Lecture 4 - Wednesday, January 18

Announcements

- Assignment 1 to be released next Monday
  + Background Study: Basic Recursion
  + Background Study: Call by Value
  + Look ahead: WrittenTest1

Lecture

Asymptotic Analysis of Algorithms

Counting Primitive Operations

Accessing an objects attribute

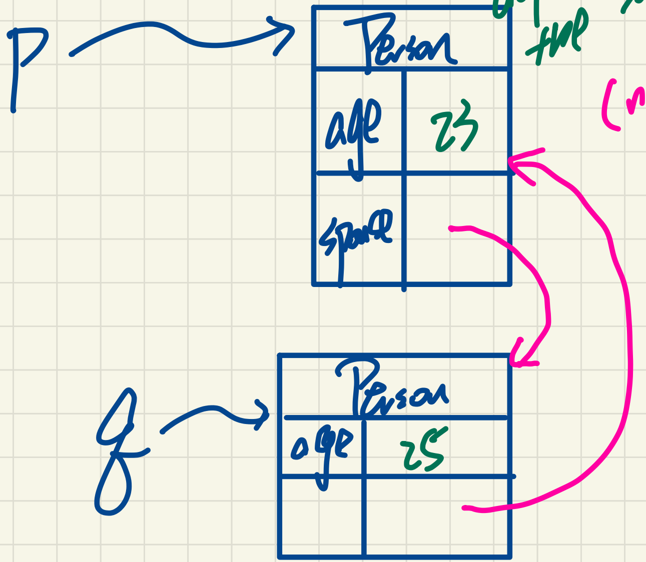
Person P = new Person(...);

In practice, the # of "dots" used to inquire some att. value does not depend on the input size. (no looping).

✓ P.age

look up the address stored in P ⇒ constant (PO)

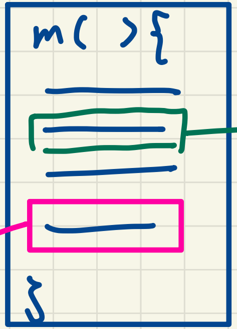
P.spouse.age → multiple lookups ⇒ PO



Method Call

obj . $\underline{\underline{m}}$ (. . .)

Case 1 : m contains POs only



each line corresponds to a PO

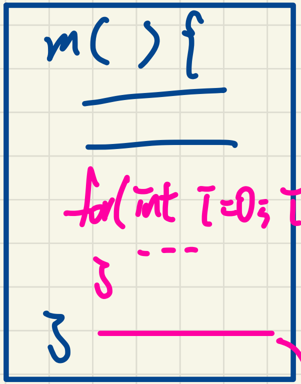
can be:

(1) a method call

(2) a loop

may still be PO

Case 2 : m contains some non-PO

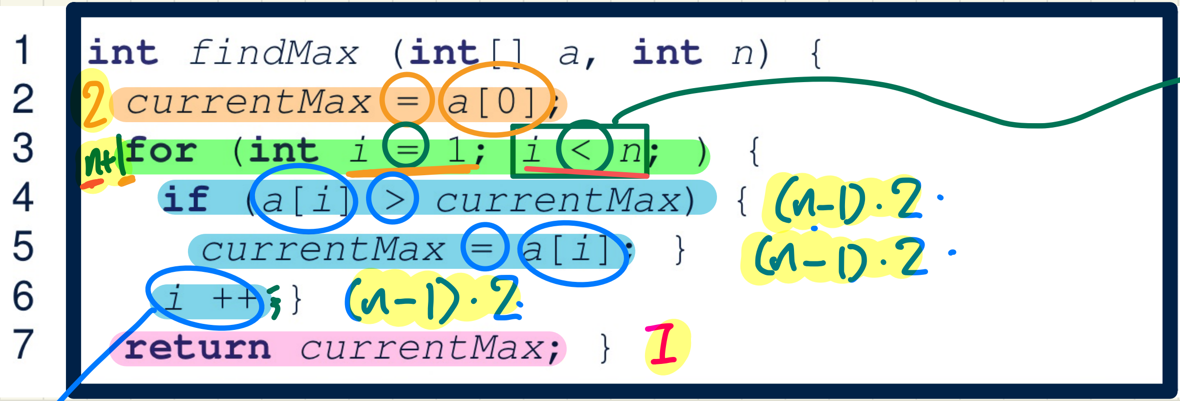


size of some input array

a method call containing non-PO

e.g. int a[] = {2, 3, 4, 5} → findMax(a, a.length)

Example 1: Counting Number of Primitive Operations



```
1  int findMax (int[] a, int n) {
2    currentMax = a[0];
3    for (int i = 1; i < n; ) {
4      if (a[i] > currentMax) {
5        currentMax = a[i]; }
6      i ++; }
7    return currentMax; }
```

