MEMO

CS620 Design Assignment-2

Team 2:

Hai Nguyen (nguyen124@marshall.edu)

Adithya Kiran Mamillapally (mamillapally@marshall.edu)

Thiruveedhula, Venkatesh (thiruveedhul@marshall.edu)

Algorithm explanation:

- **Explain the nature of the algorithm:** the nature of algorithm is dynamic algorithm. Let ***M*** is the total money we have at the beginning, I_i is the i^{th} item in the ***n-ordered*** items list with price P_i and $W_k(M)$ is the number of ways to buy first ***k*** items from ***n-ordered*** items with ***M*** dollar. So with ***M*** dollar, the way we can buy is either buying ***n-1*** items without item I_n or buying item I_n then having $M-P_n$ dollar left to buy ***n-1*** remaining items. We have the following formula:

$$W_n(M) = W_{n-1}(M) + W_{n-1}(M-P_n)$$

- **The factors**: that lead us to this algorithm are
  - The space is limited.
  - The solution can be recursively defined.
- **Pseudocode:**

  String recursiveCount(**totalValueOfItems**, **totalMoney**, **items**, **size**, **count**, **way**, **memorizedWay**)
  - Base case:
    - if(**totalValueOfItems** is equal to **totalMoney**)
      {
      Build a string that represents the **way**
      if(**way** is not equal to **memorizedWay**){
      increase **count** to 1
      return **way**
      }
      }
    - if(**totalMoney** is equal to **0**)
      {
      if(**way** is not equal to **memorizedWay**){
      increase **count** to 1
      return **way**
      }
      }
  - Recursive:
    - if(**size** > 1 and **totalMoney** >0 and **totalMoney** >= smallest item)
      {
      totalValueOfItems = **totalValueOfItems** – last item's value

      memorizedWay = recursiveCount(**totalValueOfItems**, **totalMoney**, **items**, **size**, **count**, **way, memorizedWay**)

      totalMoney = **totalMoney** – last item's value
      way appends bought item's value

memorizedWay = recursiveCount(**totalValueOfItems**, **totalMoney**, **items**, **size, count, way, memorizedWay**)
}

- **Return:** The number of **memorizedWay.** The **count** is the total of ways to buy items.

- **The asymptotic space:** is ***O(b+n)***
  - Size of primitives variables **totalValueOfItems**, **totalMoney, size, way, memorizedWay** is: 5.
  - Size of array **items**: n.
  - Size of array **count:** 1 because it just stores 1 value which is the number of ways.

  In the recursive calls, there will be stack storing copies of primitive variables. Because we know the recursion can't go deep down more than **n** levels, so the memory allocated in stack for primitives variables will be no greater than 6*n. The array and the set will be passed by reference so there is no memory increasing from this variable. So the total memory allocation is 5*n + n + 1 = 6n +1 or we can say **O(b+n)**.

- **The asymptotic running time:**
  Because the program is recursively building a binary tree from **n-items** array like in **Figure 1**; and we have a stop point when the size of array is **1** so in the worst case, the running time will be $\mathbf{O(2^{n-1})}$.

