DamConstruction Writeup

To solve this problem using brute force, one must examine every permutation of dams to figure out which order has the lowest inspection cost. The total number of permutations is $n!$ where n is the number of dams, so examining each of them would have an order of growth of $O(n!)$.

The key insight of my algorithm was that each dam, when inspected, splits the river in half, allowing me to inspect each half of the river separately. Let's say that each dam, including those already built at the edges of the river, has its y-location stored at its index in a `location` array. If we choose a dam index c and inspect it, it does not matter whether we inspect dams left of c (where index $n < c$) before dams right of c (where index $n > c$), or vice versa, so long as the dams on each side are inspected with the same order with respect to other dams on that side. This is because the right end of the river for all inspections left of c is no more than c, and the left end for all inspections right of c is no less than c. So, the dams left and right of c can be considered independent subproblems.

Given any two indices l and r, where $l < r$, which represent the already-inspected dams bordering the stretch of river on the left and right, and index c, representing the first dam being inspected, the optimal substructure is the cost of the left half of the river from indices l to c, the cost of the central dam index c being selected, and the cost of the right half of the river from c to r. This substructure exists for each dam indices l, c, and r that could be selected (where $l < c < r$), and at least one of those substructures must have minimum cost due to the well-ordering principle. The cost of the left and right subproblems could be calculated recursively, and c's cost is equal to `location[r] – location[l]`, the length of the stretch of river we are looking at right now.

Commented [1]: Once again, I am worried that I am doing this all wrong

Commented [2]: Also, I need notation

My recurrence is the following:

$$T(l,r) = \begin{cases} 0 & if\ (r-l) \leq 1 \\ min_{c \in (l,r)}\{T(l,c) + T(c,r)\} + (location[r] - location[l])\ otherwise \end{cases}$$

$T(l, r)$ is the minimum cost of the river inspection given any two dams as endpoints. `location[r] – location[l]` is the cost of inspecting dam c (unchanged no matter what c is). Each stretch of river can be divided into the cost left of c and the cost right of c, as I explained. The recurrence calculates the cost of inspecting both sides of the river for each c and finds a minimum. In my program, I kept a 2D array that used `table[l][r]` to store the value

of T(l, r). I calculated T(l, r) starting for each (l, r) where $r - l = 2$, continuing until $l = 0$ and $r = n + 1$ (the end of the river, due to the addition of the dam at 0 as index 0). My algorithm then returned `table[0][n + 1]`.

The order of growth for my algorithm is $O(n^3)$. It inspects every pair of (l, r), where $l < r$. There are $n + 2$ dams to examine, since the dams between the river's endpoints must also be inspected, so examining every pair of dams costs $O(n^2)$. For each dam pair (l, r), I examined every dam between them, with a cost of $O(r - l)$, which can never be more than $O(n)$, for a total order of growth of $O(n^3)$.