2

n+1

$(n-1) \cdot 2$

$(n-1) \cdot 2$

$(n-1) \cdot 2$

1

$i = i + 1$

*

i	i<n
1	T
2	T
⋮	⋮
n-1	T
n	F

Q. # of times i < n in Line 3 is executed?

i<n T $\underline{(n-1)} + 1$ i<n F

Q. # of times loop body (Lines 4 to 6) is executed?

$n-1$ i<n T

$2 + (n+1) + (n-1) \cdot 6 + 1$

$= ?$

stay condition

progress

for (I ; S.C. ; P) {

executed the 1st time, if any, at the end of 1st iteration

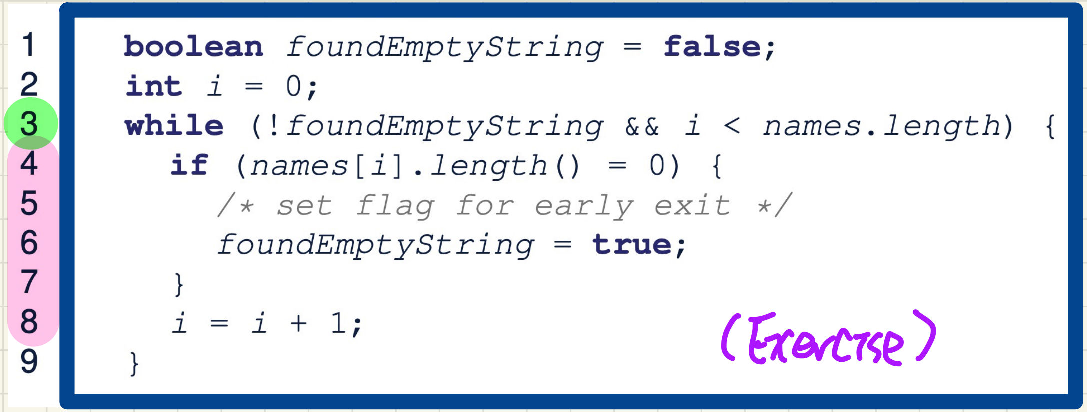
}

e.g. for (; ______ ;) {

⋮

}

Example 2: Counting Number of Primitive Operations



```
1   boolean foundEmptyString = false;
2   int i = 0;
3   while (!foundEmptyString && i < names.length) {
4     if (names[i].length() = 0) {
5       /* set flag for early exit */
6       foundEmptyString = true;
7     }
8     i = i + 1;
9   }
```

(Exercise)

Q. # of times Line 3 is executed?

Q. # of times loop body (Lines 4 to 8) is executed?

Q. # of POs in the loop body (Lines 4 to 8)?

From Absolute RT to Relative RT

t

↳ exact time taken to execute a PO

e.g. Mac M1 2ms

e.g. Mac'14 4ms

findMax contains $7\underbrace{n}_{\text{size of input array}}-2$ POs.

alg.

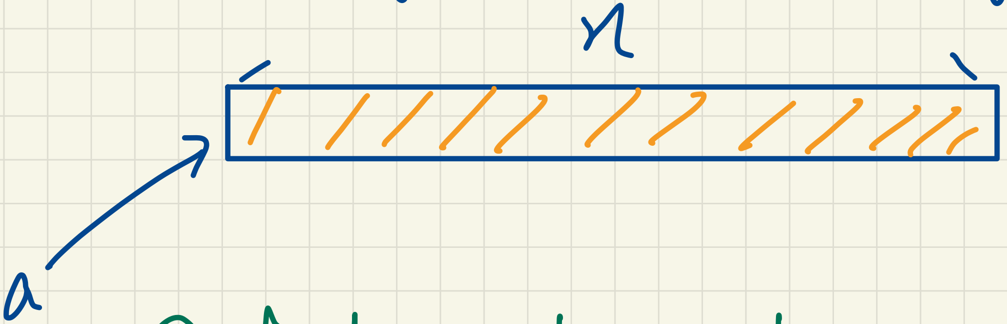
e.g. #PO $\boxed{\leq 2}$ → constant despite input size

#PO $\boxed{2n}$ → linear on the input size

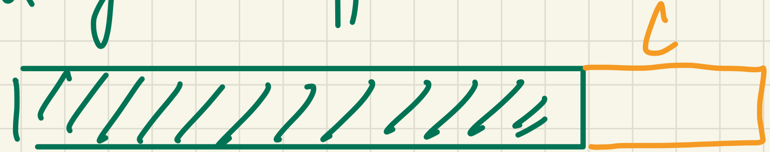
Algorithm 1 $\boxed{(\underbrace{7n-2}_{\#\text{POs}})\cdot t}$ abs. time

Algorithm 2 $\boxed{(\underbrace{10n+3}_{\#\text{POs}})\cdot t}$

Keep adding elements to array



① fixed growth approach



amortized analysis

② doubling