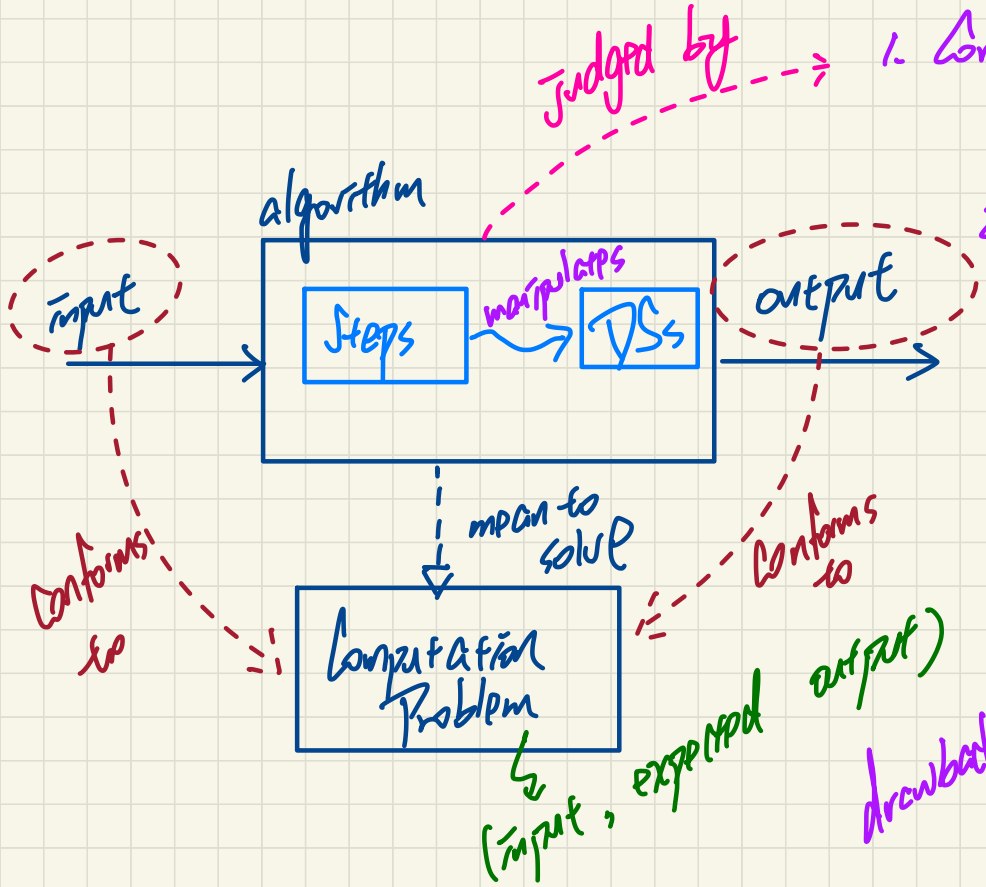
Lecture 4 - January 16

Asymptotic Analysis of Algorithms

Limitations of Experiments
Primitive Operations (POs)
Counting POs: findMax

Announcements/Reminders

- **Assignment 1** released
- Office Hours: 3pm to 4pm, Mon/Tue/Wed/Thu
- Contact Information of TAs on common eClass site
- ***splitArrayHarder***: an **extended** version coming soon



1. Correctness → Testing (dynamic)
   → Formal (static) (3342, 4315)
2. Efficiency
   → Time → Experiments
   → Math Funct (big-O)
   Space → may impact RT

drawbacks

1. full implementation (working)
2. exec. environment
3. input representativeness

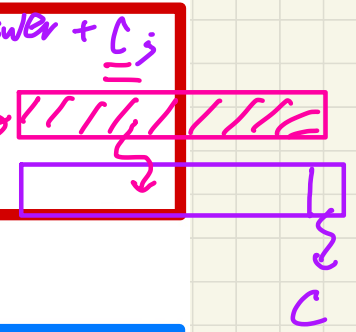
Example Experiment

Computational Problem:

- **Input**: A character *c* and an integer *n*
- **Output**: A string consisting of *n* repetitions of character *c*
  e.g., Given input `'*'` and `15`, output `***************`.

