MATH 157: Mathematics in the world

Homework 7 (Due April 10th, 2017 at 4:00PM)

Problem 0 - Meeting with me for final project

If you haven't yet had a final project meeting or scheduled one, email me immediately with your availability.

PLEASE SOLVE COMPLETELY 5 OF THE FOLLOWING 6 PROBLEMS

Problem 1

We have already encountered the notion of asymptotic functions in previous homeworks. We write $f \sim g$ if $\lim_{x\to\infty} f(x)/g(x) = 1$. Colloquially, this means that $f(x)$ and $g(x)$ behave identically for large x.

The statement $f = O(g)$, pronounced "f is O of g", has a related but different meaning. In simple terms, we should interpret it as f grows no faster than g. Note the apparent lack of symmetry (also suggested by the notation): $f = O(g)$ does not imply $g = O(f)$. Instead of delving into a formal discussion of O and its properties, we will refer to the following list of resources.

- `http://en.wikipedia.org/wiki/Big_O_notation`
- `http://adrianmejia.com/blog/2014/02/13/algorithms-for-dummies-part-1-sorting/`
- `http://stackoverflow.com/questions/487258/plain-english-explanation-of-big-o/487278#487278`
- `http://ssp.impulsetrain.com/big-o.html`

To complete this exercise, identify each of the following statements as true or false. The goal is not to prove these statements, but gain a practical intuition about big O notation.

1. $2x = O(x)$
2. $x^2 = O(x)$
3. $x^3 = O(x^4)$

4. $x^2 + 1000x = O(x^2)$
5. $x \ln x = O(x)$
6. $x = O(x \ln x)$
7. $\ln x = O(x^{0.001})$
8. $x^{10} = O(e^x)$
9. $2^x = O(e^x)$
10. $e^x = O(2^x)$

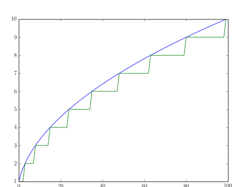
Problem 2

Accomplish the following tasks without using conditional statements. For simplicity, you can assume all integers are non-negative.

1. Write a function `toggle_rightmost_1` which toggles the rightmost 1 bit in a given integer. [1]
2. Write a function `two_powers` which check if an integer is of the form $2^a + 2^b$ for some two distinct powers a and b. [2]
3. (Extra credit) Write a function `min_special(a, b)` which computes the minimum of two integers a and b. [3]

Problem 3

1. Write a function `int_sqrt` which computes the floor of the square root of an integer using binary search.
2. Plot `int_sqrt(n)` and the regular standard square root `sqrt(n)` for $1 \leq n \leq 100$.



[1] Hint: Try to combine x and $x - 1$ with different binary operations.
[2] Hint: Use `toggle_rightmost_1`. Be careful that `toggle_rightmost_1(0) = 0`.
[3] Try to avoid the `abs` function since it uses a conditional statement inside.

Problem 4

Suppose A is a sorted array of distinct integers. We are interested in finding an index i such that $A[i] = i$. [4]

1. Write a function `Ai_eq_i` which given A returns either the integer i or -1 if no such i exists. [5]

2. Test your implementation on the following two inputs:

$$\begin{aligned} A &= [\ \ \ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], \\ B &= [-1, 1, 3, 4, 5, 6, 7, 8, 9, 10]. \end{aligned}$$

Problem 5

During a sleepless night in Las Vegas, you find yourself wandering in a casino with \$50 in your pocket. After inspecting the floor, you find a blackjack table you would like to play at. Unfortunately, the minimum bet is \$100. The dealer advises you to try your luck at roulette in order to meet the minimum.

You think about it and come up with two strategies. You can put all of your \$50 on evens. [6] If you win the house will double your bet to \$100, and if not you walk right out. Alternatively, you can bet on evens one dollar at a time. Again, you plan to do so until you either acquire \$100 or lose everything. Which strategy would you pick and why? [7] Would your answer change if you are in Monaco instead?

Problem 6

Consider a sequence of independent identically distributed random variables $X_1, X_2, \dots$ such that for each i either $X_i = 1$ or $X_i = -1$, both with probability $1/2$. The collection of partial sums $S_n = \sum_{i=1}^{n} X_i$ for $n \geq 0$ constitute a *random walk*. Such processes have found many applications from physics and biology to financial mathematics. We will use this problem to explore computationally several questions concerning the behavior of random walks.

1. Write a function `random_walk(k, n)` which returns a $k \times n$ matrix each of whose rows is a random walk. Don't use any loops. [8]

---

[4] Assume all arrays in this homework are indexed from 0. If multiple solutions i exist, any one of them would suffice.

[5] Hint: Can you reduce this problem to a binary search?

[6] In American roulette, 18 out of the 38 possible outcomes count as even. In the European version, the evens are 18 out of 37 possibilities. You can assume all outcomes are equally likely.

[7] Hint: In both cases, compute the probability of winning \$100. Do you see any resemblance between the second strategy and the *Gambler's ruin* problem from class?

[8] Hint: You can first generate a $k \times n$ matrix with entries ± 1. This can be done either by conditioning a uniform variable (see previous homework), or by manipulating the output from `np.random.binomial`. Then compute row-wise cumulative sums using `np.cumsum`.

