3121 assignment3

Q1

Subproblem: among item from 1… I pick the subset maximum dams.

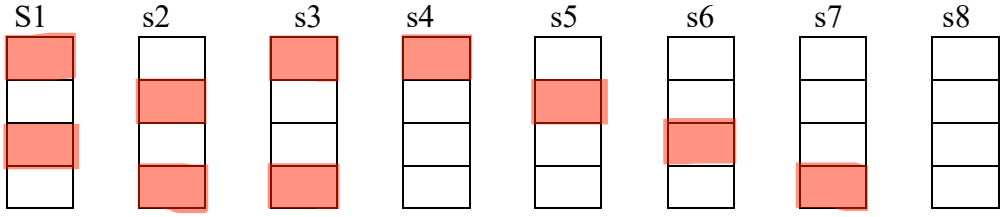
1. Make an array for M[n] and opt[n].
2. Divide it into two kinds of situation, one is include the xi, another is not include xi
3. The distance of xi legal distance is 2*ri from xi-r1 to xi+ri.
   for example, M[1] = [x1-r1,x1+r1] ,then opt(1) = 1.
   M[2] = [x2-r2,x2+r2] (constrains x2-x1>=r1 and r2), opt(2) = 2,otherwise opt(2) = opt(1).
   M[3] constrains with M[1] and M[2], if not satisfied constrains, max(opt(2),opt(1)), if not max(opt(2),opt(1))+1 ……
   Then, we can conclude that:
4. For include xi, we should choose {max(opt(xi-1),opt(xi-2)…opt(1))+1}
   For not include xi, we should choose {max(opt(xi-1),opt(xi-2)…opt(1))}
   In order to decrease the time complexity, we will store all the optimal solution in opt array.
   Some code prove:
   dam():

```
for I in n :          //O(n)
    find satisfied j from M[1]-M[i]    //O(n)
        opt[i] = Max(opt(i),opt[j]+1).
```

Thus, time complexity for this algorithm is o(n^2).

Q2

A. We have those patterns. If pebble will be place on Columns. We only have 8 kinds of patterns. I will give the 4 columns as example:



The red pattern is pebble possible be place in columns.

B. Subproblem: among item 1…I, pickup subset maximum total value with constrains a placement of pebbles to be legal, no two of them can be on horizontally or vertically adjacent squares.

1. Make an array option[8]. //This is optimal solution which store in array
2. Question A is prepare for 8 situations.
3. Maximum sum include above situation for opt(next)+ current total value(sum all square integer)
4. for example, opt(2) = max(opt(1)+value(s1),…,opt(1)+value(s8)) . However, the pattern compatible between opt(1) and opt(2) is fixed , if current opt(2) choose s1 ,opt(1) is possible match s2,s5,s7,s8 . Then, opt(1) is maximum of s2,s5,s7,s8. Therefore, the opt(2) is value of s1+max(s2,s5,s7,s8).

   Some code prove:

   For I in n

   opt(i) = Max(opt(i-1)+value(s1),opt(i-1)+value(s2)……opt(i-1)+0))

   Thus, time complexity is O(8*n), overall it is O(n).

   Q3

   Subproblem: among 1…I, pickup minimum absolute value between length of skis and height of people.

1. Sort the h[n] and l[m] by using merge sort in increasing order.
2. Make an array opt[N][M] which is optimal cost with first j pairs skis and first i student.
3. Prove the hi<hj and li<lj exist optimal solution if and only if |hi-li|,|hj-lj|
   I separate it into to two case one is hi<li<lj<hj and hi<li<hj<lj, if the two case exist optimal solution which assign with hi to li and hj to lj

   Case 1: If we match hi to li and hj to lj and l_i to h_j and l_j to h_i then hi-li+lj-hi = (lj-li)-(hj-hi) and $h_j - l_i + l_j - h_i = (l_j - l_i) + (h_j - h_i)$. As $(h_j - h_i) > 0$, the matching (lj-li)-(hj-hi)< $(l_j - l_i) + (h_j - h_i)$

   Case 2: If we match l_i to h_i and l_j to h_j and l_i to h_j and l_j to h_i then $h_i - l_i + h_j - l_j = (h_j - l_i) - (l_j - h_i)$ and $h_j - l_i + l_j - h_i = (h_j - l_i) + (l_j - h_i)$. As $(l_j - h_i) \geq 0$, the matching $(h_j - l_i) - (l_j - h_i) < (h_j - l_i) + (l_j - h_i)$

   Similarly, Sort the list l and h because we will not have si match lj and sj match li. So, there are optimal solution where array l match the array h.
   Some code prove:
   For I in n

For j in m
opt[i][j] = Min(opt[i][j-1], opt[i-1][j-1]+abs(h[i]-l[j])).

Thus, time complexity is O(nlogn)+O(mlogm)+O(mn)

Q4

A.
1. make a model for the S s0-sn and T, the most suitable model is direction graph. Let spies become vector and channel become edge. S always out and T always in. Spy is not identity, so the weight of channel between two spy are not unique. Then, we can suppose the weight of edge is one.
2. we are simplify the problem with maxflow problem. We can use minimum cut to determine the minimum number of edges cross the cut. We can find the shortest path to longest path by use Edmonds-Karp algorithm. Every time find the augmentation path, update to every edges and find next one until all the path have been found. Then, the maximum flow is the minimum path. Thus, the number of cut is fewest number of channels that we should listened.
B.
the problem to find the minimum number that we should bribe spies. It is similar with last question. We just need change the model with one vector to two vector and link two vector with one edges(except S and T).According to the question we should not have direct link with S and T. The weight of the edges are not change. Then, use maximum flow to find where is minimum cut. The number of edges is the number of spies should be bribe.

Q5
We can make a flow graph by adding two edges: s → u and v → t, suppose the two edges infinite capacity. We can use maxflow and minimum cut algorithm, finding 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. Thus, fins number of the smallest possible capacity among all cuts.