Part 2: Analysis of Recursive Algorithms

- use induction here

1. Complexity of Recursive Algorithms
2. Correctness

Outline

Induction on Recurrences

Complexity of Recursive Algorithms

Recursively Defined Functions

$f(4) \to$ need $f(3) \to$ need $f(2) \to$ need $f(1)$ ✓

need $f(2)$

Define:

$$f(n) = \begin{cases} 2 & n = 0 \\ 7 & n = 1 \\ 2f(\underline{n-2}) + f(\underline{n-1}) & n > 1 \end{cases}$$

Write out a few values of $f(n)$. We want to show $\underbrace{f(n) < 2^{g(n)}}$.
Conjecture for $g(n)$?

n	0	1	2	3	4	5	6
$f(n)$	2	7	11	25	47	97	191

Conjecture: $f(n) < 2^{n+2} \quad \forall n \in \mathbb{N}$

$f(n) < 2^{n+2}$ We'll use complete induction because $f(n)$ depends on more than $f(n-1)$.

Inductive Step: Let $n \in \mathbb{N}$

Assume $H(n)$: $\forall i \in \mathbb{N}$, $0 \leq i < n$, $f(i) < 2^{i+2}$

Show $H(n) \rightarrow C(n)$: $f(n) < 2^{n+2}$

Let $n > 1$.

By definition, $f(n) = 2 \cdot f(n-2) + f(n-1)$

$$< 2 \cdot 2^{(n-2)+2} + 2^{(n-1)+2} \quad \left(\begin{array}{l}\text{by } H(n),\ n>1 \\ 0 \leq n-1, n-2 < n\end{array}\right)$$

$$= 2^{n+1} + 2^{n+1}$$

$$= 2^{n+2}$$

So $C(n)$ holds.

Base case: $n=0$, $n=1$. $2 \leq 2^{0+2}$, $7 \leq 2^{1+2}$

So the claim holds $\forall n \in \mathbb{N}$.

$f(n) < 2^{n+2}$

Reminders about Algorithm Complexity

$T(n)$

- Running time is measured by counting steps in an algorithm
- Sufficient to count chunks of instructions (sequences that are always executed together in constant time) as one step
- Measure as a function $T(n)$ of input "size" n (# of input elements)
- For now, just concerned with worst-case (maximum over all inputs of the same size)
- Often, there is no simple algebraic expression for $T(n)$

asymptotic notation $\longrightarrow$ bounds on $T(n)$

Reminders about Algorithm Complexity

Asymptotic Notation

- Upper bound $T(n) \in O(f(n)) \rightarrow$ if some $c \in \mathbb{R}^+, B \in \mathbb{N}$
  s.t. $T(n) \leq c \cdot f(n), \forall n \in \mathbb{N}, n \geq B$
- Lower bound $T(n) \in \Omega(f(n)) \rightarrow$ if $\exists\, c \in \mathbb{R}^+, B \in \mathbb{N}$
  s.t. $T(n) \geq c \cdot f(n), \forall n \in \mathbb{N}, n \geq B$
- Tight bound $T(n) \in \Theta(f(n)) \rightarrow T(n) \in O(f(n))$
  and $T(n) \in \Omega(f(n))$

Recursive Algorithms

Recursive Binary Search

we should prove
$m - b + 1 = \lceil n/2 \rceil$
$e - (m+1) + 1 = \lfloor n/2 \rfloor$
- I'll post this later

See Notes page at end

$n = e - b + 1$

$T(n)$

```
RecBinSearch(x, A, b, e):
1.    if b == e:
2.        if x <= A[b]:     return b
3.        else:     return e + 1
      else:
4.        m = (b + e) // 2   # midpoint
5.        if x <= A[m]:
6.            return RecBinSearch(x, A, b, m)
          else:
7.            return RecBinSearch(x, A, m+1, e)
```

recursive calls.

$T(\lceil n/2 \rceil)$

$T(\lfloor n/2 \rfloor)$

Complexity of RecBinSearch

We can represent $T(n)$ of RecBinSearch as the recurrence

$$T(n) = \begin{cases} 1 \ (\text{really } \Theta(1)) & n = 1 \\ 1 + \max(T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor)) & \end{cases}$$

How do we solve $T(n)$ to get a closed form representation to move towards finding a bound?

