Problem hints

March 26, 2017

One does not need to look twice at the scoreboard to understand that the contest was too hard. I put the two easiest problem at the start, problem A and B. However it seems that problem A is actually harder than what I evaluated it to be. Problem C was the hardest problem and to make things even worse, it seems easy when you read it. Problem D is tricky because the underlying graph is not acyclic and that is not very intuitive. Problem E is not very difficult but you need to have a good understanding of shortest paths.

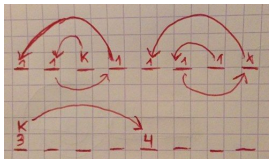
As suggested by Henri Devillez, I will just give hints and let you try to solve the problems. I think that you will learn more if you try to solve the problem by yourself. If even with them you cannot solve some of the problems don't hesitate to ask questions on the slack.

Problem A

First note that the answer is never impossible. Knowing this you can focus on finding a strategy to make all the frogs meet. I advise you to try to do it on paper for some small values of n and by first ignoring the king. Think about a spiral patters starting from the center.

Once you know how to solve n frogs without the king, notice that you can always solve the n frogs so that then end on the leftmost or rightmost position.

Now solve the problem in two step by first putting the king on the leftmost or rightmost position and then making the other pile as close as possible to the king pile so they can jump.



Problem B

Given the votes of Craig it is easy to evaluate which movie is selected. Since $n \leq 10$ you can just loop over all permutations of votes and check which one is the best. If you don't know it, the next permutation algorithm is implemented in C++.

`http://www.cplusplus.com/reference/algorithm/next_permutation/`

It is also possible to do something more efficient, I will let you think about that.

Problem C

This is by far the hardest problem. Solving it involves knowing some graph matching theory or Edmonds algorithm for general matchings. What makes it even more tricky is that it looks like a simple BFS or DFS should be enough so it is easy to get trapped into this problem with a completely wrong solution.

First notice that each location is connected to at most one river. Therefore the rivers form a matching on the graph. Finding a path between two hills that alternated between dirt and river actually means that you need to find an *augmenting path* for the given matching.

Now, if you know anything about matchings, you know that a simple BFS can be used to find an augmenting path when the graph is *bipartite*. A graph is bipartite if and only if it contains not cycles of odd length. Therefore a simple BFS will get AC as long as the input graph does not contain odd alternating cycles.

So if there are no such cycles BFS will work. Otherwise, think about what happens when contract such a cycle into a single node. Can we always convert an alternating path in the contracted graph into one in the original one?

`http://mathworld.wolfram.com/VertexContraction.html`

Problem D

This is clearly a longest path problem. The nodes are the dices and there is an edge between D_1 and D_2 iff D_2 is better than D_1. A lot of people tried to use DP to solve this. However the graph is not acyclic. For instance:

`2 3 3 3 6 6 < 1 4 4 4 4 5 < 1 1 5 5 5 5 < 2 3 3 3 6 6 6`

Build the graph and compute the SCC's. Check how many there are and how big they are. Now how can put things together to solve the problem?

Problem E

Compute all shortest path DAG's using for instance Floyd Warshall's algorithm. Start on the DAG rooted at the first node of the path. Then you can greedily compute minimal segmentation by changing the current DAG as late as possible. I let you think about what are the conditions that make you change the current DAG.