HW 0 (due Monday, at noon, January 26, 2015)

OLD CS 473: Fundamental Algorithms, Spring 2015 Version: **1.01**

This homework contains two problems. **Read the instructions for submitting homework on the course webpage**.

You also have to do quiz 0 online (on moodle).

Collaboration Policy: For this homework, each student should work independently and write up their own solutions and submit them.

Read the course policies before starting the homework.

- Homework 0 and Quiz 0 test your familiarity with prerequisite material: big-Oh notation, elementary algorithms and data structures, recurrences, graphs, and most importantly, induction, to help you identify gaps in your background knowledge. You are responsible for filling those gaps. The course web page has pointers to several excellent online resources for prerequisite material. If you need help, please ask in headbanging, on Piazza, in office hours, or by email.
- Each student must submit individual solutions for these homework problems. For all future homeworks, groups of up to three students may submit (or present) a single group solution for each problem.
- Please carefully read the course policies on the course web site. If you have any questions, please ask in lecture, in headbanging, on Piazza, in office hours, or by email. In particular:
  - Submit separately stapled solutions, one for each numbered problem, with your name and NetID clearly printed on each page, in the corresponding drop boxes outside 1404 Siebel.
  - **You may use any source at your disposal**: paper, internet, electronic, human, or other, but you must write your solutions in your own words, and you must **cite explicitly**[1] every source that you use (except for official course materials). Please see the academic integrity policy for more details.
  - No late homework will be accepted for any reason. However, we may forgive quizzes or homeworks in extenuating circumstances; ask the instructor for details.
  - Answering "I don't know" to any (non-extra-credit) problem or subproblem, on any homework or exam, is worth 25% partial credit.
  - Algorithms or proofs containing phrases like and so on or repeat this process for all n, instead of an explicit loop, recursion, or induction, will receive a score of 0.
  - Unless explicitly stated otherwise, every homework problem requires a proof.

1 Required problems

1. (60 PTS.) Moving numbers. (50 PTS.)

(A) (50 PTS.) The input is a multiset X of n positive integer numbers in the range 1 to k. Consider the famous algorithm:

[1] For example: "I found the solution to this exercise on `http://www.endoftheinternet.com/`. Since I understand the submission guidelines, I read this solution carefully, understood it, believe that it is correct, and I wrote it out in my own words. I was, of course, not so mind boggling stupid to just cut and paste some random text I found on the internet." (Of course, you need only the first sentence.)

```
play(X) :
  while X contains more than one element do
      if X contains the number 0 then
          Remove the number 0 from X
          continue
      Two distinct elements x_1 and x_2 are chosen arbitrarily from X
      y_1 = min(x_1, x_2) − 1
      y_2 = max(x_1, x_2) + 1
      X ← (X \ {x_1, x_2}) ∪ {y_1, y_2}
```

Here is an example of the execution of **play**($\{1, 2, 3, 4\}$).

$$\{1,2,3,4\} \implies \{1,2,2,5\} \implies \{0,2,3,5\} \implies \{2,3,5\} \implies \{2,2,6\} \implies \{3,1,6\} \implies \{1,2,7\} \implies \{1,8,1\} \implies \{0,2,8\} \implies \{2,8\} \implies \{1,9\} \implies \{0,10\} \implies \{10\}.$$

Prove (maybe using induction, but you do not have to) that **play** always terminates.
(Hint: Come up with an argument why in each step some non-trivial progress is being made. As a warm-up exercise, prove that the algorithm always terminates if the initial input has three numbers.)

(B) (10 PTS.) (Harder.[2]) Consider the algorithm **work**, a variant of **play**, that increases the maximum number by two (instead of by one) in each step. Prove that the modified algorithm **work** terminates, and give an upper bound, as small as possible, on the number of steps needed in the worst case for it to terminate, as a function of n and k.

2. (40 PTS.) Random walk.

A ***random walk*** is a walk on a graph G, generated by starting from a vertex $v_0 = v \in V(G)$, and in the $i$þstage, for $i > 0$, randomly selecting one of the neighbors of v_{i-1} and setting v_i to be this vertex. A walk $v_0, v_1, \ldots, v_m$ is of length m.

(A) (20 PTS.) For a vertex $u \in V(G)$, let $P_u(m, v)$ be the probability that a random walk of length m, starting from u, visits v (i.e., $v_i = v$ for some i).
Prove that a graph G with n vertices is connected, if and only if, for any two vertices $u, v \in V(G)$, we have $P_u(n-1, v) > 0$.

(B) (20 PTS.) Prove that a graph G with n vertices is connected if and only if for any pair of vertices $u, v \in V(G)$, we have $\lim_{m \to \infty} P_u(m, v) = 1$. (Hint: Use (A).)

[2]The *harder* designation means that we do not expect all the students in the class to solve this part (in particular, it would be suspicious if everybody solves it). Solving a hard problem might require absurd amount of time. You should expect not to be able to solve all the homework questions in this class, especially the parts marked as hard. You had been officially warned.