,
Then finally write $T(n)$ as a closed form.

Ex ⊛ $T(n) = 2T(\frac{n}{2}) + 5n$ w/ $T(1) = 7$

we have

$T(n) = 2T(\frac{n}{2}) + 5n$ ← $T(\frac{n}{2}) = 2T(\frac{n/2}{2}) + 5(\frac{n}{2})$ ⊛

$= 2\left[2T(\frac{n}{2^2}) + 5(\frac{n}{2})\right] + 5n$

$= 2^2 T(\frac{n}{2^2}) + 2(5n)$ ← $T(\frac{n}{2^2}) = 2T(\frac{n/2^2}{2}) + 5(\frac{n}{2^2})$ ⊛

$= 2^2\left[2T(\frac{n}{2^3}) + 5(\frac{n}{2^2})\right] + 2(5n)$

$= 2^3 T(\frac{n}{2^3}) + 3(5n)$

$= \dots$

$= 2^k T(\frac{n}{2^k}) + k(5n)$ ← in general

ends when $n/2^k = 1$ so $k = \lg n$ so:

$T(n) = 2^{\lg n} T(1) + \lg n\,(5n)$

$= n(7) + 5n\lg n$

$T(n) = 7n + 5n\lg n = \Theta(n \lg n)$

Note: These can be less straightforward
We'll need some log stuff as we go forward!
Such as, by example:

$2^{\lg n} = n$ → or $= (2^{\lg n})^2 = n^2$

$4^{\lg n} = (2^2)^{\lg n} = 2^{2\lg n} = 2^{\lg(n^2)} = n^2$

$$4^{\lg n} = (2^2)^{\lg n} = 2^{2\lg n} = 2^{\lg(n^2)} = n^2$$

$$3^{\log_4 n} = 3^{\frac{\log_3 n}{\log_3 4}} = \left(3^{\log_3 n}\right)^{\frac{1}{\log_3 4}} = n^{1/\log_3 4} = n^{\log_4 3}$$

↑ COB

↑ COB (sneaky!)