Assignment 5

Abhilash I

February 27, 2009

1 Counting

1.1 Problem Description

You have to report the number of $n - ary$ trees that can constructed from N nodes.

1.2 Input Format:

Each line denotes a test case with two integer nN Input ends with end of file.

1.3 Output Format:

For each test case output the number of $n - ary$ trees that can be formed from N nodes.

1.4 Sample Input

```
2 1
```

1.5 Sample Output:

```
1
```

2 Diameter of a Binary Tree

2.1 Problem Description

The diameter of a tree is the number of nodes on the longest path between two leaves in the tree.The diameter of a tree T is the largest of the following quantities:

- The diameter of T's left subtree
- The diameter of T's right subtree
- The longest path between leaves that goes through the root of T(this can be computed from the heights of the subtrees of T)

The Fig 1 and Fig 2 below shows two trees each with diameter nine, the leaves that form the ends of a longest path are shaded (note that there is more than one path in each tree of length nine, but no path longer than nine nodes).

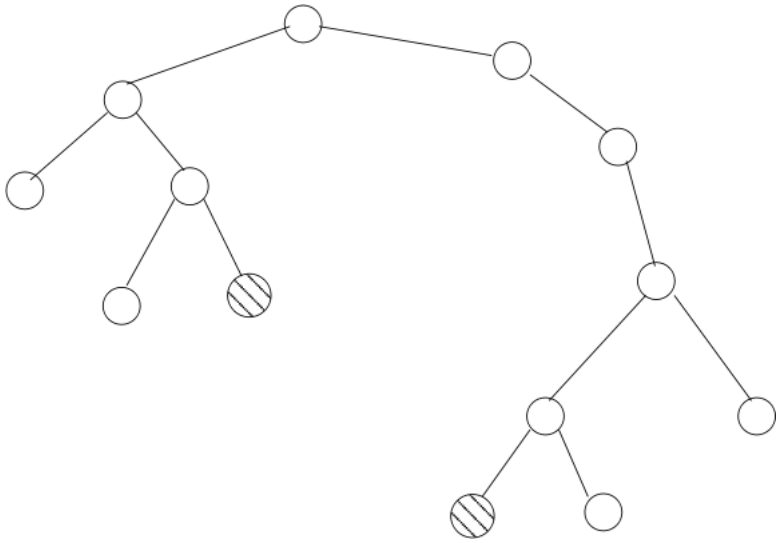

Figure 1: Diameter 9 through root

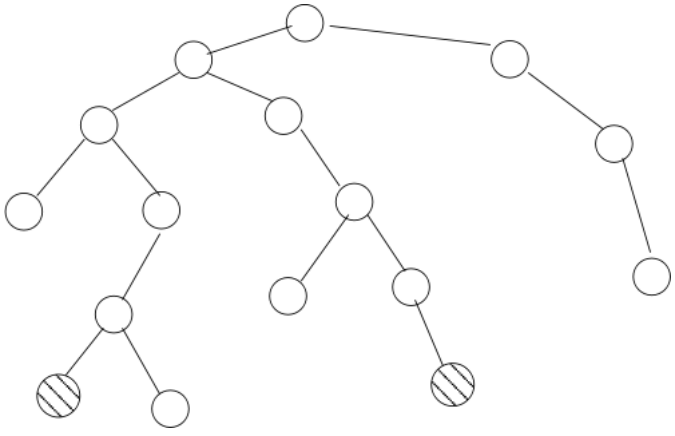

Figure 2: Diameter 9 not through root

2.2 Input Format:

Sequence of Numbers(without repetition), terminated by newline character.

2.3 Output Format:

Height of tree T. Diameter of tree T.

2.4 Sample Input

```
10 15 5 7 4 3 12 6 17
```

2.5 Sample Output:

```
3
6
```

3 Huffman code construction

3.1 Problem Description

Suppose we have messages consisting of sequences of characters. In each message, the characters are independent and appear with a known probability in any given position; the probabilities are the same for all positions.
E.g. - Suppose we have a message made from the five characters a, b, c, d, e, with probabilities 0.12, 0.40, 0.15, 0.08, 0.25, respectively. We wish to encode each character into a sequence of 0s and 1s so that no code for a character is the prefix of the code for any other character. This prefix property allows us to decode a string of 0s and 1s by repeatedly deleting prefixes of the string that are codes for characters.

