CS 787: Advanced Algorithms	Homework 1
Out: 09/22/15	**Due: 10/06/15**

Guidelines

- This homework consists of 8 problems. Only the problems marked with an asterisk (three in all) will be graded. You are not required turn in solutions to the unmarked problems but you are highly encouraged to solve and write up your solutions to all of them. We will provide solutions for all the problems.
- Some of the problems are difficult, so please get started early. Late submissions do not get any credit.
- Please typeset your solutions.
- Homework should be done in pairs. Please write your names clearly on your homework.

Problems

1. You are playing a game of chance but have lost the dice that came with it. Each move in the game depends on the sum of two standard dice rolls (that is, the sum of two independent numbers, each distributed uniformly at random between 1 and 6). You have a fair coin at your disposal. Using this coin you would like to simulate the distribution of the sum. Give a protocol for doing so that uses as few coin flips as possible to generate each draw of the sum.

2. Clock solitaire is played using a standard deck of 52 cards as follows. The deck is divided randomly into 13 piles of 4 cards each such that each card is equally likely to end up in any of the 13×4 positions. The piles are labeled $A, 2, 3, \cdots, J, Q, K$ in an arbitrary manner. The game begins by picking the topmost card in the pile labeled A. At every subsequent step, the player picks the topmost card from the pile with the same label as the card previously picked. The game ends when either all the cards have been picked, or the player attempts to pick a card from an empty pile. In the former case, the player wins, and in the latter case she loses. Determine the probability that the player wins the game.

3. For $i \in \{1, \cdots, n\}$, let X_i denote a random variable whose value is drawn independently and uniformly at random from the interval $[0, 1]$. Let $X = \min_{i \in [n]} X_i$. Compute the expectation of X.

4. (*) Recall that in the *Coupon Collector* problem, at every step you receive a random coupon out of a set of n different coupons; The goal is to determine how many coupons you need to collect in expectation before you have seen at least one of each kind. Consider the variant where at every step the probability of receiving the ith coupon is p_i. Note that $\sum_{i=1}^{n} p_i = 1$. Prove that the expected number of steps in the process is at least P and at most $O(P \log n)$, where $P = \max_{i \in [n]}(1/p_i)$.

5. (*) In the *Max-Cut* problem, we are given an unweighted graph G with vertex set V and edge set F; Our goal is to partition V into V_1 and V_2, so that the total number of edges "cut" by the partition, $|F \cap (V_1 \times V_2)|$, is maximized. Consider the following randomized algorithm: for every vertex $v \in V$, independently flip an unbiased coin; Let V_1 be the set of all vertices for which the coin came up heads, and let V_2 be the remaining vertices.

   (a) What is the expected number of edges cut?

   (b) Determine an upper bound on the probability that fewer than $|F|/4$ edges were cut. Try to obtain as small a bound as you can.

6. Recall that a function f mapping reals to reals is said to be convex if and only if for every $a, b \in \Re$, $(f(a) + f(b))/2 \geq f((a+b)/2)$. Prove that for any convex function f and any real-valued random variable X, $\mathrm{E}[f(X)] \geq f(\mathrm{E}[X])$. This is called *Jensen's inequality*.

7. Extend the Chernoff bound discussed in class to the case of arbitrary random variables $X_i \in [0, 1]$ following the approach below.

   (a) Show that $f(x) = e^{tx}$ is a convex function.

   (b) Show that if C is a random variable taking values in $[0, 1]$, and B is a $\{0, 1\}$ random variable with $\mathrm{E}[C] = \mathrm{E}[B]$, then for any convex f, $\mathrm{E}[f(C)] \leq \mathrm{E}[f(B)]$.

   (c) Use these to reprove the Chernoff bound for sums of independent random variables over $[0, 1]$.

8. (*) Give a polynomial time algorithm for the attached ACM ICPC problem "The Lost House". Prove that your algorithm is correct and analyze its running time. (You are not required to write or implement code.)

3141 - The Lost House

Asia - Beijing - 2004/2005

One day a snail climbed up to a big tree and finally came to the end of a branch. What a different feeling to look down from such a high place he had never been to before! However, he was very tired due to the long time of climbing, and fell asleep. An unbelievable thing happened when he woke up he found himself lying in a meadow and his house originally on his back disappeared! Immediately he realized that he fell off the branch when he was sleeping! He was sure that his house must still be on the branch he had been sleeping on. The snail began to climb the tree again, since he could not live without his house.

When reaching the first fork of the tree, he sadly found that he could not remember the route that he climbed before. In order to find his lovely house, the snail decided to go to the end of every branch. It was dangerous to walk without the protection of the house, so he wished to search the tree in the best way.

