COMP9101 Assignment 3

Name: Xuhua Le
Student Number: z5047516

Question1.

Solution 1 (DP):

We could denote these N dams as: $d_1, d_2, ..., d_N$.
According to the question, we could think in this way: the i^{th} dam d_i starts at $(x_i - r_i)$, ends at $(x_i + r_i)$ with covering length of $2r_i$.

Hence, our task: find a subset of compatible proposals of largest number.
Then, for every $i \leq N$, we solve the following subproblems:
Subproblem $P(i)$: find a subsequence σ_i of the sequence of dams $S_i =< d_1, d_2, ..., d_i >$ such that:
(1). σ_i consists of non-overlapping dams.
(2). σ_i ends with dam d_i (Used to simplify recusion).
(3). σ_i is of largest number among all subsequences of S_i which satisfy (1), (2).

Let $T(i)$ be the total number of the optimal solution $S(i)$ of the subproblem $P(i)$.
For $S(1)$, we choose dam d_1, thus $T(1) = 1$.
Recursion: assuming that we have solved sunproblems for all $j < i$ and stored them in a table, then we let:
$T(i) = max\{T(j) + 1 : j < i \ \& \ (x_j + r_j) < (x_i - r_i)\}$
In the table, besides $T(i)$, we also store j for which the above max is achieved.

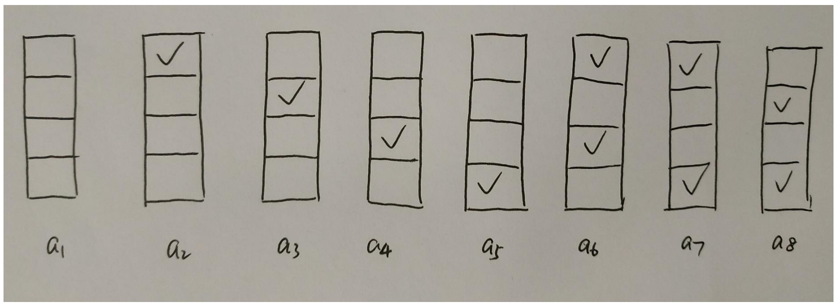
Solution 2 (Greedy Method):

According to Pizza, we could treat all the dams to points whose lengths are infinitely small, then this problem is similar to the first example in the lecture *The Greedy Method*: We have N activities a_i $1 \leq i \leq N$ with starting time $s_i = x_i - r_i$ and finishing time $f_i = x_i + r_i$, no two activities can take place simultaneously, and our task: find a maximum size subset of compatible activities.

$\Rightarrow$ Back to our problem, we could use the same greedy way in this question: we could treat i^{th} proposal as i^{th} activity with starting time $s_i = x_i - r_i$ and finishing time $f_i = x_i + r_i$. Then we use the same way: among the activities which do not conflict with the previously chosen activities, always choose the one with the earliest finishing time.

Question2.

(a).
According to the question, there are 8 types of legal patterns that can occur in any column, and denote them as: $a_1, a_2, ..., a_8$.
And the legal patterns are: 1 way for 0 pebble, 4 ways for 1 pebble, 3 ways for 2 pebbles.



(b).
Call two patterns a_i, a_j (a_i is on the left side of a_j) compatible if they can be placed on adjacent columns to form a legal placement.
We could build a table $T[i, c]$ which tells the optimal placement for columns c till n with the pattern in the column c is a_i.
Now, we have:
$T[i, c] = sum\_of\_integers(a_i) + \max_j T[j, c + 1]$,
where the maximum is taken over those patterns a_j which are compatible with a_i.

Question3.

Solution:
We could set a table $S[1, ..., n, 1, ..., m]$ with n rows (studenst) and m columns (skis), and sort the table in increasing order by the heights and lengths. At every stage i, j decide whether it's best to assign the j^{th} pair of skis to the i^{th} students or not.
We could build two lists, $M = [s_1, s_2, ..., s_m]$ denotes the sorted skis array and $N = [h_1, h_2, ..., h_n]$ denotes the sorted students array.

Firstly, to reset the whole table, from $i = 0$ to $i = n$, we set $S[i, 0] = 0$. From $j = 0$ to $j = m$, we set $S[0, j] = 0$.
If the number of students and skis are equal, this is the optimal situation, which means short students have short skis, tall students have long skis.
Here is the subproblem, we assume that the first i people already have the optimal solution, and now comes a new j skis, now we want to find an optimal solution for these i people and j skis, there comes 2 possible ways:
1. this new j skis are not chosen, i people remains their previous status.

2. this new j skis are chosen by some one.
we only have to find an optimal solution from these two possible solutions:
$S[i,j] = \min S[i,j-1], S[i-1,j-1] + |h_j = s_j|$.
We traverse from $i = 0$ to $i = n$ and $j = 0$ to $j = m$, and only when $i \leq j$ we apply the selection above, when we finish traversing we can have the optimal solution.

Question4.

(a).
We could treat this problem to a problem that tries to find MaxFlow/Min Cut in a directed graph.
Firstly, we need to model the graph: treat each spy s_i $(1 \leq i \leq n)$ as a node, and treat the channel between spies as the edge between the nodes, and we also have a source S and a sink T. Besides, no edge leaves the sink T and no edge enters the source S.
Then, a key problem is: what's the capacity on each edge so that the Min Cut value from the maxflow algorithm is equal to the number of edges of Min Cut? When the capacity of edge is 1, the Min Cut is equal to the number of edges comes through the cut.
After we have built the model, we could use standard Max Flow - Min Cut Algorithm.
When S sends a message to T, there must be a message flow flowing from the left side of the min cut to the right side of the min cut. Then the message flow must pass the channel on at least one min cut, because all the channels on the min cut have been listened. And these channels are the minimum number we need to listen to, which meets the problem's requirement.

(b).
In order to simplify the question we say that the model we make in (a) is G.
In this case we have to change the network in question (a) to run the standard Max Flow - Min Cut algorithm, the way to do is to replace all the vertex in G with edges whose capacity is 1 and replace each the edges in G with a vertex. The new model is called G^*.
In this way when we run Max Flow Min Cut algorithm on G^*, the edges that crossed the Min Cut will be one of the vertex in G, so by doing so we can Min Cut vertex in G, and these vertexes are those we have to bribe.

Question 5.

Solution:
We could modify the flow network by adding two edges: $s \to u$ and $v \to t$, both of infinite capacity. We then run our standard Max Flow - Min Cut Algorithm on the modified network, 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.