Lecture 8 Jan 26

Induction proofs – part II

Geometric series: $1 + a + a^2 + a^3 + \dots + a^n$

claim: $= \dfrac{1-a^{n+1}}{1-a}$ for all $n \geq 0$ integer
for all $a > 0$ real, $a \neq 1$

Example: $a = 3$, $n = 4$

$1 + 3^1 + 3^2 + 3^4 + 3^5 = 1 + 3 + 9 + 27 + 81 = 121$

$$\frac{1-3^{4+1}}{1-3} = \frac{1-3^5}{1-3} = \frac{1-243}{1-3} = \frac{-242}{-2} = 121$$

Proof: $\boxed{1 + a + a^2 + \dots + a^n = \dfrac{1-a^{n+1}}{1-a}} \to P(n)$

Base case: $n = 0$: LHS $= 1$
RHS $= \dfrac{1-a}{1-a} = 1$

Induction step: Assume $P(k)$ is true, i.e
$1 + a + \dots + a^k = \dfrac{1-a^{k+1}}{1-a}$

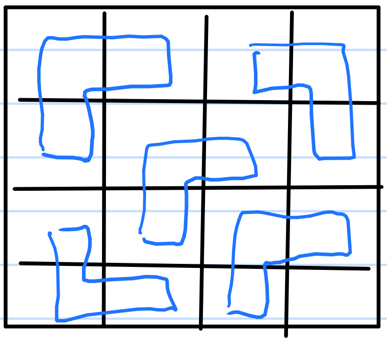
Goal: Show that $P(k+1)$ is true i.e

$$1 + a + \dots + a^{k+1} = \frac{1-a^{(k+1)+1}}{1-a}$$

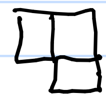
$$\underbrace{1 + a + a^2 + \dots + a^k}_{\frac{1-a^{k+1}}{1-a} \text{ by IH}} + a^{k+1} = \frac{1-a^{k+1}}{1-a} + a^{k+1} = \frac{1-a^{k+1}}{1-a} + \frac{a^{k+1} - a^{k+2}}{1-a}$$

$$= \frac{1-a^{k+2}}{1-a}$$

Tiling problem:

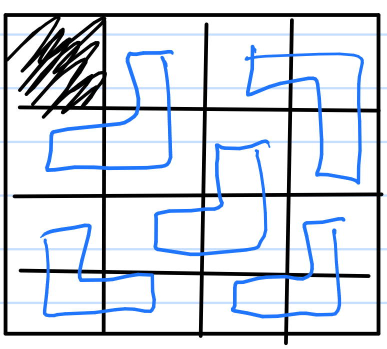


This board cannot be completely covered by trominoes

Goal: cover board with trominoes: 

Consider a $2^n \times 2^n$ board where one square is already covered.

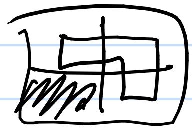
Goal: cover the rest with trominoes



Want to prove this is always possible

Proof by induction:

$P(n)$ Any $2^n \times 2^n$ board with one square already can be tiled with trominoes, for $n \geq 1$

