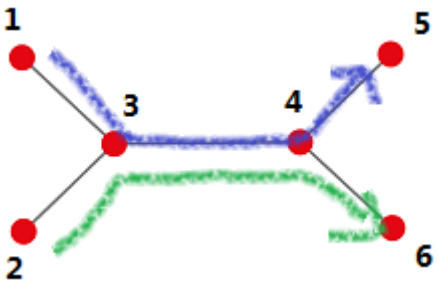
The editorial of 'Annual Prarde'

by Gaoyuan Chen (cgy4ever)
2012-08-22

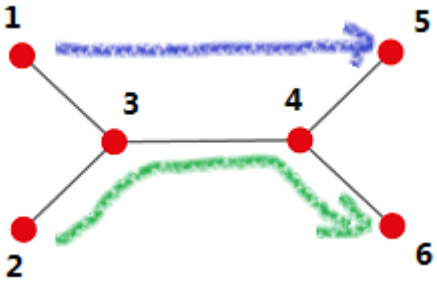
The problem asks us to put paths on a directed graph, and then a "cost" will be calculated as a function which is a little bit complicated. The aim is to make this "cost" as small as possible.

Part 1

First let us make the problem easier: In the original problem, a city can be visited by more than one hero, and can be visited many times by one hero. For example, let's focus on this case:



Two heroes both visited city 3 and 4. Can we avoid this case happen?



Yes, we can. For each pair of (i, j), let dist[i][j] be the length of minimal path from i to j in the original graph(or infinite if such path didn't exist), we add a new path between i and j with the length dist[i][j].

On one hand, any configuration in the original graph will correspond to one configuration in the new graph, and each city can be visited at most once. On the other hand, any configuration

in the new graph can correspond to a configuration in original solution. So the minimal answer in both graph will be equal.

Then the problem becomes: for each city, we can only visited it once.

Part 2

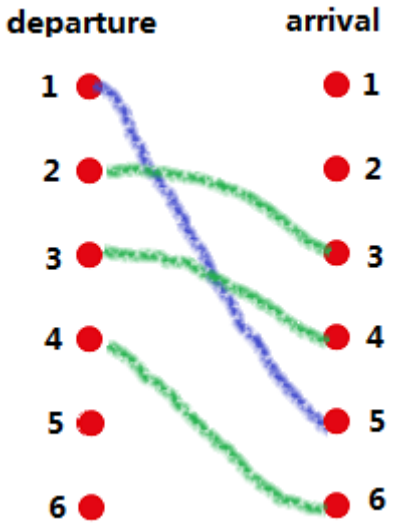
The next step is to understand the "cost". First we have some observations:

1. The configuration will be: a set of edges in the new graph. For example, in above figure, we choose these edges: 2->3, 3->4, 4->6, 1->5.

2. A set of edges will correspond to one configuration, if and only if: for each vertex v, the number of edge goes into v, and the number of edges goes from v, both will be not grater than 1.

3. let M be the number of edges in the set we selected. then we will pay exactly C * M dollars beside the cost of moving on edges.

Then this model is very similar to "min path cover" problem, so we can solve this by correspond the "edges" to a matching. For example, the configuration above will correspond to this matching:



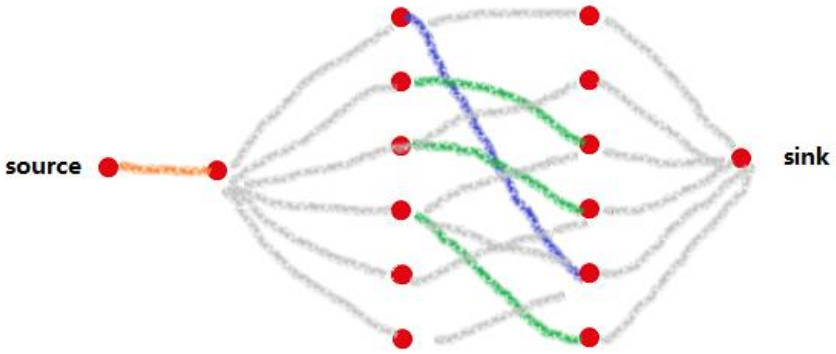
So the cost will be: all 4 edges length + 2 * C. Where 2 is (n –

number of edges in this matching).

If K is small, we can try to solve this problem by try all possible number of edges in matching with minimal cost max flow. But now K can be very large, so we can't redo this work K times.

Part 3

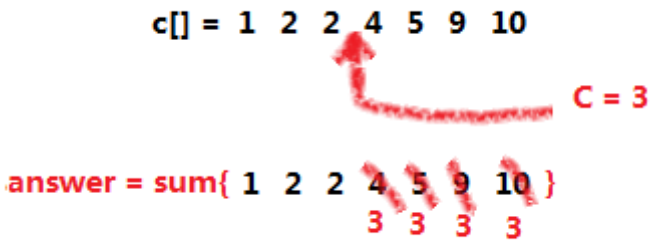
Now let think about the process of the flow algorithm:



Each time we extend the flow by 1, with a cost. Let c[i] be the cost of i-th extend flow. We know: $c[1] \leqslant c[2] \leqslant \ldots \leqslant c[n]$.

And the answer will be: min{c[1] + ... + c[k] + (n-k) * C} where k can be 0 to n, inclusive.

So we can pre-calculated the array c. Then for each C, we just replace all elements in c by C where its value larger than C. For example:



Now we have solved this problem, and:

Part 1 can run in $O(N^3)$ with Floyd-Warshall algorithm.

Part 2 is a minimal cost flow problem with size $|V| = O(N)$, $|E| = O(N^2)$.

Part3 can be solved in O(K * logN) with binary search.