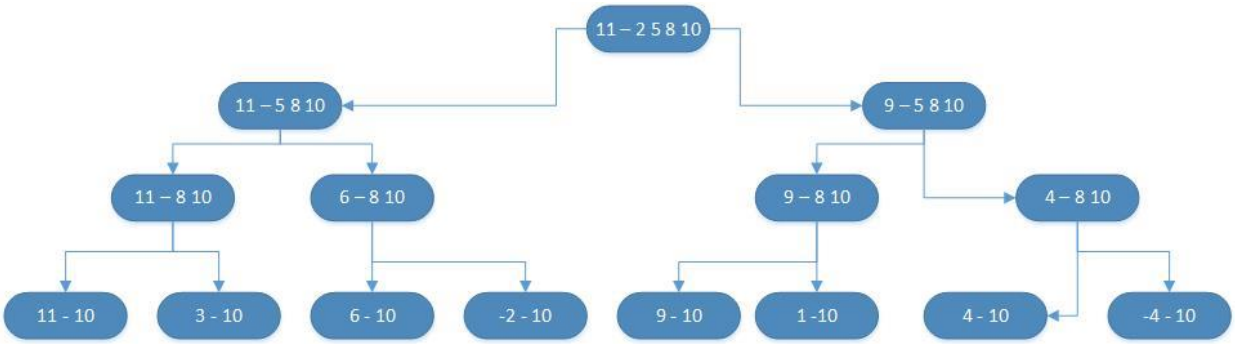

Figure 1: An example for the worst case: buying from 4 items $2, $5, $8 and $10 using $11.

The asymptotic running time can be improved by pruning the tree. The base cases or the conditions for recursion in pseudo code are pruning ways.

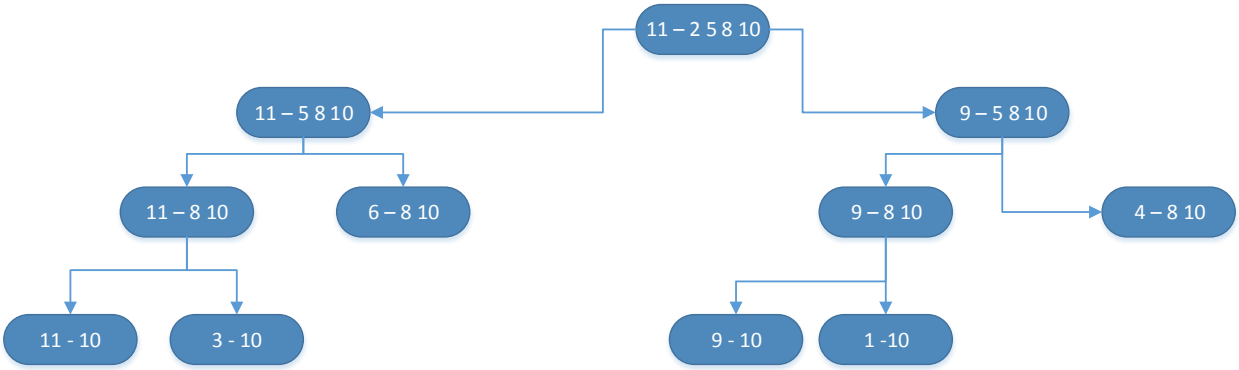

Figure 2: Tree is pruned.

Code summary:

Our program is simple. It contains 2 static functions, **main** function and **recursiveCount** function. The core function is **public static void recursiveCount(int totalValueOfItems, int totalMoney, ArrayList<Integer> items, int size, ArrayList<Integer> count, String way, String memorizedWay)** function which takes 8 input parameters:

- totalValueOfItems: This parameter is used to keep track the current total value of items in the store. The total value of items should be greater than the **totalMoney** we have because each item can only be bought once.
- totalMoney: This parameter is the money we have to buy items. The money is reduced each time we buy an item from the list.
- items: This parameter is the list of items in the store.
- size : The size is a tracking index for the last item of remaining items after buying or ignoring first item from previous items list.
- way: way is a string that contains items' value concatenated together. The program can generate duplicate ways so we have to eliminate it by using **memorizedWay**.
- memorizedWay: this parameter is used to remember previous way and then compare it to the next way. The duplicate way won't be counted.