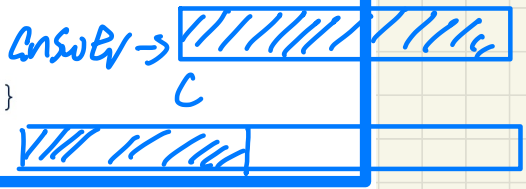
Algorithm 1 using *String* Concatenations:

```
public static String repeat1(char c, int n) {
  String answer = "";
  for (int i = 0; i < n; i ++) { answer += c; }
  return answer; }
```



Algorithm 2 using *StringBuilder* `append`'s:

```
public static String repeat2(char c, int n) {
  StringBuilder sb = new StringBuilder();
  for (int i = 0; i < n; i ++) { sb.append(c); }
  return sb.toString(); }
```



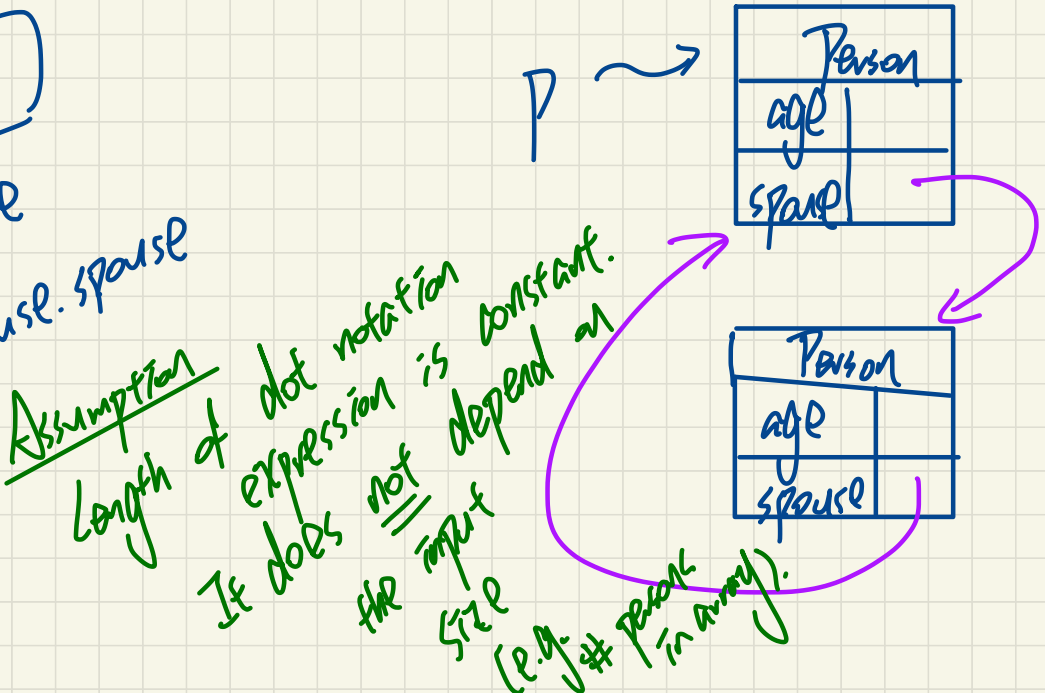
Accessing an object's attribute

```
Person p = new Person(...);
```

p.age

p.spouse

p.spouse.spouse



Assumption

Length of dot notation expression is constant.

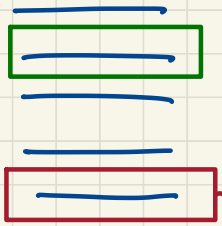
If does not depend on the input size (e.g. # person in array).

Method Call : may or may not be a PO

obj.m(...)

Case 1 : m considered primitive

m() {



a PO

}

can be (1) a method call (that's considered a PO) (2) a loop

Case 2 m not primitive

m() {

for(int i=0; i<n; i++){ ...

}

}

size of input array

findMax(a, a.length)

i<n && i>2 → 2~n.

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; }
```

i	incl. i<n
1	true
2	true
⋮	
n-1	true
n	false

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

n times (n-1 times i<n → true ; 1 time i<n → false)

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

n-1 times (when i<n evaluates to T).