DESIGN & ANALYSIS OF ALGORITHMS

SEP 20

GOWTHAM
W1607812

Hien Vu (9:00 to 10:30 AM - Monday)

6 chapters from book (CLRS)

Any language for assignments/tests

Classes uses pseudocode (recommended for us as well)

HW Deadline → Sunday 11.59pm

SEARCHING PSEUDOCODE

As a programmer don't have 2 returns (1 exit point)

```
Search(A,x,n)
    i=0                     — 1
    while (i<=n-1)          — n+1
        if A[i]==x          — n
            return True     — 1
        i=i+1               — n
    return False            — 1
```

PROVING

* You can prove by proving line by line
* LOOP INVARIANT
    ↳ next page
* If invariant true at beginning, it will be true for next itr

$3n+1 = O(n)$

↳ in our class prove

PROVING

1) Simple steps (small) - prove by line by line

2) LOOP - Loop Invariant

both are same

3) RECURSION - Mathematical Induction

MATHEMATICAL INDUCTION

1) Prove for base case (initial value)

2) Assume statement is true when $n=k$

3) Use that to prove for $n=k+1$

↳ use $n=k$, take it as true and prove other part

LOOP INVARIANT

1) found is false

2) Y is not in $A[0 \ldots i-1]$

INITIALISATION:

found is false

Y is not in $[\,]$

MAINTENANCE:

If $A[0] \neq Y$

found = false

If $A[0] == Y$

TERMINATION:

① @ $i=N$

Y is not in $A[0 \ldots n-1]$

② found == true

Y is in $A[0 \ldots i-1]$

TAKE HOME STUFFS

1. ~~len() operator - TC in Python?~~ - $O(1)$
2. Review Mathematical Induction (1 que in Test)
3. Loop Invariant?
4. Be prepared to calculate precise R.T. on Wednesday
5. How to find the correct loop invariant condition?

HOME LEARNINGS

ASSERTION: Statement that is either True or False (bools)
used in programming as a condition
Ex: assert $(x >= 5)$

PRE-CONDITION: Requires a boolean condition satisfied
before start of the function

POST-CONDITION: Requires boolean condition met at the
end of the function

We need to assume in loops that

* it terminates
* it produces the desired output

In order to get there, we use a concept called LOOP INVARIANT

LOOP INVARIANT: A boolean condition that is true
at the beginning of the loop and at
the end of each iteration.

WHY STUDY LOOP INVARIANTS

* To prove properties of the loops
* To prove partial correctness of the loops

HOW TO FIND A LOOPING INVARIANT?

* Prove condn is true in the beginning before looping
* Prove condn is true at each iteration (one iteration)
* Prove loop termination condition.

PROVING CORRECTNESS

1) For simple programs → line by line prove

2) For loops with numbers → unwind & prove

3) For loops with variables → loop invariant

4) For programs with recursion → Math Induction

↓ used for loops as well

LECTURE 2

SEP 22

Find max value of a list

findMax(A, n):

1 — max = A[0]

n — for (i = 1 ... n-1)

n-1 —— if A[i] > max

$0 \le f(n) \le n$ —— max = A[i]

1 — return max

LOOP INVARIANT:

max contains largest value in A[0...i-1]

TERMINATION:

max contains largest value in A[0...n-1]

TC: $2n + f(n) + 1$

$3n+1 = O(n)$ — W.C

$2n+1 = \Omega(n)$ — B.C

$\therefore TC = \Theta(n)$

$O, \Omega, \Theta \Rightarrow$ we are talking about the asymptotic (largest values)

(2) Find if all elements are distinct

(A) isDistinct(A, n)

sort(A)

for i=0 to n-2

if A[i] = A[i+1]

return false

return true

* When you see a loop, try finding loop invariant
(HINT: Dry run in mind is closely associated to finding L.I)

* LOOP INV: A[0...i-1] are distinct

TC: $O(n \log n)$

(B) isDistinct(A, n)

COUNTSORT $O(n)$ soln (for integers)

↳ best solution

create count[0...max]

for i=0 ... max

count[i] = 0

— L.I: Count[i] is 0 for all A[0...i-1]

for j=0 ... n-1

— L.I:

count[A[j]] += 1

— L.I: count[A[i]] = 1 for all A[0...i-1]

for i=1 .... max

if count[i] > 1 return False

return True

TC: $O(max + n)$ → FASTER

③ Find Max using recursion

(A) findMax (A, n, i) — T(n)

if i = n-1 — 1

  return A[i] — 1

m = findMax(A, n, i+1) — T(n-1)

if A[i] > m — 1

  m = A[i] — 1

return m — 1

Rec.Rel ⇒ $T(n) = T(n-1) + 4$

METHODS:

① Back sub

② Rec Tree

③ Master's theorem

$TC = O(n)$

USING MERGESORT TECHNIQUE

(B) findMax (A, n, i, j)

if i = j — 1

  return A[i] — 1

m = $\lfloor (i+j)/2 \rfloor$ — 1

mleft = findMax(A, n, i, m) - $n/2$

mright = findmax(A, n, m+1, j) - $n/2$

if mleft > mright then — 1

  return mleft — 1

else return mright — 1

Rec.Rel ⇒ $T(n) = 2T(n/2)$

$TC = O(n)$