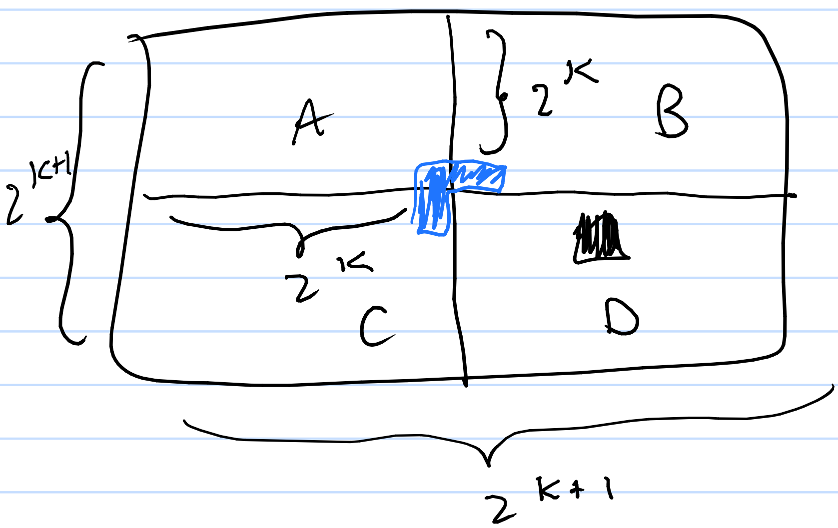
Base case: $n=1$ 

No matter where covered, can always place a tromino

(2) Induction: Assume $\underbrace{P(K)}$ is true

$\hookrightarrow$ $2^K \times 2^K$ board with one sq - cover can be tiled

Goal: show $P(K+1)$ is true



(1) sub-board D has one square covered and is of size $2^K \times 2^K$

$\Rightarrow$ Because of induction hypothesis, D can be tiled

(2) Place tromino in center, to cover one square of A, B & C

Now A, B & C all have one square covered and they are each of size $2^K \times 2^K$

$\Rightarrow$ Because of IH, the rest of A, B & C can be tiled with trominoes

Bad proof:

Prove: $P(n)$ $\boxed{n+1 < n}$ for all n

Induction proof:

Assume $P(k)$ is true, i.e. $k+1 < k$

Goal: Show that $P(k+1)$ is true i.e. $(k+1)+1 < k+1$

$\underbrace{(k+1)}_{<k}+1 < k+1$ (because of IH)

Problem: No base case

Like toppling over dominoes, need to knock down first one

Claim: $8^n - 3^n$ is divisible by 5 for any $n \geq 1$

Ex: if $n=2$, $8^2 - 3^2 = 64 - 9 = 55$

Proof: Base case: For $n=1$, $8^1 - 3^1 = 5$

Induction step: Assume that $\underbrace{8^k - 3^k \text{ is divisible by } 5}_{8^k - 3^k = 5 \cdot a}$

Goal: $8^{k+1} - 3^{k+1}$ is divisible by 5

$8^{k+1} - 3^{k+1} = 8^{k+1} - 8 \cdot 3^{k+1} + 5 \cdot 3^{k+1}$

Lecture 9 Jan 27

Loop invariant proofs Jan 27 2017

Algo FindMin (A, n)

Input: Array A of n integers
Output: Return the smallest element in A

$i \leftarrow 1$
$m \leftarrow A[0]$
while $(i < n)$
  if $(A[i] < m)$ then $m \leftarrow A[i]$
  $i \leftarrow i + 1$
return m

Loop invariant : $\boxed{\text{At iteration } i,\ m = \min\{A[0], \dots, A[i-1]\}}$

Goal: prove the loop invariant holds

(1) Initialization : Before the start of the loop
LI holds
$i = 1, \quad m = A[0] = \min\{A[0], \dots, A[\underbrace{i-1}_{0}]\}$

Analogous to proving base case

(2) Maintenance : [Assume L.I. holds at beginning of an iteration of loop.
we must show that L.I. holds at the end of that iteration

Assume $m = \min\{A[0], \dots, A[i-1]\}$

Two conditions

If $(A[i] < m)$, then replacing m with $A[i]$ results in m being $\min\{A[0], \dots, A[i]\}$

If $(A[i] \geq m)$, then m remains unchanged, m is now $\min\{A[0], \dots, A[i]\}$

After increasing i by one: $\boxed{m = \min\{A[0] \dots A[i-1]\}}$

L. I.

(3) Termination: 3.1. Algo will stop because counter variable i gets increased by one at each iteration. So, it will eventually reach n.

3.2 When loop terminates
$m = \min\{A[0], \dots, A[n-1]\}$

Loop stops when $i = n$
Loop invariant says $m = \min\{A[0], \dots, A[i-1]\}$
$\Longleftrightarrow m = \min\{A[0], \dots, A[n-1]\}$

Lecture 10 Jan 31

Want to see running time growth w/o implementing pseudo

Primitive operations → Running time independent of other things

FindMin analysis

Read each line, see how much "time" each one takes

while condition is a conditional

Best case: Want to not execute optional things

if cond false most/all of the

→ smallest found at A[start]

Worst case: Array is sorted in decreasing order

if is always true

Measuring running time:

How many times will loop be exec: stop−start

This is for worst case:

Out of the loop:

$T_{index} + 3T_{assign} + T_{arith} + T_{comp} + T_{cond} + T_{return}$

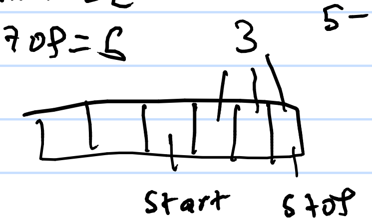
Loop:

$(stop-start) \cdot (2T_{comp} + 2T_{cond} + 2T_{index} + 3T_{assign} + T_{arith})$

stop−start because index starts at start+1

ex. start=2
stop=5

5−2=3



Assumption:

All primitive operations take around the same time

Makes it easier for us
→ Need # of primitive operations

Back to worst case, counting primitive ops:

$8 + (stop - start) \cdot 10$

linear in size of array
intercept is 8
slope 10

Selection sort: simpler version than what we saw before, using findMin from before

minIndex → 3 prim op, subtraction, call, assign

$3 + T_{findMin}(i, n-1) = 3+8+10(n-1-i) = 3+8+10n-10-i$
$= 3+(10n-2)-i$

As loop executes, array gets smaller (for findMin), so running time keeps decreasing

$\underbrace{1+2}_{\text{outside loop}} + \underbrace{(22+10(n-1-0))}_{\text{iteration 0}} + (22+10(n-1-1)) + \ldots + (22+10(n-1-(n-1)))$

Doing iteration i: $20 + 10 \cdot (n-1-i)$

$$= 3 + \sum_{i=0}^{n-1} 22 + 10(n-1-i) = 3 + \sum_{i=0}^{n-1} 10n + 12 - 10i$$

goes n times

$$= 3 + n(10n+12) + \sum_{i=0}^{n-1} -10i = 3 + 10n^2 + 12n - 10 \cdot \boxed{\sum_{i=0}^{n-1} i}$$

$$\boxed{\sum_{i=1}^{n} i = \frac{n(n+1)}{2}}$$

$$= 3 + 10n^2 + 12 - 10 \cdot \frac{(n-1)(n-1+1)}{2}$$
$$= 3 + 10n^2 + 12n - \frac{10(n-1)n}{2}$$

$= \boxed{5n^2 + 17n + 3}$

Runtime as func

Lecture 13 Feb 7 2017

Zheng Dai

zheng.dai@mail.mcgill.ca

Today -> Analyzing runtime of recursive algorithms

Factorial (int n)

$T(n)$

if(n==1) — 2

return n; — 1

$T(n-1)$

else — 0

return n * Factorial(n-1) — ? + 4

can't just plug in runtime of method, because we're counting that now

n	$T(n)$
1	3
2	$2+4+T(1)=2+3+3=9$
3	$2+4+T(2)=15$
4	$2+4+T(3)=21$

$6(n-1)+3$

$6n-3$

$$T(n) = \begin{cases} 3 & \text{when } n=1 \\ T(n-1)+2+4 & \text{when } n>1 \end{cases}$$

Want an explicit formula

or else, if you wanted $T(100)$, you'd need $T(99), T(98), \ldots$

Substitution Approach

We want to get to $T(1)$

(1) $T(n) = \boxed{T(n-1)} + 6$

$\boxed{T(n-1)} = T((n-1)-1)+6 = T(n-2)+6$

(2) $T(n) = (T(n-2)) + 6) + 6 = T(n-2) + 12$

$T(n-2) = T(n-3) + 6$

(3) $T(n) = (T(n-3) + 6) + 12 = T(n-3) + 18$

...