When **recusiveCount** is called, first it check to see if the **totalValueOfItems** is equal to **totalMoney** or not. If it is so then whatever in the **items** will be appended to the string **way**. The **memorizedWay** is used to memorize the **way**. If the **memorizedWay** is the same as new way then the **count** won't be increased; otherwise, we increase the **count** and memorized the new way**.** If the **totalMoney** is 0 then buying nothing is one way of buying; we also check to memorize new way and increase the count in this case. If the item **size** is greater than 1, **totalMoney** is greater than 0 and the **totalMoney** is bigger than smallest item from the list, then we do these steps:

- reduce **size = size -1**
- reduce the **totalValueOfItems = totalValueOfItems – item's value** at index **size**
- call recursiveCount(**totalValueOfItems**, **totalMoney**, **items**, **size**, **count, way**, **memorizedWay**);
- **remainingMoney** = **totalMoney – item**'s value at index **size**

- append item value to string **way**: **way** = **way** + **item** + " "
- call recursiveCount(**totalValueOfItems**, **remainingMoney**, **items**, **rightSize, leftSize**, **count, way**, **memorizedWay**);

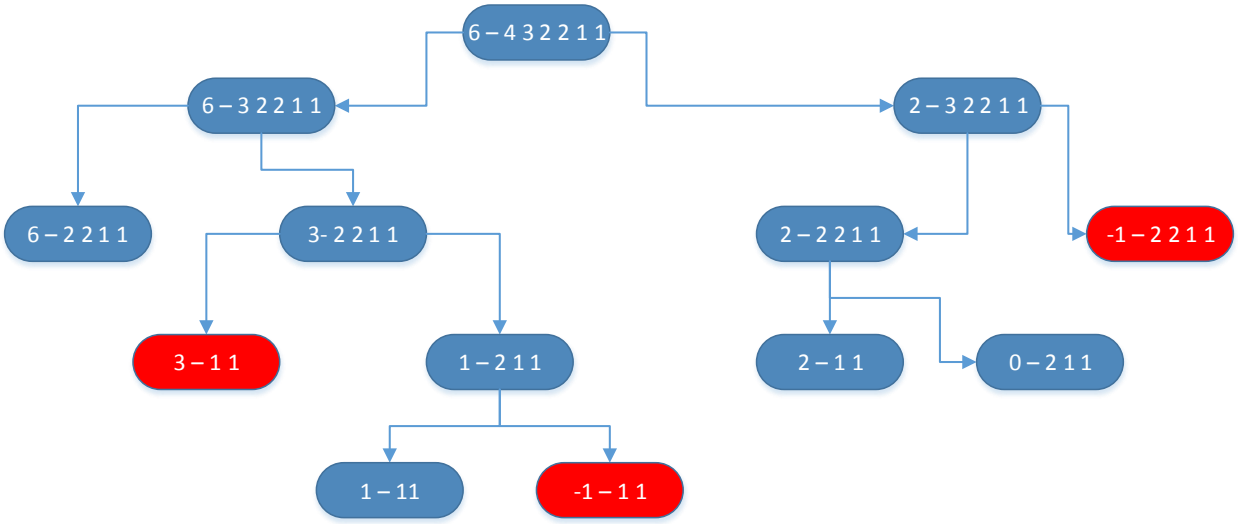

Figure 3: The tree represents possible ways to buy from 6 items ($4,$3,$2,$2,$1,$1) with $6. The red nodes are stopped nodes.

Team Summary:

Roles of the team members:

- Hai Nguyen and Venkatesh Thiruveedhula worked on the code.
- Adithya & Hai Nguyen fixed errors and improved the program efficiency.
- All team members worked in writing the Memo.

Meeting-1:

Time: On 30/03/2016fromt 3:30pm
Place: Applied Engineering Complex
Duration: 2 hours
Topics Discussed:
1. Understanding and analyzing the requirements of assignment
2. Algorithm that is used in program
3. Assigning work

Meeting-2:

Time: On 03/04/2016 from 7:30 pm
Place: Drinko Library
Duration: 4 hours
Topics Discussed:
1. Coded & tested program

2. Discussed the code summary
3. Discussed the preparation of memo
4. Made changes in the memo
5. Finalized the document

Meeting-3:

Time: On 10/04/2016 from 10:00 am
Place: Online meeting
Duration: 8 hours
Topics Discussed:
1. Modified program
2. Revised document
3. Finalized program
5. Finalized the document