**Assignment 3**

z5142340 Haiyu LYU

1. We build a hash table called that can save previous dam. For example. is a dam place and is a dam just in front of . need to meet the conditions: and . If can not satisfy the conditions. We check if meet the above restrictions. Repeat this algorithm until you find a dam that meets the restrictions or no such dam. There are two result in. The first one is ( is the number of dams in front of and meeting distance conditions). The second result is that there is no meet , so . For each dam , we use this algorithm to get hash table .

We set optimal subset as table . We set and the first dam . The pseudocode of the algorithm is as follows:

(a). Case1: There is no pebble in column. (1 pattern)

Case2: The column only contains 1 pebble (1000, 0100,0010,0001). (4 patterns)

Case3: The column may contain 2 pebbles (1010,1001,0101). (3 patterns)

Hence, there are 8 legal patterns that can occur in any column.

(b). Because there are 8 ways in total. We assume they are and those value is .

Suppose we have two ways, and , are to the left of and both of them are legal.

Suppose we have a table C, which is our best placement solution. We assume that is the optimal placement for and then we should put the value of in column . The best is the max ( is legal for ). We have:

Calculating each and take , and the backtracking take . Therefore, this algorithm runs in overall.

1. Firstly, we should use merge sort to sort heights and sort . Then we should create a 2-D table called subset which size is . It is necessary to initialize this table before running the algorithm. we put 0 into first row and first column, like this:

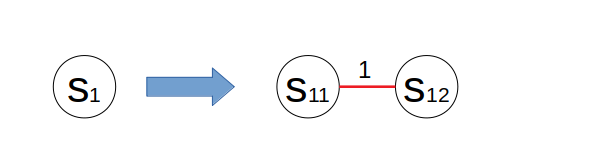
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| lengths  heights | 0 | 1 | …… | m |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 |  |  |  |
| …… | 0 |  |  |  |
| n | 0 |  |  |  |

The pseudocode of the algorithm is as follows:

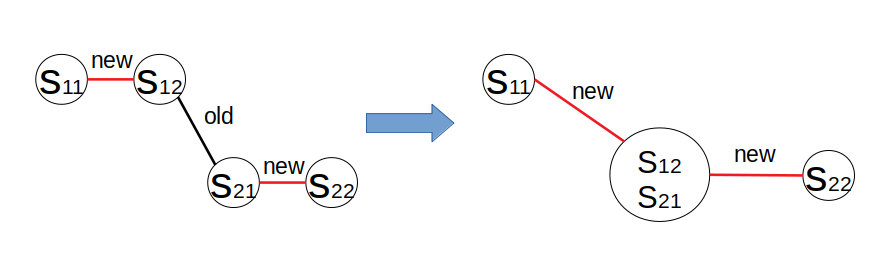
(a). We can see each spy as a vertex and each channel as edge in network. The original problem that find the fewest number of channels becomes that finding the min-cut. In order to find it, we just need to use Edmonds-Karp algorithm to calculate max-flow. We assume the capacity of each edge as 1. After calculate the max-flow we can confirm min-cut. Because the edges in network flow is the channel. We need to compromise the edges which are crossed by min-cut. Finally, we get the fewest number of channels that satisfies the conditions.

(b). At first, we should check if there is an edge between. If there is such an edge, then we will have no solution.

We assume that there are N vertex (except S and T), and M edge in network flow. We separate one vertex to two vertexes. For instance, vertex is separated to and . We add a new edge between two new vertexes and the capacity of this edge is 1.



Now we have N new edges and M old edges. Old edges connect original vertexes, like s1-s2. So we need to delete old edges by merging 2 vertexes. The method is shown below.



After merging vertexes, we only consider the new edge in network flow. We can also use E-K algorithm to get the max-flow and min-cut. Because those edges represent vertexes. Therefore, we can bribe those vertexes(spies).

1. Firstly, we add two edges . Then we can use Fold-Fulkerson method for finding maximal flow and minimal cut one the new network. Using this method to achieve that vertex u is at the same side of the cut as the source s and vertex v is at the same side as sink t.