Assignment: Using DP To Save Money Building Dams

Avraham Leff

April 17, 2023

1 Assignment-Specific Packaging

The general packaging is unchanged from the basic "Homework Policies" document (see Piazza posting).

This assignment's "DIR" **must be named** *DamConstruction*, and your file **must be named** *DamConstruction.pdf*

2 Motivation

This assignment will give you more experience with:

- Devising a DP algorithm
- Programming a DP implementation
- Analyzing the algorithm's performance

3 Background

You work for a construction company that has been awarded a government contract to build a series of dams on a long river region (such as this). If you think of the river as a two-dimensional region, the series of dams can be abstracted as a set of line-segments at various y-value specifications. The contract specifies precisely these y-values, e.g., as an array of integers Y.

Construction is about to begin when the WPA (*Wildlife Protection Authority*) intervenes, claiming that each dam may cause irreparable damage

to the river environment. The WPA requires that each dam be evaluated with respect to its potential environmental damage.

Importantly: the *overall cost* of such evaluations may depend on the order in which they're performed. This because the cost of evaluating a single one of the proposed dam is proportional (for the purposes of this assignment, replace "proportional" with "equal") to ***the length of the undammed area*** over which the evaluation takes place. This is because, once a dam has been built, it separates the previously contiguous regions from one another (amazingly, the WPA is OK with that environmental effect!). As it happens, dams have previously been constructed at either end of the river ("river beginning" at $y = 0$, and "river end" at position $y = riverEnd$) so it's possible to determine the cost incurred by dam construction at any of the specified positions.

Example: if the length of the river is 1000 miles, and you construct a dam at $y = 500$, the WPA evaluation will cost 1000 (because the entire river length must be evaluated). A subsequent evaluation at $y = 250$, however, will cost only 500. A little thought should convince you that, for more complex Y, the overall cost of the evaluation can differ depending on the sequence in which you evaluate individual dam instances.

4 The Problem

Your task: devise a **DP algorithm** that, given the above problem parameters, returns the minimum overall evaluation cost for the dam construction.

5 Requirements

This assignment contains both non-programming and programming components.

5.1 Non-Programming Work

This component is weighted as 50% of your grade.

Your writeup file must contain, in the following order, the following sections.

1. Using a counting argument, what is the cost of a "brute force" algorithm in terms of the number of dams to be evaluated?

2. (**1-2 paragraphs**): a high-level explanation of the "key insight(s)" that underlay your DP solution for this problem. Introduce precise notation as necessary.
3. Now proceed to the next level of detail: based on your DP insight(s) above, describe the *optimal substructure with overlapping subproblems* in this problem.
4. Now proceed to the next level of detail: define a recursive formulation and present the recurrence (using the notation from the previous step)
5. What is the performance of your algorithm in terms of the number of dams to be evaluated? Justify your answer!

Your write-up **cannot exceed** two pages of size 11 font print. Diagrams are welcome!

5.2 Programming Work

This component is weighted as 50% of your grade.

- You **must use a DP technique** to implement a solution
- You may use either top-down or bottom-up styles of DP in your implementation

5.2.1 Programming A Solution

> Please review the general requirements for a programming assignment! I've tried to reduce the chances of "mistakes" occurring through the use of a "skeleton class", but ultimately, **you are responsible** for ensuring that I can compile and test your code without incident.

Begin by downloading the `DamConstruction.java` skeleton class (in the *DamConstruction* directory) from this git repository. Read the Javadoc carefully, then implement the API.

Note: for this assignment, you can assume that the test inputs are such that a dynamic programming solution can be completed in "reasonable" amount of time (on my machine, safely within 500 milliseconds).

Also: an individual test case will use a Y whose length is < 55.

- You may (encouraged to) "research" the concept of "dynamic programming" algorithms in general.
- You may not "research" this specific problem in any way!
- You may only use code written by yourself (and the JDK) to implement a solution.

5.2.2 Sample API Usage

```
  @Test
  public void demo() {
    final int[] Y = {50};
    final int riverEnd = 100;
    Arrays.sort(Y);
    final int[] evaluationSequence = {50};
    final int expectedSolution = 100;
    logAndInvoke(Y, riverEnd, expectedSolution, evaluationSequence);
  }

private void logAndInvoke(final int[] Y, final int riverEnd,
                          final int expectedSolution,
                          final int[] evaluationSequence)
{
  final SoftAssert softAssert = new SoftAssert();
  try {
    logger.info("Constructing DamConstruction instance with Y={} and "+
                "riverEnd={}", Arrays.toString(Y), riverEnd);
    final DamConstruction dc = new DamConstruction(Y, riverEnd);
    logger.debug("Invoking dc.solve()");
    final int answer = dc.solve();
    logger.info("Implementation returned {}", answer);
    softAssert.
      assertEquals(answer, expectedSolution, "mismatch on solveIt answer")
          ;
    logger.info("Invoking dc.cost({})", evaluationSequence);
    final int cost = dc.cost(evaluationSequence);
    logger.info("Implementation returned cost = {}", cost);
    softAssert
      .assertEquals(cost, expectedSolution, "mismatch on cost answer");
```

```
  } // try
  catch (Exception e) {
    logger.error("Unexpected problem", e);
    throw e;
  }
  finally {
    softAssert.assertAll("Results of test case");
  }
}
```

```
[INFO ]   TestNGTestListener onTestStart - Started test demo
[INFO ]   DamConstructionTest logAndInvoke - Constructing DamConstruction
instance with Y=[50] and riverEnd=100
[INFO ]   DamConstructionTest logAndInvoke - Implementation returned 100
[INFO ]   DamConstructionTest logAndInvoke - Invoking dc.cost([50])
[INFO ]   DamConstructionTest logAndInvoke - Implementation returned cost = 100
[INFO ]   TestNGTestListener onTestSuccess - >> Test demo succceeded (took 23 ms)
```