acm International Collegiate Programming Contest

IBM event sponsor

Southwestern Europe Regional Contest

Solution outlines

November 17, 2013

Universitat Politècnica de València

etsinf Escola Tècnica Superior d'Enginyeria Informàtica

DSIC Departament de Sistemes Informàtics i Computació

1 Congratulations!

Congratulations and thank you for participating in the 2013 Southwestern Europe Programming Contest! We hope you enjoyed the contest and your stay in Valencia.

In this text you will find short explanations of the problems you just tried to solve in the contest. All the problems will be published at the ICPC Live Archive in a few weeks.

Have a safe trip back home.

2 The problems team

Contest Director: Jon Ander Gómez Adrián
Chief Judge: Ximo Planells

Problem Set team:

- Paco Álvaro
- Darya Bogdanova
- Enrique Flores
- Sergio García
- Jon Ander Gómez
- Mahbubul Hasan
- Shiplu Jawlader
- Derek Kisman
- Rujia Liu
- Abdullah al Mahmud
- Shahriar Manzoor
- Carlos Martínez
- Joan Pastor
- Ximo Planells
- Mario Rodríguez
- Emilio Vivancos

Judges:

- Paco Álvaro
- Enrique Flores
- Carlos Martínez
- Joan Pastor
- Moisés Pastor
- Emilio Vivancos

3 Problem A: Mixing Colours

Author: Paco Álvaro
Type: Dynamic Programming

This problem can be solved using Dynamic Programming. Concretely, the well-known Cocke-Younger-Kasami (CYK) algorithm. We only have to transform the rules to a Context-Free Grammar duplicating each colour combination. From *Blue* + *Yellow* → *Green* we create:

- *Green* → *Blue Yellow*
- *Green* → *Yellow Blue*

If we have a long sequence of colours, the resulting probability can be really close to zero and produce numerical errors. We can solve this problem by maximizing the logarithm of the probability

$$\hat{c} = \arg\max_{c}\ p(c) = \arg\max_{c}\ \log(p(c))$$

Thus, the computation of the probability

$$p(c) = p(a) \cdot p(b)$$

becomes

$$\log(p(c)) = \log(p(a)) + \log(p(b))$$

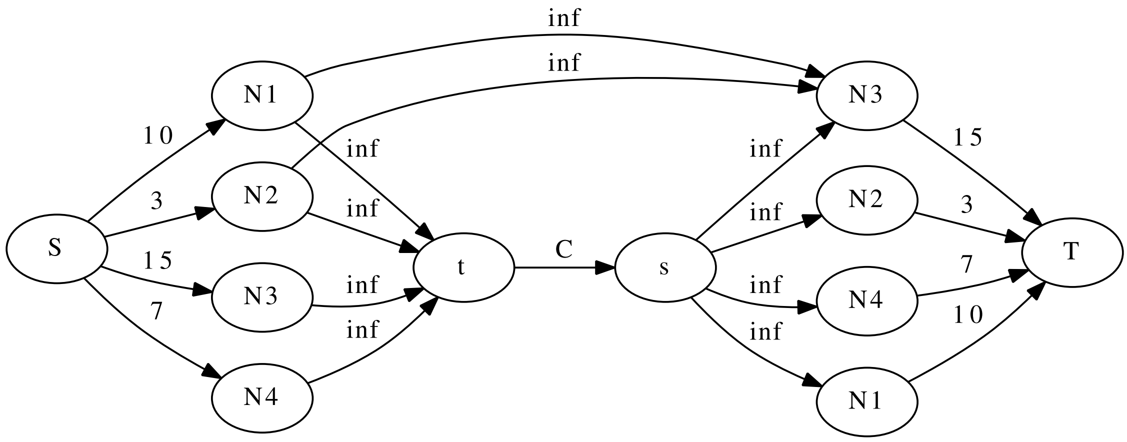
More information:

