Rubix Bakery Tech Eval
User Guide

Document History

Version	Author	Date	Description
0.0.1	Vinh Phan	15/03/2013	Initial Document

Table of Contents

1. Bakery Algorithm Explanation

Requirements

Given a customer order, it is required to determine the cost and pack breakdown for each product to achieve the minimal number of packs.

Strategy

Applying recursive approach to scan through all feasible options and then select the best option which gives a minimal number of packs.

Explanation

Firstly, set up Data Structure for the algorithm. I use 2 arrays, one to keep the pack types of a product, the other for the number of packs required to contain a given items.
For example,

- We are given "10 VS5". In another word, product code is VS5, and we need to arrange 10 items into packs of VS5
- VS5 has 2 pack types that are 3 and 5
- The main function will create 2 arrays, namely
    - Vs_val = [5, 3]
    - Vs_id = [0, 0]
- ***Notes***: the "vs_val" must be sorted in descending order.

Secondly, execute the "recursive_packs" function to retrieve the minimal packs of the product.
Algorithm input parameters:

- Arr_val: is the "vs_val" in the aforementioned example
- Arr_id: is the "vs_id"
- maxItems: is 10, the items of the customer order
- Cursor: is 0 as default. It is an internal parameter to optimize processing

Algorithm stopping points

1. Checking termination. If the algorithm finds a solution to fill up items into packs, it will return all recursive loops and save the result in each loop.
2. Checking the pack size fitting the given items. If the number of items is less than the pack capacity, it will change to a smaller one until it finds a fit pack.

Feasible outcomes

1. Return "arr_id" which contains the minimal packs to pack the items.
2. The algorithm cannot find a solution to fit all items in packs. It will return an empty list of "arr_id"

3. Compile and Run

The script is developed in Javascript and Python 3. I build an web-based application to visualize the results of 3 sample test cases.

The application is hosted on AWS S3, please find the url to access: http://bakery-test-eval.s3-website-ap-southeast-2.amazonaws.com/

The python script is developed on Google Lab, please find the url to compile and run: https://colab.research.google.com/drive/1vZsIhp8axzfB5DZt4L8zZC78Vgw5GONk

4. Test Cases

I use 3 datasets given in the test statement for testing. The input data and successfully passing tests are given below

#	Product Code	Packs	Quantity	Price / Pack
1	VS5	5	2	$8.99
		3	0	$6.99
Total Cost				$17.98
2	MB11	8	1	$24.95
		5	0	$16.95
		2	3	$9.95
Total Cost				$54.8
3	CF	9	0	$16.99
		5	2	$9.95
		3	1	$5.95
Total Cost				$25.85

5. Source Code

```
/****************************************
        Recursive func to find minimal packs
    Input:
        @arr_val: pack types of the product
        @arr_id:  number of packs (default: 0)
        @maxItems: number of items requires packing
        @cursor:  internal parameter (default: 0)

    Output:
        @arr_id: number of packs to pack items

    ****************************************/
    function recursive_packs(arr_val, arr_id, maxItems, cursor=0){
        let idx = -1, indicator = false, terminated = false;
        if(cursor == arr_id.length){
            return;
        }
        else{
            for(let i=cursor; i < arr_id.length; i++){
                indicator = false;
                if(arr_val[i] <= maxItems){
                    indicator = true;
                }
                if(indicator == true){
                    recursive_packs(arr_val, arr_id, maxItems-arr_val[i], i);
                    idx = i;
                }
                // BEGIN: Check termination
                if(idx != -1){
                    let tmp = 0;
                    for(let x=0; x < arr_id.length; x++){
                        tmp += arr_val[x] * arr_id[x];
                    }
                    if((tmp + arr_val[idx]) == maxItems){
                        terminated = true;
                        break;
                    }
                }
                // END: Check termination
            }
            if(idx != -1 && terminated == true){
                arr_id[idx] += 1;
            }
        }
    }
```