ILLINOIS TECH

Most Profitable Purchase

CS 430 Introduction to Algorithms

Instructor: Lan Yao

By

Krishna Chatrathi (A20520106)
Hemanth Vennelakanti (A20526563)

Contents

1. Introduction

In the world of retail, pricing strategies often involve offering promotions and discounts to attract customers and increase sales. This is especially true for items that are commonly sold in bulk or as part of combinations. For instance, a retailer might give customers who buy more than one of the same items a discount or offer them exclusive offers when they buy several different items at once.

Retailers must have a clear understanding of the most cost-effective way to make purchases in order to manage pricing and promotions effectively. This requires determining the best combination of items and promotions that result in the lowest overall payment.

2. Objective

In this report, we present a solution to this problem through the design and implementation of a program that calculates the minimal payment for a purchase

3. Algorithm Design

- We first initialized a result object with the price of buying the items outright, an empty array to hold the promotions used to reach the minimum cost, and a copy of the original quantity requirements
- The Algorithm then goes through every special offer in the array iteratively and determines if it can be applied to the current set of required items.The function calls itself **recursively** with the updated quantities required, calculates the price of the products using the current special offer, and calculates the price of the remaining items recursively.
- The result object is updated with the new minimum cost, the promotions used, and the updated quantities if the cost of the current special offer plus the cost of the remaining items is less than the current minimum cost.
- The function returns an object with the final results, including the minimum cost, the promotions used to get there, and the quantities required.
- We used recursive approach to calculate the

4. Algorithm Implementation

```
const profitablePurchase = function (price, special, needs) {
  let ret = {
    cost: price.reduce((acc, e, i) => acc + price[i] * needs[i], 0),
    promotions_applied: [],
    quantities: needs.slice(),
  };

  for (let [idx, s] of special.entries()) {
    let sPrice = [...s].pop();
    let negativeFlag = false;
    let nextNeeds = needs.map((e, i) => {
      let val = e - s[i];
      if (val < 0) negativeFlag = true;
      return val;
    });
    if (!negativeFlag) {
      let subResult = profitablePurchase(price, special, nextNeeds);
      if (sPrice + subResult.cost < ret.cost) {
        ret.cost = sPrice + subResult.cost;
        ret.promotions_applied = [s, ...subResult.promotions_applied];
        ret.quantities = subResult.quantities;
      }
    }
  }

  return ret;
};
```

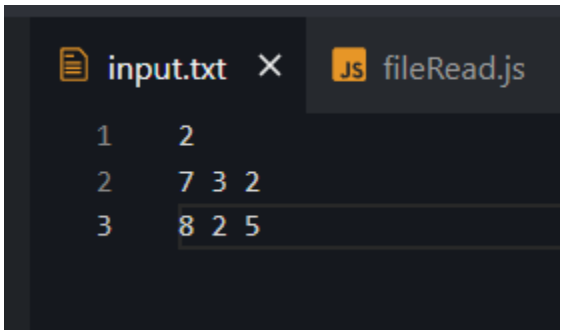
5. Time Complexity

- Let N be the number of items and M be the number of promotions.
- The main part of the algorithm iterates through each promotion in the special array using a for...of loop. Within this loop, the algorithm performs some constant time operations, such as populating the nextNeeds array and making comparisons, and then recursively calls itself with nextNeeds as the argument.
- The recursion continues until all possible combinations of promotions and quantities of items have been explored.
- reduce() method used to calculate the cost of individual items has a time complexity of O(N), as it iterates through the price array once.
- Therefore, the overall time complexity of the algorithm can be estimated as **O(N^M)**, as the recursion and the reduce() method are the main contributors to the time complexity

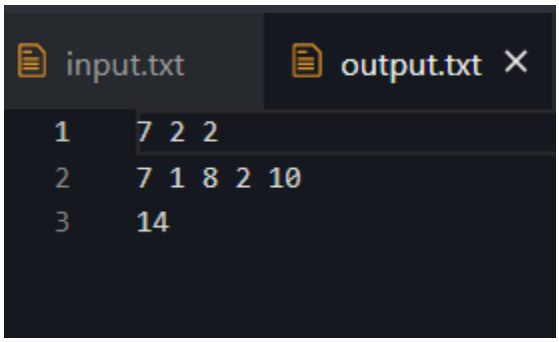
6. Observations

Test Case 1:

Input



Output

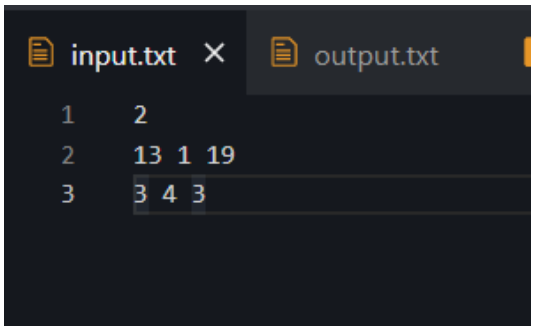


In input.txt the first line is number of items that you are willing to purchase and in the subsequent lines it is item index/no.of items/and price of items.