- Context-free grammar at Wikipedia: http://en.wikipedia.org/wiki/Context-free_grammar
- CYK algorithm at Wikipedia: http://en.wikipedia.org/wiki/CYK_algorithm
- Explanation by Chris Manning: http://www.youtube.com/watch?v=hq80J8kBg-Y

4 Problem B: It can be Arranged

Author: Shiplu Hawlader
Type: Maximum Flow, Binary Search

This problem can be solved using maximum flow. Take each course as a node and add an edge from source with capacity equals to the number of rooms required for that course. Also, if it is possible to go from course i to course j, add an edge from N_i to N_j with infinite capacity. The following network shows the configuration for the second test case



The network has the course nodes duplicated such that it has been split into two parts. Also, the edge from node t to s has capacity C, which represents the maximum number of classes hired.

Initially the number of classes is the sum of all the classes required by the courses. In this example, C will be initially 35. If we run maximum flow in this network from S to T with $C = 35$, when the algorithm finishes we only need to check if the capacities from S to each course N_i have been satisfied. Finally, run a binary search with $C \in [1, 35]$ to find the minimum value of C that satisfies all courses.

More information:

- Maximum flow at Wikipedia: http://en.wikipedia.org/wiki/Maximum_flow_problem
- Chapter 26 from the book Introduction to Algorithms by Cormen et al.

5 Problem C: Shopping Malls

Author: Jon Ander Gómez
Type: Graphs

This is a typical shortest-path problem although some edges have different cost depending on the direction you walk them. You just need to create a directed graph with the restrictions of the problem.

The problem can be solved using Dijkstra algorithm's or Floyd-Warshall.

More information:

- Floyd-Warshall at Wikipedia: http://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
- Dijkstra's algorithm at Wikipedia: http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

6 Problem D: Decoding the Hallway

Author: Mahbubul Hasan
Type: Ad Hoc

Slow Solution:

If we check the first iterations we will find patterns.

$$L$$
$$(L)L(R)$$
$$(LLR)L(LRR)$$

so if the previous string is S, the new string would be SL(reverse of complement of S).

To solve this problem, we loop through S to find some L and consider it as the middle point at nth stage. Then we check if the left portion is suffix of $n-1$ and right portion is suffix of reverse of complement of $n-1$.

Now considering the size of the $n-1$th string, we can solve this part using the solution of "encoding" problem.

Fast Solution:

Let's define couple of notation before approaching the solution: $\{S\}$ = reverse of string S. For example, if $S = LLR$ then $\{S\} = RLL$. $[S]$ = complement of string S. That is $L \to R$, $R \to L$. So if $S = LLR$, $[S] = RRL$. We will use (S) just to group a string. No special meaning Now if we evaluate the strings for different n, we will see a pattern. For $n = 1, L$ For $n = 2, LLR$ For $n = 3, LLRLLRR$

The pattern can be noticed if the strings are grouped: L, $(L)L(R)$, $(LLR)L(LRR)$.

So, if S is the string for n, then for $n + 1$ the string will be: $(S)L[\{S\}]$. Let us denote S_n by nth string. Now, the crucial observation is that, All the substrings of length $|S_n|$ of $|S_{n+3}|$ appears in $|S_n|$. I would urge the alternate writer to convince yourself/prove it before code it up.

The proof is quite simple,
$S_n = S$
$S_{n+1} = (S)L[\{S\}]$
$S_{n+2} = \ldots$
and once you find the expression for S_{n+3} you will be able to convince yourself the claim.

So the solution is to find smallest n, such that, $|S_n| >= 100$. Then, you just pre generate S_1 to S_{n+3} and you just check if the given pattern is substring of S_i for $i \leq n$.

7 Problem E: Joe is Learning to Speak

Author: Jon Ander Gómez
Type: Data structures

The subsequences of n words is known as n-grams in computational linguistics and probability. We just need an efficient data structure able to store the n-grams up to the number that Joe can memorize. A hash table should be enough.