(K) $T(n) = T(n-K) + 6K$ ⇐ guess

$T(n-K) = T(1)$

$n-K = 1$

$K = n-1$

$T(n) = T(n-(n-1)) + 6(n-1)$

$T(n) = T(1) + 6(n-1)$

$T(n) = 6n - 6 + 3 = \underbrace{6n-3}$

check this with table

1	$6 - 3 = 3$ ✓
2	$12 - 3 = 9$ ✓
3	$18 - 3 = 15$ ✓
4	$24 - 3 = 21$ ✓

$T(n) = 6n-3 \rightarrow O(n)$

Binary search (A, start, stop, key) $n = stop - start + 1$

```
if (start == stop)                       — 2
    if (A[start] == key)                 — 3
        return start;                    — 1
    else
        return 1                         — 1
else
    mid = ⌊(start+stop)/2⌋               — 3

    if (A[mid] <= key)                   — 3
        r = BinarySearch(A, start, mid, key);    — 1 + T(⌊n/2⌋)
    else
        r = BinarySearch(A, mid+1, stop, key);   — 2 +
    return r;                            — 1
```

ignore meth call act

$\begin{cases} T(\frac{n}{2}) & \text{when } n \text{ even} \\ T(\lfloor\frac{n}{2}\rfloor)+1 & \text{when } n \text{ odd} \end{cases}$

study worst case

$$T(n) = \begin{cases} 6 & \text{if } n=1 \\ 2+9+T(\frac{n}{2}) & \text{if } n \text{ even} \\ 2+9+T(\lfloor\frac{n}{2}\rfloor+1) & \text{if } n \text{ odd} \end{cases}$$

n	1	2	~~3~~	4	~~5~~	~~6~~	~~7~~	8
T(n)	6	17	~~28~~	28	~~39~~	~~39~~	~~39~~	39

Don't care about $n \neq 2^x$
why? exercise

1   2   4   8   16   32

$$T(n) = \begin{cases} 6 & \text{if } n=1 \\ 11 + T(\frac{n}{2}) & \text{otherwise} \end{cases}$$

(1) $11 + T(\frac{n}{2})$ want to get $T(1)$

$T(\frac{n}{2}) = 11 + T(\frac{n}{4})$

(2) $T(n) = 11 + (11 + T(\frac{n}{4})) = 22 + T(\frac{n}{4})$

$T(\frac{n}{4}) = 11 + T(\frac{n}{8})$

(3) $T(n) = 22 + (11 + T(\frac{n}{8})) = 33 + T(\frac{n}{8})$

(K) $T(n) = 11k + T(\frac{n}{2^k})$ ← guess

Bese case → $\frac{n}{2^k} = 1 \rightarrow 2^k = n \rightarrow \log_2 n = K$

$T(n) = 11(\log_2 n) + T(1) = 11(\log_2 n) + 6$

$11(\log_2(1)) + 6 = 6$

$11(\log_2(2)) + 6 = 17$

$11(\log_2(4)) + 6 = 28$

$11(\log_2(8)) + 6 = 39$

$O(\log n)$

MergeSort (A, start, stop)

if (start == stop) — 2

return; — 1

else

mid = ⌊(start + stop)/2⌋; — 3

MergeSort (A, start, mid); — 0 + $T(\frac{n}{2})$

MergeSort (A, mid+1, stop); — 1 + $T(\frac{n}{2})$

Merge (A, start, mid, stop); — ~9n

$$T(n) = \begin{cases} 3 & \text{in base} \leftarrow n = 1 \\ 6 + 9n + 2T(\frac{n}{2}) & \text{otherwise} \end{cases}$$

(1) $T(n) = 6 + 9n + 2T(\frac{n}{2})$

(2) $T(n) = 6 + 9n + 2(6 + 9(\frac{n}{2}) + 2T(\frac{n}{4}))$

$= (1+2)6 + 9(n+n) + 4T(\frac{n}{4})$

(3) $T(n) = (1+2)6 + 9(n+n) + 4(6 + \frac{9n}{4} + 2T(\frac{n}{8}))$

$= (1+2+4)6 + 9(n+n+n) + 8(T(\frac{n}{8}))$

$\vdots$

(K) $T(K) = \left(\sum_{i=0}^{K-1} 2^i\right)6 + 9(Kn) + 2^K(T(\frac{n}{2^K}))$

$\left(\frac{1-2^K}{1-2}\right)6 + 9(Kn) + 2^K(T(\frac{n}{2^K}))$

$= (2^K - 1)6 + 9(Kn) + 2^K(T(\frac{n}{2^K}))$

$2^K = n \rightarrow K = \log_2 n$

$= (n-1)6 + 9(\log_2 n)(n) + n(3)$

$O(n \log n)$