COMP9101 ASS3

Z5141180 Yizheng Ying

1. Subproblem : find out the largest number of dams that can be bulit in meters from head.

Recursion : 

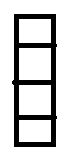
If we build a dam at , so we need to find the larget number of dams within  meters from head, the result should be 

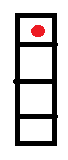
If there is no dam at , we need to find out the larget number of dams within  meters, the result should be 

Edge case : 

Time complexity: 

1. 8 legal patterns that can occur in any column.

The first [situation](C:/Program%20Files%20(x86)/Youdao/Dict/8.5.2.0/resultui/html/index.html" \l "/javascript:;):, all the squares are empty.

The second [situation](C:/Program%20Files%20(x86)/Youdao/Dict/8.5.2.0/resultui/html/index.html" \l "/javascript:;):,,,, in each case ,there is only one pebble placed in each column, there are four cases.

The third [situation](C:/Program%20Files%20(x86)/Youdao/Dict/8.5.2.0/resultui/html/index.html" \l "/javascript:;):,,, in each case, there are two pebble placed in each column, there are three this kind of cases.

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

1. Subproblem : get the max intergers in the square that are covered by pebbles in ith column

which is compatible with the i-1th column.

We set the location of pebbles on one column using a four bit vector, , where  iff there is a pebble on row i in this column. Then there are eight feasible binary patterns. We will use the notation  to denote the ith bit of  where .

Recursion: opt(comp(i),i)=max{opt(comp(i-1),i-1)+}, where , maximum score for placing pebbles in the first i−1 columns, and placing pebbles in column i according to the pattern , chose the max sum and comp(i) means the compatible patten in ith column with i-1th column.

Edge case : 

Time complexity: 

1. Firstly, we sort team members based on their height in non-decreasing order and sort skis based on the length in non-decreasing order.

Case 1: n=m

Assign the member and skis one by one in order is the optimal.

Proof: If we exchange any skis or member’s order, the result won’t less than the optimal strategy.

If 



If 



If 



If 



If 



Only when  and , the exchange makes no change. In other three cases, the change one becomes larger. So our method is the optimal one.

Case 2: n<m

Subproblem: Assign the ith member with the jth skis and the sum of difference between height and length is the smallest.

If i =j, we need to  where i=j.

Recursion: opt(i,j)= where j>i.

when the ith member chooses the jth skis, we get opt(i-1,j-1)+|h(i)-l(j)| and when the ith member does not choose the jth skis, we get opt(i,j-1).

Edge case : i=1 or j=1.

1. We can use Max Flow-Mini Cut algorithm. We assume that each spy as a vertex and each channel as an edge. Then add a super source S and a super sink T, source S only has flow out and sink T only has flow in. The capacity of each edge is 1 so that min cut equals tonumber of edges crossing the cut.
2. Because we cannot compromise channel so that we assume that the capacity of each edge is infinite. For example: the red line mean the capacity is infinite

A

C

D

B

E

We can assume that each vertex as two vertexes and linked by an edge which capacity is 1, like the following figure:

1

1

A’’

C’

D’

B’’

E’

A’

B’

C’’

D’’

E’’

1

1

1

So that we can get the figure with each edge capacity is 1:

1

A’

A’’-C’/B’’-C’

D’’

B’

E’’

1

1

C’’-D’/C’’-E’

1

1

So that we can use Max Flow-Mini Cut algorithm to get the number of spies that we need to bribe.

However, when S can communicate with T directly by a channel, there is no solution.

1. We add the edge from s to u and v to t respectively and the capacity are both infinite. Then we can use Max Flow-Mini Cut algorithm to find a minimum s−t cut. Another option is to use a supersource connected to s and u and a supersink connected to t and v by edges of infinite capacity.