We'll use Repeated Substitution/unwinding to get a guess.

Solving Recurrence Relations

Example: Recursive Factorial

```
Fact(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * Fact(n-1)
```

Worst case running time

$$T(n) = \begin{cases} 1 & \text{if } n=0 \text{ or } n=1 \\ 1 + T(n-1) & \text{if } n>1 \end{cases}$$

what's the closed form?

Solving Recurrence Relations

Unwinding $T(n)$

$$T(n) = \begin{cases} 1 \leftarrow \Theta(1) & n=0 \text{ or } n=1 \\ 1 + T(n-1) & n>1 \end{cases}$$

$$T(n) = 1 + \underbrace{T(n-1)} = 1 + 1 + \underbrace{T(n-2)}$$

$$= \underbrace{1 + 1 + 1} + T(n-3)$$

Pattern: after i substitutions,

$$T(n) = i + T(n-i)$$

$$T(n) = (n-1) + T(n-\overset{=1}{(n-1)}) = n-1+1 = n$$

So our guess is $T(n) = n$.

We need to prove it.

Solving Recurrence Relations

Proving $T(n) = n$ by Induction

$$T(n) = \begin{cases} 1 & \text{if } n=0 \text{ or } n=1 \\ \underline{\underline{1+T(n-1)}} & \text{if } n>1 \end{cases}$$

Inductive Step: Let $n \in \mathbb{N}, n \geq 1$

Assume $H(n)$: $T(k) = k$, $1 \leq k < n$, $k \in \mathbb{N}$

Show $H(n) \rightarrow C(n)$: $T(n) = n$

$\underline{\text{Let } n > 1}$, $T(n) = 1 + T(n-1)$ (by definition)

$= 1 + (n-1)$ (by $H(n)$, because $n>1$, $1 \leq n-1 < n$)

$= n$.

Conclude $C(n)$.

Base case: $T(1) = 1$ ✓

Conclude $T(n) = n$ $\forall n \in \mathbb{N}, n \geq 1$

Recursive Binary Search

Unwinding $T(n)$

$$T(n) = \begin{cases} 1 & n = 1 \\ 1 + \max(T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor)) & n > 1 \end{cases}$$

Make some simplifying assumptions

Suppose $n = 2^k$, $k \in \mathbb{N}$.

Then $T(n) = \begin{cases} 1 & n = 1 \\ 1 + T(n/2) & n > 1 \end{cases}$ (because we assumed n is even)

$$T(n) = T(2^k) = 1 + \underbrace{T(2^{k-1})} = 1 + 1 + T(2^{k-2})$$

$$\dots$$

$$= \underbrace{1 + \dots + 1}_{k} + T(2^{k-k})$$

what's k in terms of n?

$$= k + 1 = \underline{\lg} n + 1 \Rightarrow \log_2 n$$

$\rightarrow$ $T(n)$ can't be $\lg n + 1$ for all n, b/c $T(n) \in \mathbb{N}$, and we made simplifications

<u>Conjecture</u>: $T(n) \in \Theta(\underline{\lg n})$

Recursive Binary Search

Lower bound on $T(n)$

To prove $T(n) \in \Omega(\lg n)$, need $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}$ s.t. $T(n) \geq c \cdot \lg n,\ n \geq B$

Pick $B = 2$, $c = 1$, prove by Complete Induction.

Inductive Step: Let $n \in \mathbb{N}$, $\underline{n \geq B}$

Assume $H(n)$: $T(i) \geq c \cdot \lg(i)$, $\forall i\ B \leq i < n$

Show $H(n) \to C(n)$: $T(n) \geq c \cdot \lg(n)$

$T(n) = 1 + T(\lceil n/2 \rceil)$ (because T is monotonically incr.) $\to T(\lceil n/2 \rceil) \geq T(\lfloor n/2 \rfloor)$

$\geq 1 + c \cdot \lg(\lceil n/2 \rceil)$ (because $n \geq B > 1 \to n \geq 2$, $B \leq \lceil n/2 \rceil < n$ by $H(n)$)

$\geq 1 + c \cdot \lg(n/2)$ (because $\lg$ is mon. incr.)