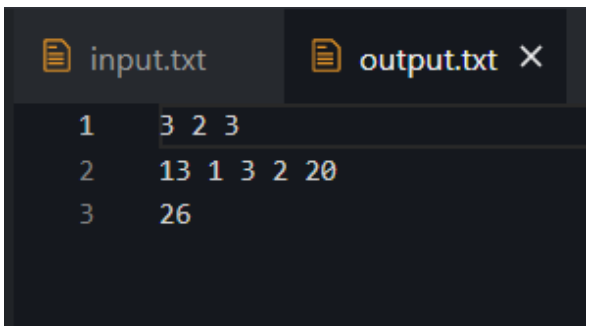
In the output.txt except the last line we are showing the promotions that are applied or the price of the individual items. In the last line we show the optimized/lowest total price possible after considering the promotions for the given item.

Test Case 2:

Input

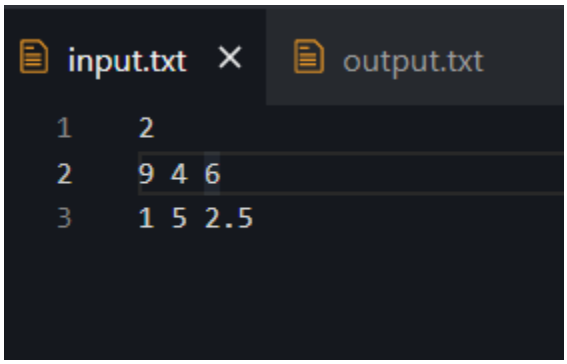


Output

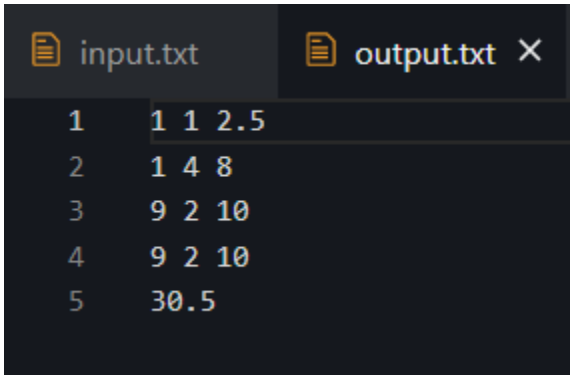


Test Case 3:

Input

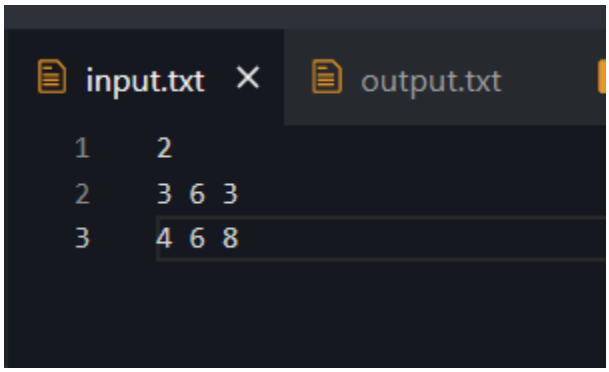


Output

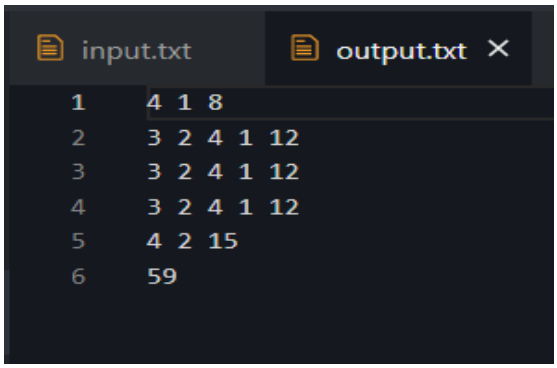


Test Case 4:

