Analyzing Code

CS 61B Spring 2016: Discussion 7

Announcements

Project 2 due Monday 3/7!

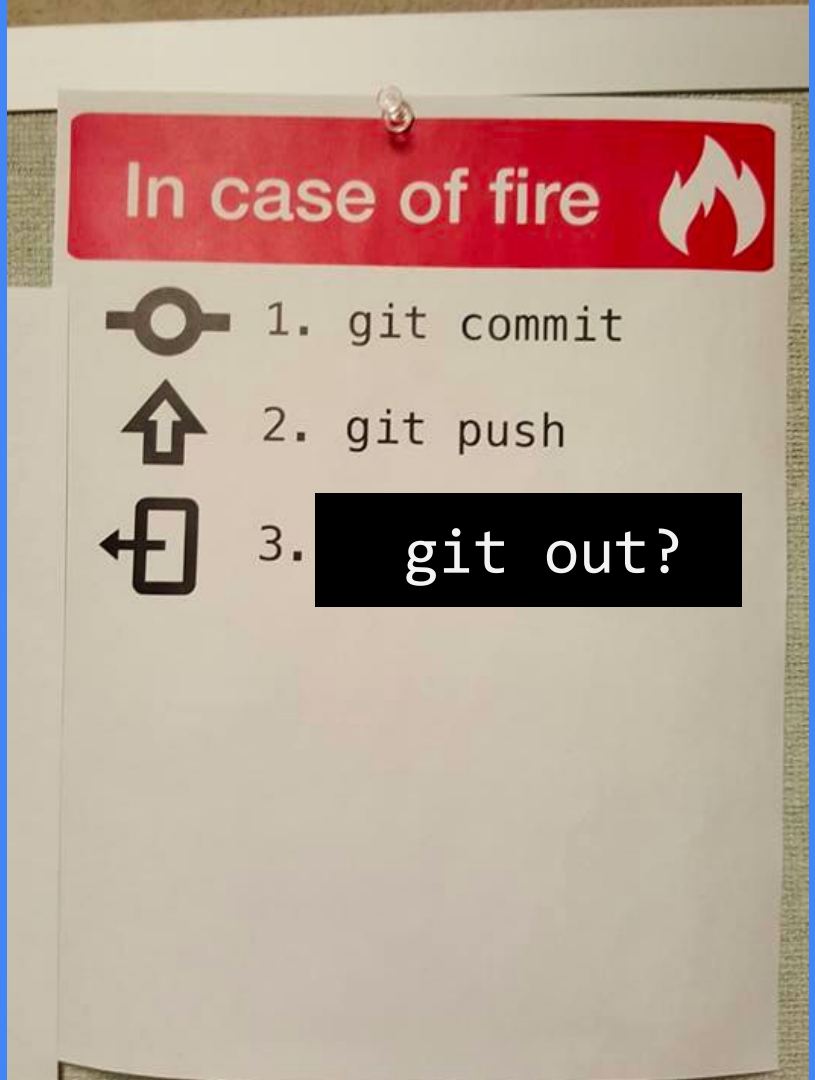
Lab this week is to get project 2 help -- come in and bother the TAs and lab assistants!

UNIX Tip of the Day: ???

~~UNIX~~ Tip of the Day:

Please work on Project 2!
If you haven’t started... START!


In case of fire
1. git commit
2. git push
3. leave building


In case of fire
1. git commit
2. git push
3. git out?

Asymptotic Notation

- Big O: Used for bounding above (less than).
- Big Omega: Used for bounding below (greater than).
- Big Theta: Used for bounding both above and below (equals).

Big Theta

$$R(N) \in \Theta(f(N))$$

means that there exists positive constants, k1 and k2, such that

$$k_1 \cdot f(N) \leq R(N) \leq k_2 \cdot f(N)$$

for values of N that are “very large”

Big O

$$R(N) \in O(f(N))$$

means that there exists a positive constant such that

$$R(N) \leq k \cdot f(N)$$

for values of N that are “very large”

	Informal meaning:	Family	Family Members
Big Theta $\Theta(f(N))$	Order of growth is f(N).	$\Theta(N^2)$	$N^2/2$ $2N^2$ $N^2 + 38N + N$
Big O $O(f(N))$	Order of growth is less than or equal to f(N).	$O(N^2)$	$N^2/2$ $2N^2$ $\lg(N)$
Big Omega $\Omega(f(N))$	Order of growth is greater than or equal to f(N).	$\Omega(N^2)$	$N^2/2$ $2N^2$ e^N
Tilde $\sim f(N)$	Ratio converges to 1 for very large N.	$\sim 2N^2$	$2N^2$ $2N^2 + 5$

Reminders

- We are looking at analyzing code for “very large” N.
- Ignore constant factors. The overall behavior has the “same shape” (order of growth).
  - i.e. $O(500N) = O(N)$, $O(200N^2 + 5N + 100) = O(N^2)$
  - A program that runs in N+1 to 2N+1 operations is really just “N” (linear order of growth)

True or false…?

Problem 1

Problem 1

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2 \log n) < O(n^3) < O(2^n) < O(n!) < O(n^n)$

Problem 1

1. True, but Theta would be a better bound.
2. False, O is the correct bound, since n^2 is strictly better than n^3.
3. True, but Theta would be a better bound.
4. False, O is the correct bound, since logn is strictly better than nlogn.
5. True, since 3^n (the dominant term in f(n)) is strictly worse than n^2 (the dominant term in g(n)).
6. True, since both dominant terms are n^2; n^2 is in big Theta of n^2.
7. False, n * logn is strictly worse than logn * logn.

Now to analyze some code...

Problem 2

Problem 2

A. Worst case: O(M+N), Best case: Omega(N)
B. Worst case: O(N^2), Best case: Omega(NlogN)

If you have some time later, try the follow-up questions that concern exactly what the function is doing and how to improve its runtime.

Let’s write some code of our own!

Problem 3

Problem 3

```
public static boolean findSum(int[] A, int x) {
  for (int i = 0; i < A.length; i++) {
    for (int j = 0; j < A.length; j++) {
      if (A[i] + A[j] == x) {
        return true;
      }
    }
  }
  return false;
}
```

Problem 3

```
public static boolean findSumFaster(int[] A, int x) {
  int left = 0;
  int right = A.length - 1;
  while (left <= right){
    if (A[left] + A[right] == x)
      return true;
    else if (A[left] + A[right] < x)
      left++;
    else
      right--;
  }
  return false;
}
```

Problem 3

Naive solution runtime: O(N^2)
Optimized solution runtime: O(N)