



The amazing sailors

Problem

Submissions

Leaderboard

Discussions

Tragedy has struck the suez canal, a giant ship is stuck blocking the traffic from crossing. India's oil shipments from USA cannot cross the suez canal now, leading to oil prices crossing Rs. 100 (Saudi smirk)

The amazing part-time sailors of India, Dr. Mayank Chaturvedi and Dr. Vinayak Shukla with their captain Dr. Amandeep are called by the Indian government for the rescue. After a tough voyage, they finally have a map that claims to offer them a safe alternate passage to the suez canal. The sailors now want to compress this map and send it back to the Indian government.

Luckily the three amazing sailors are Mathematicians and they have figured out that this compression of the map can be obtained by the following way:

Each letter in the map as usual is assigned a binary code but slightly different than ASCII. The letters with higher frequency have shorter binary representation and the ones with lower frequency have longer binary representation. Also, it is made sure that no code for any character appeared as the prefix of a code for another character, to prevent ambiguity while decoding. (This representation is such that the final message encoding would have minimum length for ANY binary encoding of characters, given no character encoding is prefix of another)

To decode, the sailors also planned to add the appearing characters (in map) with their frequencies along with the compressed map.

However, alas, it was a Friday and Dr. Mayank Chaturvedi, heavily drunk on cheap rum, messed up the input and encoding. Instead of the intended method, he assigned the binary codes in the opposite order of frequency, that is, more frequent characters ended up with longer codes and vice versa. Also, the least frequent character got a frequency value 0 and the rest got the difference between their actual frequency values and the actual minimum frequency (see example below).

(Read the notes below before attempting)

Your Job is to decrypt the map and give out the readable sentence

Notes:

- Higher frequency characters get assigned longer (or equal length) and higher valued binary codes than lower frequency characters.
- In case of a tie in frequency, the character which appears before the other in the alphabet gets assigned the lower valued code.
- 1 is considered a higher valued code than 0.
- No code for any character is allowed to be the prefix of the code for another character. (i.e. 0 for 'a' and 01 for 'b' is not allowed as 0 is a prefix for 01)
- The output string can have 1 to 95 characters (both inclusive).

Input Format

The first line contains T, the number of test cases. For each test case, the first line contains N, the number of distinct characters appearing in the encrypted string. The next N lines contain the character and the frequency of that character, separated by a space. (Characters will be given in an alphabetical order) Finally, the next line contains the encrypted string.

Constraints

- $1 \leq T \leq 255$
- $1 \leq N \leq 26$
- Input encrypted string length ≤ 1200 characters
- The output string will consist of only lowercase English characters (a-z).

Output Format

For each test case, output the decrypted string in separate lines.

Sample Input 0

```
1
4
a 3
b 0
c 1
d 0
11111111111011011010
```

Sample Output 0

```
aaaabcccd
```

Explanation 0

Here, the binary codes for the letters, calculated from the frequencies are: b: 0 d: 10 c: 110 a: 111

Notice how none of them are the prefix of any other code.

Decrypting the input string according to the calculated codes gives us the solution string "aaaabcccd".

Sample Input 1

```
1
7
a 1
b 0
```



i 0
m 0
n 0
o 1
t 0
101101111111111011111001111111111011110



Sample Output 1

imonaboat

f t in


Submissions: 33
Max Score: 100
Difficulty: Medium
Rate This Challenge:
☆☆☆☆☆
More

Current Buffer (saved locally, editable)  

C  

```
1▼ #include <stdio.h>
2 #include <string.h>
3 #include <math.h>
4 #include <stdlib.h>
5
6▼ int main() {
7
8▼     /* Enter your code here. Read input from STDIN. Print output to STDOUT */
9     return 0;
10 }
11
```

Line: 1 Col: 1

 Upload Code as File ☐ Test against custom input

Run Code Submit Code