Burak Bozdağ
150170110

BLG336E - Analysis of Algorithms II, Spring 2020

Homework 3 Report

Test Selection and Prioritization with Dynamic Programming

Part 1

In this part, I designed an algorithm in order to select a subset of test suites. I developed a dynamic programming solution that selects a subset of test cases by considering the two criteria:

1) The number of previously discovered bugs by a test suite,
2) The running time of a test suite.

The output of my program contains the optimum set of selected test suites and the number of previously discovered bugs by running those test suites.

I used an optimization function for my algorithm. The mathematical representation of the optimization function is as follows:

$$\text{Maximize } \sum_{i=1}^{n}(v_i x_i)$$

$$\text{Subject to } \sum_{i=1}^{n}(w_i x_i) \leq W$$

Where v_i = bugs, x_i = [0,1], W = available time, w_i = running time of the suite.

The worst-case scenario would be in O(nm) complexity. (m: while loop repeat)

This algorithm is based on greedy methods, there can be more optimized solutions but this algorithm will be enough for our applications.

My algorithm also works when the running times of the test suites are given as real numbers instead of discrete values. Because my program processes these values as double numbers.

Part 2

After part 1, I dynamically implemented a new feature to my program. I coded a dynamic programming solution for the test case prioritization problem.

My edit distance algorithm calculates the differences between the given pairs. The output of my program contains the ordered test cases for each selected suite in addition to the outputs of the first part.

I used Levenshtein edit distance algorithm and adapted it for arrays with numbers. The mathematical representation of the optimization function I used is given below:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Where a and b are arrays that are compared, i and j are array lengths respectively.

Burak Bozdağ
150170110

The complexity of my algorithm is $O(n^2)$ since n = the length of the array and we are comparing two arrays' elements.

Compilation and Running

To compile: *g++ 150170110.cpp -o a.out*

To run: *./a.out data.txt*