For each sentence we check if any word is unknown. After updating the 1-grams with the new words, we check 2-grams, 3-grams, etc.

More information:

- N-grams: http://en.wikipedia.org/wiki/N-gram
- Prof Jurafsky's book: Speech and Language Processing. 2nd Edition.
- Search StackOverflow for interesting discussions about hashing functions.

8 Problem F: Odd and Even Zeroes

Author: Shahriar Manzoor
Type: Number Theory

First of all, computing the number of trailing zeroes in the $n!$ is counting how many times 5 and all its powers lower than n are contained in n, the sum is the number trailing zeroes. For instance, if we have $n = 99$, 99! will have the following number of trailing zeroes:

$(5 * 19 \leq 99)$ and $(25 * 3 \leq 99) \Rightarrow 19 + 3 = 22$ trailing zeroes.

If we observe the evolution of the number of trailing zeroes we can see the following:

```
0 0 1 0
1 0 2 0
2 0 3 0
3 0 4 0
4 0 5 0
5 1 5 1
6 1 5 1
7 1 5 1
8 1 5 1
9 1 5 1
10 2 6 0
11 2 7 0
12 2 8 0
13 2 9 0
14 2 10 0
15 3 10 1
16 3 10 1
17 3 10 1
18 3 10 1
19 3 10 1
20 4 11 0
21 4 12 0
22 4 13 0
23 4 14 0
24 4 15 0
25 6 16 0
26 6 17 0
27 6 18 0
28 6 19 0
29 6 20 0
30 7 20 1
31 7 20 1
32 7 20 1
33 7 20 1
34 7 20 1
35 8 21 0
36 8 22 0
```

where the first column is the value of n, the second one the number of trailing zeroes in $n!$, the third one the counter of numbers whose quantity of trailing zeroes in their factorial is even. The last column is the second one modulus 2. The number of zeroes in the fourth column up to n is the value we are looking for.

The way this sequence of 0s and 1s evolves is what we need to find out in order to compute the result value given n in a fast way. Let's see some initial blocks of this sequence, each line contains 125 digits and each block 625 digits:

```
   0 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000
 125 11111000001111100000111111111100000111110000011111111111000001111100000111111111110000011111000001111111111100000111110000011111
 250 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000
 375 11111000001111100000111111111100000111110000011111111111000001111100000111111111110000011111000001111111111100000111110000011111
 500 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000

 625 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000
 750 11111000001111100000111111111100000111110000011111111111000001111100000111111111110000011111000001111111111100000111110000011111
 875 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000
1000 11111000001111100000111111111100000111110000011111111111000001111100000111111111110000011111000001111111111100000111110000011111
1125 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000

1250 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000
1375 11111000001111100000111111111100000111110000011111111111000001111100000111111111110000011111000001111111111100000111110000011111
1500 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000
1625 11111000001111100000111111111100000111110000011111111111000001111100000111111111110000011111000001111111111100000111110000011111
1750 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000

1875 00000111110000011111000000000011111000001111100000000000111110000011111000000000001111100000111110000000000011111000001111100000
2000 11111000001111100000111111111100000111110000011111111111000001111100000111111111110000011111000001111111111100000111110000011111
```

```
2125 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000
2250 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111
2375 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000

2500 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000
2625 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111
2750 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000
2875 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111
3000 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000

3125 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111
3250 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000
3375 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111
3500 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000
3625 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111

3750 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111
3875 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000
4000 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111
4125 00000111110000011111000000000011111000001111100000000001111100000111110000000000111110000011111000000000011111000001111100000
4250 11111000001111100000111111111100000111110000011111111111000001111100000111111111100000111110000011111111110000011111000001111
```