Input

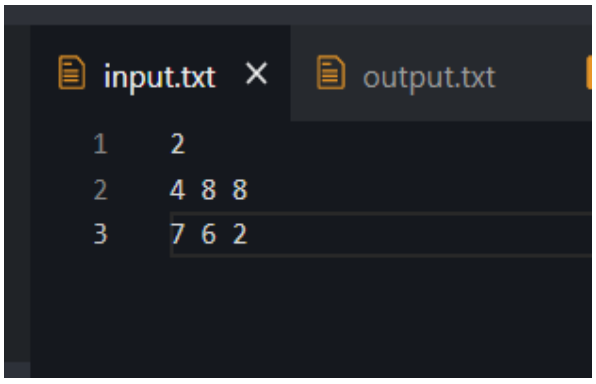


Output

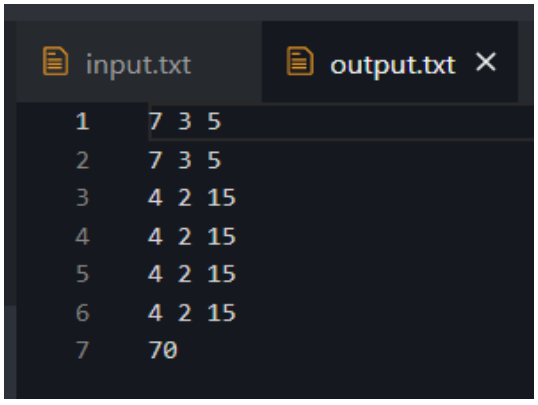


Test Case 5:

Input



Output

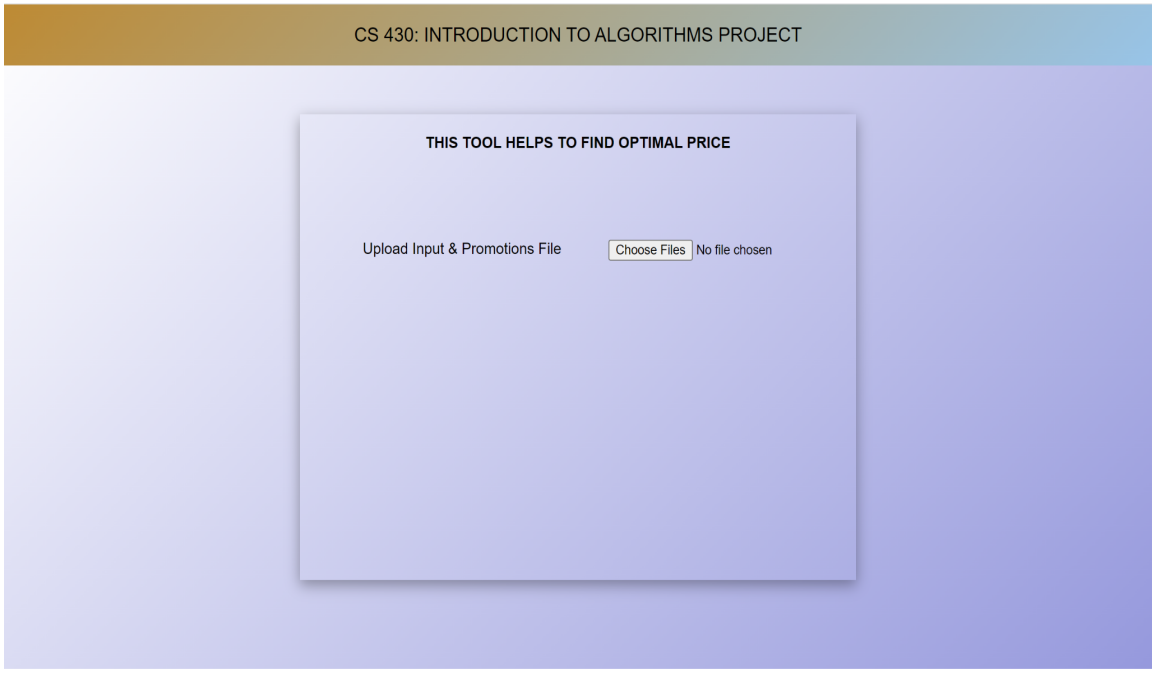


7. GUI

This project uses the below techstack

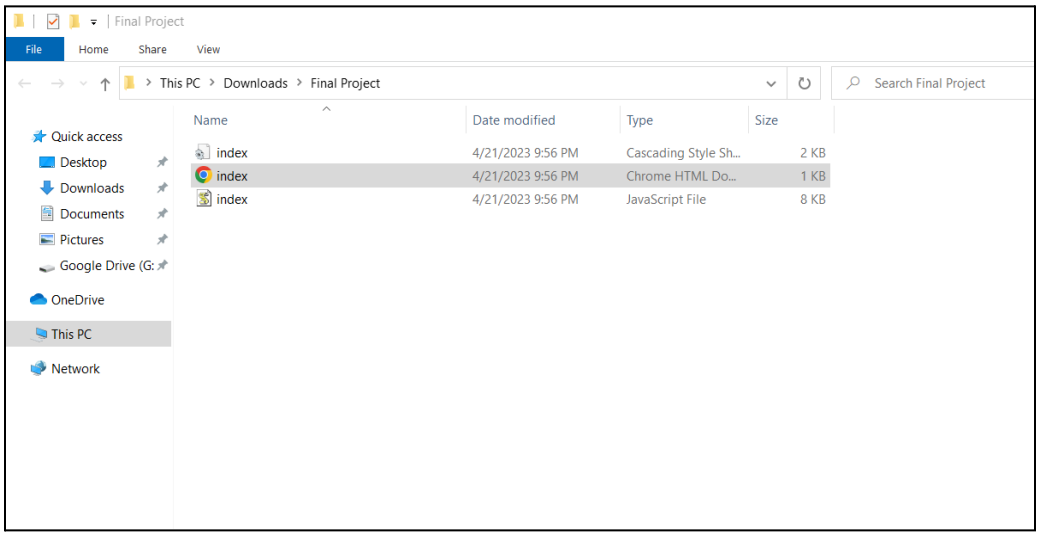
- HTML5
- CSS
- JAVASCRIPT

Below is the Graphical User Interface

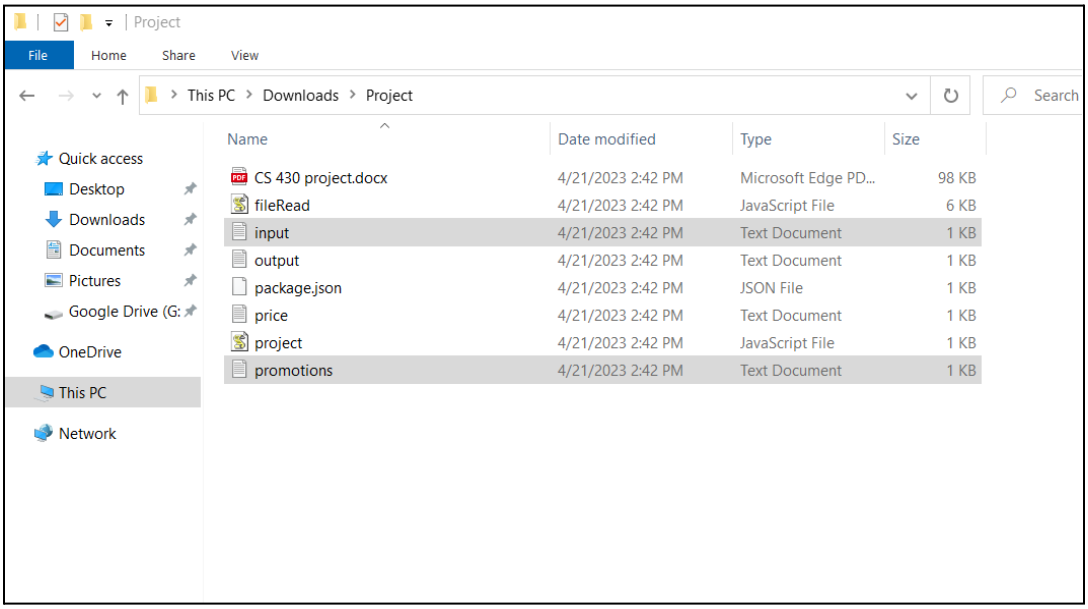


Instructions to use GUI:

Step-1: Select the file “index.html” from “Final project” folder and open it in google chrome or microsoft edge.



Step-2: Upload both “input.txt” and “promotions.txt” by clicking “Choose Files” upload button.
(you can select both “promotions.txt” and “input.txt” from “Project” folder. Change the text in “input.txt” as you wish to upload)



Step-3: As soon as you upload those files, an output is displayed for “Most profitable purchase” and an “output.txt” file is automatically downloaded for reference.