$= 1 + c(\lg(n) - \lg(2))$ (log identity)

$= c \cdot \lg n + 1 - c^{=1}$

$= c \cdot \lg(n)$

So $C(n)$ holds.

Base case ...

$T(n) \in \Omega(\lg n)$

Recursive Binary Search

To prove $T(n) \in O(\lg n)$,
need $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, B \leq n \Rightarrow$
$T(n) \leq c \cdot \lg n$

Upper bound on $T(n)$

$B = ?$ $c = ?$

$$T(n) = 1 + T(\lceil n/2 \rceil)$$
$$\leq 1 + c \cdot \lg(\underline{\lceil n/2 \rceil})$$
$$\leq 1 + c \cdot \lg\left(\frac{n+1}{2}\right)$$
$$= 1 + c \cdot (\lg(n+1) - \lg(2))$$
$$= \underline{c \cdot \lg(n+1)} + 1 - c$$

(want to say that by H(n))
what is $\lceil n/2 \rceil$ less than? $\leq \frac{(n+1)}{2}$

→ problem, we want $c \cdot \lg(n)$
need to get rid of $+1$

$\lceil n/2 \rceil \leq \frac{n+1}{2}$ so $\lceil n/2 \rceil - 1 \leq \frac{n+1}{2} - 1 = \frac{n-1}{2}$

Change our proof to show $T(n) \leq c \cdot \lg(n-1)$ $(\leq c \cdot \lg(n))$

$$T(n) = 1 + T(\lceil n/2 \rceil) \leq 1 + c \cdot \lg(\lceil n/2 \rceil - 1) + 2 \leq 1 + c \cdot \lg\left(\frac{n-1}{2}\right) + 2$$
$$\leq 1 + c(\lg(n-1) - \lg(2)) + 2 = c \cdot \lg(n-1) + \underline{1 - c} + 2$$
$$= c \cdot \lg(n-1) + 2$$

Pick $c = 1$

Recursive Binary Search

Upper bound on $T(n)$

Base case: $T(1) = 1 \le \lg(1-1) = \lg(0) \rightarrow$ not defined.

So $B > 1$

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

$\lg(2-1) = \lg(1) = 0$

$2 \not\le \underline{0}$

What if we add 2?

$T(n) \le \lg(n-1) + 2$?

in $H(n)$,

$\forall i, B \le i < n$

$n > 2$

$\lceil n/2 \rceil \geq B$

This would solve all our problems!

Recursive Binary Search

Upper bound on $T(n)$

Notes

from pseudocode

$$m-b+1 = \left\lfloor \frac{e+b}{2} \right\rfloor - b + 1 \qquad \left(\lfloor x \rfloor + k = \lfloor x+k \rfloor \text{ if } k \in \mathbb{Z} \right)$$

$$= \left\lfloor \frac{e+b-2b+2}{2} \right\rfloor = \left\lfloor \frac{e-b+1+1}{2} \right\rfloor = \left\lceil \frac{e-b+1}{2} \right\rceil \qquad \left(\left\lfloor \frac{k+1}{2} \right\rfloor = \left\lceil \frac{k}{2} \right\rceil \ \forall k \in \mathbb{N} \right)$$

$$= \left\lceil n/2 \right\rceil$$

$$=$$

$$e-m = e - \left\lfloor \frac{e+b}{2} \right\rfloor = e + \left\lceil \frac{-e+b}{2} \right\rceil \qquad \left(-\lfloor x \rfloor = \lceil -x \rceil \right)$$

$$= \left\lceil e - \frac{e+b}{2} \right\rceil \qquad \left(\lceil k+x \rceil = \lceil x \rceil + k, \ k \in \mathbb{Z} \right)$$

$$= \left\lceil \frac{e-b}{2} \right\rceil = \left\lfloor \frac{e-b+1}{2} \right\rfloor \qquad \left(\left\lfloor \frac{k+1}{2} \right\rfloor = \left\lceil \frac{k}{2} \right\rceil, \ k \in \mathbb{Z} \right)$$

$$= \left\lfloor n/2 \right\rfloor$$

(n is the # of elements to search
i.e. $e-b+1$)