Table 1: Huffman Encoding Table

Symbol	Probability	Code#2	Code#2
a	0.12	000	1111
b	0.40	001	0
c	0.15	010	110
d	0.08	011	1110
e	0.25	100	10

In the above, Code 1 has the prefix property. Code 2 also has the prefix property. The problem: given a set of characters and their probabilities, find a code with the prefix property such that the average length of a code for a character is a minimum.

Code 1: $(0.12)(3) + (0.4)(3) + (0.15)(3) + (0.08)(3) + (0.25)(3) = 3$
Code 2: $(4)(.12) + (1)(.4) + (3)(.15) + (4)(.08) + (2)(.25) = 2.15$

Huffmans algorithm is one technique for finding optimal prefix codes. The algorithm works by selecting two characters a and b having the lowest probabilities and replacing them with a single (imaginary) character, say x, whose probability of occurrence is the sum of probabilities for a and b. We then find an optimal prefix code for this smaller set of characters, using this procedure recursively. The lighter node gets 0 and the Heavier one gets 1 (You can safely assume that at any instant the probabilities won't be equal).The code for the original character set is obtained by using the code for x with a 0 appended for "a" and a 1 appended for "b". We can think of prefix codes as paths in binary trees.Figure 3

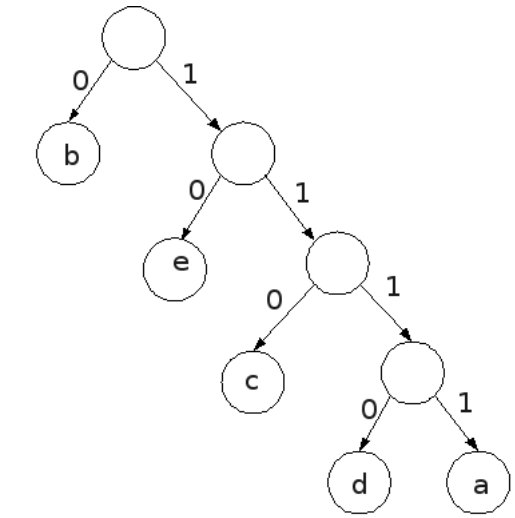

Figure 3: Huffman Encoding Tree

3.2 Input Format:

Number of Characters in the sequence followed by each character and its corresponding probability.

3.3 Output Format:

Average length of the code followed by each character and its corresponding code word. Sequence of characters remain same as that of Input.

3.4 Sample Input

```
5
a 0.12
b 0.40
c 0.15
```

```
d 0.08
e 0.25
```

3.5 Sample Output:

```
2.15
a 1111
b 0
c 110
d 1110
e 10
```

4 Root to Leaf Path

4.1 Problem Description

We'll define a "root-to-leaf path" to be a sequence of nodes in a tree starting with the root node and proceeding downward to a leaf (a node with no children). We'll say that an empty tree contains no root-to-leaf paths. So for example, the following tree has exactly four root-to-leaf paths.

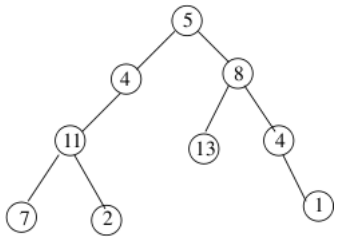

Figure 4: Binary tree for root to leaf path

Root-to-leaf paths:
Path 1: 5 4 11 7
Path 2: 5 4 11 2
Path 3: 5 8 13
Path 4: 5 8 4 1

Given a sequence of numbers, construct a Binary Search tree T, print out all of its root-to-leaf paths, one per line.

4.2 Input Format:

Sequence of numbers (without repetition), terminated by newline character

4.3 Output Format:

Root-to-leaf paths, one per line, from left to right.

4.4 Sample Input

```
10 15 5 7 4 3 12 6 17
```

4.5 Sample Output:

```
10 5 4 3
10 5 7 6
10 15 12
10 15 17
```