Topic 2: Algorithm Analysis

By: Professor Lynam

What are some algorithms you know already?

Desirable Algorithm Characteristics

- A good algorithm…
- Produces the correct answer for all legal inputs
- Solves the problem efficiently (time & storage)
- Has reproducible output (it’s not random)
- Doesn’t enter infinite loops!
- Etc.

Problem Size

- Definition: *Instance Characteristic*
- *The instance characteristic(s) of a problem is/are the size(s) of the problem*

- For example:
- What would the instance characteristic be for sorting a list?
  - *n (the number of elements in the list)*
- How about for an in-order traversal of a binary tree?
  - The number of nodes in the tree
- Squaring a 2D matrix?
  - The number of rows *and* the number of columns! (2 instance characteristics)

Algorithm Speed

- Why bother investigating the speed of an algorithm?
  - To compare its speed to that of other algorithms
- There are two primary approaches to measuring algorithm speed:

1. Execute and time the algorithm
   - Problems with this approach?
   - Have to code, test, and debug the algorithm before it's ready to be timed!
   - Also, we're measuring wall-clock time, but the algorithm is running on CPU time, which is also running the OS, other programs, etc…
2. Count the operations the algorithm performs
   - Tedious, and possibly not useful (assumes each operation takes the same amount of time)… But can help us find a more general answer for algorithm efficiency

Step-Counting/Operation Counting

1. Augment the algorithm with operation counts
2. Estimate executions of selections & iterations
3. Remove/Ignore the algorithm's statements
4. Produce a step-count expression in terms of the algorithm's instance characteristics

- What do we end up with? A course estimate of the algorithm's efficiency

Step-Counting Example 1

```
double sum = 0;
for (int i=0; i<n; i++) {
        sum = sum + list[i];
}
mean = sum / n;
```

- Before we start, we need to know something important: what is this algorithm's instance characteristic? Is there more than one?
  - Just one, the amount of data in the array, *n*

Step-Counting a For Loop

- Our example has a for loop in it. So we need to know how to step-count one!
- First, imagine we initialize an operation counter (o = 0;).
- Second, imagine we augment the code with o++; statements to count the loop's operations.

```
o = 0;  # initialize the operation counter
o++;   # initialize the for loop
for (initialization; condition; increment) {
        o++; # this is for the conditional evaluation if it evaluates to true
        loop body # we'll worry this later
        o++; # this is for the increment of the for loop
}
o++ # this is for the conditional evaluation if it evaluates to false
```

Step-Counting a For Loop

- So, for a for loop, we need to count:
  - The initialization expression before the loop
  - True evaluations of the loop condition *inside* the loop body
  - False evaluations of the loop condition *after* the loop
  - The increment expression at the end of the loop body
- What about while or do-while loops? Same theory as for loops, but take out the steps that don’t make sense!

Step-Counting Example 1

```
double sum = 0;
for (int i=0; i<n; i++) {
        sum = sum + list[i];
}
mean = sum / n;
```

- Note: these are steps 1 and 2 of the process:

1. Augment the algorithm with operation counts
2. Estimate executions of selections & iterations

```
o = 0;
double sum = 0;
o++; # assignment of variable
o++; # loop initialization
for (int i=0; i<n; i++) {
        o++; # true loop condition evaluation
        sum = sum + list[i];
        o++; o++; o++; o++; # 1 for assignment, 2 for +, 3 and 4 for array
location calculation
        o++; # loop increment
}
o++; # false loop condition evaluation
mean = sum / n;
o++; o++; # 1 for assignment, 2 for division
```

Step-Counting Example 1

- Step 3: Remove/Ignore the algorithm's statements (but retain loop information!

```
o = 0;
double sum = 0;
o++; # assignment of variable
o++; # loop initialization
for (int i=0; i<n; i++) {
        o++; # true loop condition evaluation
        sum = sum + list[i];
        o++; o++; o++; o++; # 1 for assignment, 2 for +,
3 and 4 for array location calculation
        o++; # loop increment
}
o++; # false loop condition evaluation
mean = sum / n;
o++; o++; # 1 for assignment, 2 for division
```

```
o = 0;
o++; # assignment of variable
o++; # loop initialization
Iterate n times:
        o++; # true loop condition evaluation
        o++; o++; o++; o++; # 1 for assignment, 2 for +, 3 and
4 for array location calculation
        o++; # loop increment
o++; # false loop condition evaluation
o++; o++; # 1 for assignment, 2 for division
```

Step-Counting Example 1

- Step 4: Produce a step-count expression in terms of the algorithm's instance characteristic(s)
- What it be for this step count example?
- Answer: 6n + 5

```
o = 0;
o++; # assignment of variable
o++; # loop initialization
Iterate n times:
        o++; # true loop condition evaluation
        o++; o++; o++; o++; # 1 for assignment, 2 for +, 3 and 4 for array location calculation
        o++; # loop increment
o++; # false loop condition evaluation
o++; o++; # 1 for assignment, 2 for division
```

How to Step-Count an If Statement

- Three possible approaches:

1. Optimistic: the body of the if statement is *never* executed!
2. Pessimistic: the body of the if statement is *always* executed!
3. Estimation: estimate what percent of the time the if body is executed, and make your step-count expression based on that

- What would be the safest choice?
- Pessimistic