2. Plot the progression progression of $k = 10$ random walks of length $n = 10^5$. Overimpose the plots of $\pm 2\sqrt{x}$. You can use a loop of length k to go over different random walks.

3. Use $k = 100$ and $n = 10^5$ to compute the variance at each step of a random walk. Don't use any loops.

   Produce a log-log plot of your results. Conjecture a value α such that $Var(S_n) = O(n^\alpha)$. [9]

4. Given a random walk $S_0, S_1, S_2, \ldots$, we can count how many times it returns to the origin. The variable

$$R_n = |\{i \mid X_i = 0 \text{ and } 0 \leq i \leq n\}|$$

   represents how many times the random walk is at the origin up to its n-th step. Since the random walk starts at the origin $S_0 = 0$, we take the convention that $R_0 = 1$.

   Write a function `num_returns(X)` which accepts a $k \times n$ matrix consisting of random walks. The output should be a $k \times n$ matrix whose value at position (i, j) is the number of returns the i-th random walk has makes to the origin up to and including step j. Since each random walk starts at 0, all entries of your result should be positive. Don't use any loops. [10]

   Take column-wise means of the output of `num_returns` to compute approximations for $\mathbb{E}(R_n)$. Produce a log-log plot of your results and use it to conjecture a value β such that $\mathbb{E}(R_n) = O(n^\beta)$.

5. (Extra credit) Prove the conjectures you made above.

6. (Extra credit) Investigate the sequence of variables

$$M_n = \max\{|S_i| \mid 0 \leq i \leq n\}.$$

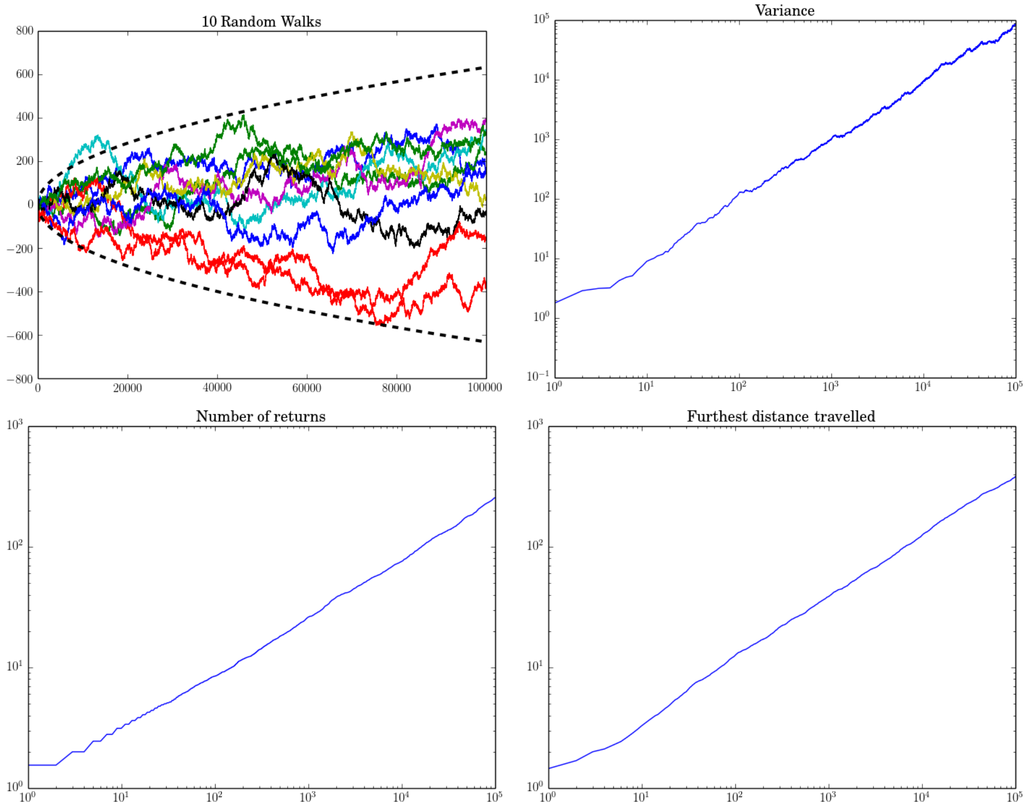
   The value of M_n is the furthest distance the underlying random walk has traveled from the origin up to step n. Make a conjecture of the form $\mathbb{E}(M_n) = O(n^\gamma)$ and prove your claim.

---

[9] Hint: You can use `np.var` to compute the column-wise variance for the output of `random_walk(k, n)`.

If $y = C \cdot x^\alpha$, then $\log y = \alpha \cdot \log x + \log C$. It follows that the log-log plot of x and y is a line of slope α. For the latter part, you can compare your plot to a line and compute its slope to guess α. The value of α is easy enough to guess directly, so you don't need to fit a regression line.

[10] Hint: You can compare the input X to 0 using `np.equal`. Convert the Boolean output to integers (0 and 1), compute row-wise cumulative sums, and add 1.


10 Random Walks
Variance
Number of returns
Furthest distance travelled