It can be observed that blocks of a power of 25, in this case 25^2, follow a pattern, every 10 blocks the second 5 are simmetric respect of the first 5 blocks. This behaviour is repeated independently of the power of 25, for 25^1 it can be observed the same behaviour.

```
0000011111000001111100000 0000011111000001111100000 0000011111000001111100000 0000011111000001111100000 0000011111000001111100000
1111100000111110000011111 1111100000111110000011111 1111100000111110000011111 1111100000111110000011111 1111100000111110000011111
```

Well, now we have to deduce the amount of zeroes at each block corresponding to a power of 25. The formula is

$$x_i = \frac{2}{5} \cdot 25^i + 5 \cdot x_{i-1}$$

we also have to keep track of the parity in order to use x_i or $25^i - x_i$ as the number of zeores (numbers whose factorial has an even number of trailing zeroes).

Then, given n we have to substract 25^i while $n > 25^i$ and accumulate x_i or $25^i - x_i$ depending on the block. This can be discovered thanks to the expression ($j \bmod 10 \leq 4$) combined with the parity derived from 25^{i+1}. j is the block index at 25^i. Initially parity should be zero. Once n gets lower than 25^i we go down to 25^{i-1} until 25^0.

9 Problem G: Vivo Parc

Author: Emilio Vivancos
Type: Graph coloring

This is an instance of the graph coloring problem where:

- one enclosure is represented by a node.
- there is a undirected vertex from node "a" to node "b" if enclosure "a" can be seen from enclosure "b".
- Each species is represented by one color (form 1 to 4).

Due to the low number of nodes, it can be solved using with a backtracking approach.

More information:

- Graph coloring at Wikipedia http://en.wikipedia.org/wiki/Graph_coloring

10 Problem H: Binary Tree

Author: Mahbubul Hasan
Type: Dynamic Programming

First lets solve the problem ignoring U instructions. For each element of the T sequence note the next right children (*next R*), and next left children (*next L*). Then we can solve this with dynammic programming. Recurrence would be: dp[next L] + dp[next R]. Since either I will goto left / right and same subproblem.

Finally, for each U, we will go up, and use the dp of the previous problem.

11 Problem I: Trending Topic

Author: Jon Ander Gómez
Type: Data structures

Given the amount of words that may appear in the test cases we need a fast way to convert each word into a integer and perform all the calculations using numbers instead of strings. A map, hash map or trie is needed for this transformation.

Once we only have numbers, we can use a heap to maintain the ranking of topics. When asked for the trending topics, we can extract the top K words and all the words the same frequency as the K-th one. Note that we need a way to update the words inside the heap because we have to remove the words that appeared 7 days ago.

A slightly faster solution is to have a counter of the times that each word appears and the list of words in the last 7 days to update the counters appropriately. When we find a <*top k*> query, we can calculate the threshold in linear time and then print all the words above this threshold.

12 Problem J: Cleaning the Hallway

Author: Mahbubul Hasan
Type: Geometry

Slow solution:

This is almost well known union of circle problems. But there is some hole in the circles. Both are concentric but that is not any special point. We slice up along the x axis. We draw vertical lines at, $x = x_i - r_i$, $x = x_i + r_i$ for any circle with center (x_i, y_i) and radius r, also at $x = x_j$ where (x_j, y_j) is intersection point of any two circles.

Note that, in any slice there is no intersection of the segments. So for each segment, we check all the circular arcs (up arc and down arc separately) and keep those that are inside the segment. then we sort these segments from $+y$ to $-y$. And in stack style we sum up the area.

Fast solution:

Suppose the outer circle is cw oriented and inner circle is ccw oriented. Now we will consider the circles separately. For each circle, we find intersection points with other rings (not circle). We omit the parts of the circle that are completely within some rings. To do so, we find out intersections with outer ring and inner ring, and the portion inside it is omitted. We find out all the $n-1$ such regions, sort them and find out the total non omitted part. $n \cdot \log n \cdot n = n^2 \cdot \log n$. For each of the non-omitted part we add up their signed sum.