Fortunately, there lived many warm-hearted worms in the tree that could accurately tell the snail whether he had ever passed their places or not before he fell off.

Now our job is to help the snail. We pay most of our attention to two parts of the tree the forks of the branches and the ends of the branches, which we call them key points because key events always happen there, such as choosing a path, getting the help from a worm and arriving at the house he is searching for.

Assume all worms live at key points, and all the branches between two neighboring key points have the same distance of 1. The snail is now at the first fork of the tree.

Our purpose is to find a proper route along which he can find his house as soon as possible, through the analysis of the structure of the tree and the locations of the worms. The only restriction on the route is that he must not go down from a fork until he has reached all the ends grown from this fork.

The house may be left at the end of any branches in an equal probability. We focus on the mathematical expectation of the distance the snail has to cover before arriving his house. We wish the value to be as small as possible.

As illustrated in Figure-1, the snail is at the key point 1 and his house is at a certain point among 2, 4 and 5. A worm lives at point 3, who can tell the snail whether his house is at one of point 4 and 5 or not. Therefore, the snail can choose two strategies. He can go to point 2 first. If he cannot find the house there, he should go back to point 1, and then reaches point 4 (or 5) by point 3. If still not, he has to return point 3, then go to point 5 (or 4), where he will undoubtedly find his house. In this choice, the snail covers distances of 1, 4, 6 corresponding to the circumstances under which the house is located at point 2, 4 (or 5), 5 (or 4) respectively. So the expectation value is (1 + 4 + 6) / 3 = 11 / 3. Obviously, this strategy does not make full use of the information from the worm. If the snail goes to point 3 and gets useful information from the worm first, and then chooses to go back to point 1 then towards point 2, or go to point 4 or 5 to take his chance, the distances he covers will be 2, 3, 4 corresponding to the different locations of the house. In such a strategy, the mathematical expectation will be (2 + 3 + 4) / 3 = 3, and it is the very route along which the snail should search the tree.

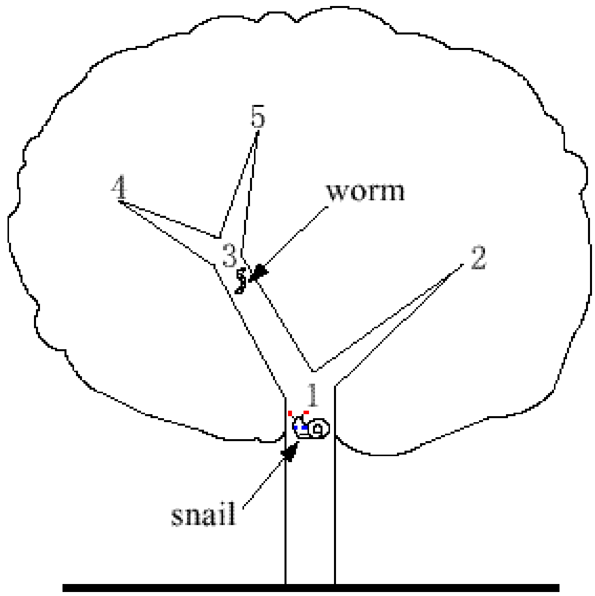

Figure-1

Input

The input contains several sets of test data. Each set begins with a line containing one integer N, no more than 1000, which indicates the number of key points in the tree. Then follow N lines describing the N key points. For convenience, we number all the key points from 1 to N. The key point numbered with 1 is always the first fork of the tree. Other numbers may be any key points in the tree except the first fork. The i-th line in these N lines describes the key point with number i. Each line consists of one integer and one uppercase character 'Y' or 'N' separated by a single space, which represents the number of the previous key point and whether there lives a worm ('Y' means lives and 'N' means not). The previous key point means the neighboring key point in the shortest path between this key point and the key point numbered 1. In the above illustration, the previous key point of point 2 or 3 is point 1, while the previous key point of point 4 or 5 is point 3. This integer is -1 for the key point 1, means it has no previous key point. You can assume a fork has at most eight branches. The first set in the sample input describes the above illustration.

A test case of N = 0 indicates the end of input, and should not be processed.

Output

Output one line for each set of input data. The line contains one float number with exactly four digits after the decimal point, which is the mathematical expectation value.

Sample Input

```
5
-1 N
1 N
1 Y
3 N
3 N
10
-1 N
1 Y
1 N
2 N
```

```
2 N
2 N
3 N
3 Y
8 N
8 N
6
-1 N
1 N
1 Y
1 N
3 N
3 N
0
```

Sample Output

```
3.0000
5.0000
3.5000
```

Beijing 2004-2005