Elad Ohayon

4/24/2023

1. The brute force algorithm, in terms of dams to be evaluated, is M * N! where N = the number of dams to be evaluated and M is the cost to evaluate a sequence of dam constructions. This is because we must get all the possible sequences of dams and then measure the cost of each sequence individually to determine what the minimum cost across all sequences is.

2. My key level insight going into this assignment was that the problem seemed to be very similar to Floydd Warshall's algorithm meaning it has overlapping subproblems and has the same general substructure. Both require the use of a 2d matrix, and the cost to make a "move" is diagonally computed. To further elaborate, in my dp solution for this problem, there are a right and left subsets which are used to determine the cost of adding each dam. These subsets represent the right and left dams of the current dam being evaluated where damBuilt[i][j] represents the cost of building a dam between Y[i] and Y[j] where Y represents our dams array. My solution capitalizes on this by performing a dp to store results costs to reduce runtime. And to determine the next dam that should be performed by checking all next possible cuts against eachother.

3. The optimal substructure in this algorithm is the fact that the next most optimal dam to build depends on both previous dams that were made and upcoming dams. For example, if Riverend=1000 and Y= {250,500,750} the first most optimal dam is 500 as it both evenly divides the river with respect to the upcoming dams of 250 and 750. This leads us into the optimal subproblem which is that when computing the cost of a dam, we need to determine the

left and right boundaries and see which one minimizes the cost the most in the long run. We do this by storing all possible mincuts and selecting thr correct one each time through dynamic programming.

4. The recurrence relation for this algorithm goes as follows: the cost of building the dam from i,j is represented by damsBuilt[i,j] and dams represents the dams in ascending order. The base case is when i+1==j where the cost of building is 0. The recursive step is when i+1<j, in which case the cost of building the specific dam is calculated, we use a for loop to calculate the minimum cost across all dams and select building that one. The simplified recurrence relation is 0 if i==j-1, and min(previous,damArray[j] – damArray[i] + damsBuilt[i][k] + damsBuilt[k][j]); if k<j. Where previous is the previous cost of putting a dam and damArray is Y sorted with 0 and riverend added.

5. The cost of this algorithm is O($n^3$) where n represents the number dams to be evaluated. This is because it is an O(n^2) operation to determine the smallest and an O(n) check to determine the minimum cost to build a dam across all sequences. This is represented by the 3 for loops required to solve the problem.