Prerequisites for EN 600.439/639: Computational Genomics
Fall 2013, Prof. Ben Langmead

The official prerequisites for the class are Data Structures (EN 600.226) and Intermediate Programming (EN 600.120). More specifically, before starting this class you should be comfortable with (a) data structures and algorithms, (b) complexity of algorithms, and (c) understanding Python code. It will be helpful if you can also write Python code, but this is not required. Your submissions may be in any language.

Below are several questions. **You should know the answers to these**. If there are only a few you can't answer, that's fine. If you need to review any of these concepts, try these resources:

- Leiserson, Charles E., Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. Ed. Thomas H. Cormen. The MIT press, 2001.
- *Programming Python*, *Learning Python*, *Python Cookbook*, and other Python books available for free via JHU's Safari Bookshelf
- Jeff Erickson's Algorithms Course Materials: http://www.cs.uiuc.edu/~jeffe/teaching/algorithms

Algorithms

- What is "big O" notation?
- What is the difference between O(n), o(n) and Θ(n)?
- What does it mean mathematically when we say an algorithm uses O(n) time or space?
- What is the difference between a *worst-case* and an *expected* bound?
- What is the computational complexity of Quicksort?
- How much space is required to store a list of *n* integers in the range [1, *m*]?
- What is *NP hardness*?
- How do you prove an algorithm is NP hard?

Understanding Python

(You should be able to answer these questions without actually running the code.)

- What does the following code snippet do?

```
def func1(p, t):
    occurrences = []
    for i in xrange(0, len(t) - len(p) + 1):
        mismatch = False
        for j in xrange(0, len(p)):
            if t[i+j] != p[j]:
                mismatch = True
                break
        if not mismatch:
          occurrences.append(i)
    return occurrences
```

- What does the following code snippet do?

```
def func2(a, x):
   lo, hi = 0, len(a)
   while lo < hi:
```

```
        mid = (lo+hi) // 2
        midval = a[mid]
        if midval < x:
            lo = mid+1
        elif midval > x:
            hi = mid
        else:
            return mid
    return -1
```

- What does the following code snippet do?

```
tab = {1:1, 2:1}
def func3(n):
  if n <= 2:
     return 1
  if n in tab:
     return tab[n]
  else:
     tab[n] = func3(n-1) + func3(n-2)
     return tab[n]
```

- Does Python use *mutable* or *immutable* strings?
    - Why does that matter?
- What is the Python *numpy* module used for?

I will try to avoid idiomatic or confusing Python code in my examples in class.

Other basics

- What is a *graph* or *network*?
    - What is the difference between a *directed* and an *undirected* graph?
- What is a *tree*?
    - What is the difference between a *rooted* and an *unrooted* tree?
    - What is a *binary* tree?
    - What is a *balanced* tree?
    - What is *depth-first traversal* of a tree?
    - What are some other ways to traverse trees?
- What is the *pigeonhole principle*?
- What is a *string*?
    - What is a *substring*?
- What is a *total order*?