COMP 9101

Assignment 3

Student ID : z5174741

Student name : GUANQUN ZHOU

Q1:

For this question, I think the method mentioned in “Activity Selection with maximal total

duration”in lecture is the best option.

I will define **subproblem**:

for ) proposal, find a subset of dams

such that:

1. Satisfy the requirement that there is not another dam within meteres (upstream or

downstream).

1. ends with dam .
2. is of maximum total number of dams among all subset of dams which satisfy requirement 1 and 2

Let be the number of the dams of the optimal solution of subproblem .

So the **recursion** is:

After we find optimal solution for every ) proposal ,we need to go back

to look for the maximum among all proposals and finally find the optimal

solution:

**Prove of the optimality:**

Let the optimal solution of subproblem be the subset:

We claim: the truncated subset is an optimal solution to subproblem , where

If there were a subset of a larger total number of the dams than the number of subset and also ending with dam , we could obtain a sequence by extending the sequence with dam and obtain a solution for subproblem with a longer total number of the dams than the total number of dams of sequence, contradicting the optimality of .

**Time complexity**:  
the algorithm computes proposals and for each proposal, it needs to compute all preceding solutions in order to find the optimal solution.

Thus, the time complexity is .

Q2:

1. There are **8** possible patterns:
2. The empty patterns:

|  |
| --- |
|  |
| 1 |
|  |
|  |

1. **4** patterns where each pattern has exactly one pebble:

2

|  |
| --- |
|  |
|  |
|  |
|  |

3

|  |
| --- |
|  |
|  |
|  |
|  |

4

|  |
| --- |
|  |
|  |
|  |
|  |

5

|  |
| --- |
|  |
|  |
|  |
|  |

1. **3** patterns where each pattern has exactly two pebbles (on the first and third square, on the first and forth square, one the second and forth square):

|  |
| --- |
|  |
| 6 |
|  |
|  |

|  |
| --- |
|  |
|  |
| 7 |
|  |

8

|  |
| --- |
|  |
|  |
|  |
|  |

1. For this question, the 2D recursion method is the optimal option.

So I will build a 2D table with 8 legal patterns as the table rows and n columns.

Then I define the **subproblem**:

find the pattern which is the optimal solution for the column such that:  
1. is range from , is range from

2. The pattern is compatible with the pattern of ( column.

3. We can obtain the maximal sum of the integers in the squares that are covered by

pebbles.

So the **recursion** is:

First we need to define is the sum of the integers that are covered by pebbles of pattern  
we compute all possible patterns for column. In this way, we can obtain all possible values of and we can compute all sums of and corresponding optimal solution for Then we can choose the maximal sum from the results we obtain.

In other words:  
In every recursion, we will compute a for every possible pattern for column.

And every contains：  
1. The sum of the and corresponding optimal solution for .

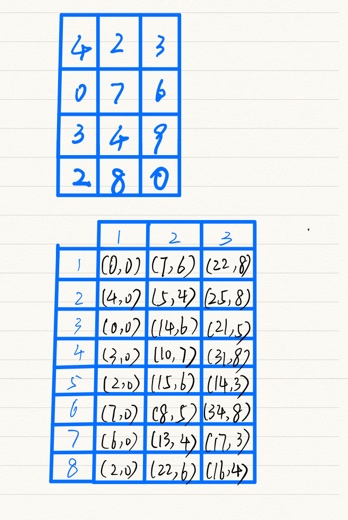
2. The index of the chosen pattern of so that for every possible pattern for column we can obtain the maximal sum.

We define that s is the maximal sum of first columns and the index of the pattern of column is .

We will compute s in this way:

We will compute in this way:

After obtaining the optimal solutions for the last column, we need to go back to look for the maximal sum among all the optimal solutions for the possible patterns for the last column and finally obtain the optimal placement.



The graph above is an example, and we say that the maximal sum of the integers in the squares that are covered by pebbles is 34 and the patterns chosen are 6-8-6.

The covered checkerboard should be like this:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Time Complexity:**

This algorithm go through all n columns. On each column, we need to consider 8 patterns. For every possible pattern, we need to calculate the optimal solution for this pattern by computing the sum of the value and corresponding previous optimal solution.

In the end, we need to go back to look for the maximal sum among all the optimal solutions for the possible patterns for the last column and finally obtain the optimal placement.

Thus, the time complexity is .

Q3:

For this question, the method mentioned in “Integer Knapsack Problem (no duplicate items)”would be the optimal option.

We need to build a 2D recursion to find the optimal solution.

First we define  for the height of skiers and for the length of skis.

Note that

Then we sort in increasing order.

Notice that if an assignment is optimal and then the skis assigned to skier should be shorter than the skis assigned to skier , otherwise you could swap their skis without increasing the sum of the absolute values of the difference between the heights of skiers and length of skis.

In other words:

If there exist and , then the optimal assignment is:

.

So the **subproblem** is :

finding the optimal assignment of first skis to first skiers such that:

is range from , is range from

So the **recursion** is:

If then there is only one assignment that assigns skis according to the skier’s height.

If :

We need to compute the subproblem:

= first skis and first skiers, such that is minimized.

So now we have two options:

**If** :

**then**

**else**

If , assign .

If , continue to

Final solution will be given by

Q4:

1. For this problem, we can treat it as a max flow problem.

Step 1:

We can treat as source and treat as sink.

Step 2:

We set the capacity of the edge starts from as infinity

capacity.

And we do the same to the weight of the edge ends in .

Step 3:

We set the capacity of each other edge as unit

capacity(1).

Step 4:

We use Edmond Karp algorithm to find the max flow.

Meanwhile, we can obtain the minimal cut. As we set the capacity of each other edge as unit capacity (1), so the minimal cut is equal to the number of edges which the minimal cut goes through, which means the minimal cut is equal to the number of channels we need to comprise.

1. For this problem, the method we need is similar to the one mentioned in (a).

What different is that we need to duplicate each spy vertex and connect the two vertices with an edge of a unit capacity(1).

And we can set the capacity of all other edges as 0.(In other word, we can treat the edges in original graph as vertices transformed from duplicate vertices).

Then we repeat the 4 steps in (a) and we will obtain the minimal cut finally.

After we have already obtained the minimal cut, since we duplicate each spy vertex and connect the two vertices with an edge of a unit capacity(1), so according to the minimal cut ,we can obtain the number of cut edges which connect two duplicate vertices. And that number is the number of the spies we have to bribe.

Q5:

For this problem, we need 3 steps.

Step1:

We add two edges: and .

Step2:

We set the capacity of both edges and as infinity

capacity.

Step3:

We treat this problem as max flow problem. We use Edmond Karp.

algorithm to find the max flow. Meanwhile, we can obtain the

minimal cut.