ALGORITMS AND PROBLEM SOLVING LAB

15B11CI411

PROJECT – STORE STOCK PURCHASE OPTIMISING PROGRAM

Submitted To: Dr. Neeraj Jain

Dr. Raju Pal

Submitted By: Abhishek Bhati Enrl. No. 9917103139

Shivam Goel Enrl. No. 9917103158

Batch: F5

Department of CSE/IT

Jaypee Institute of Information Technology, Sector 128, Noida

MAY 2019

PROJECT DESCRIPTION

The project entitled “Store Stock Purchase Optimizing Program” helps the owner of a small-scale local store to make efficient purchase decisions for the store stock. Due to the limited capacity of a store, the owner can keep only a finite number of objects for sale. He has to maximize the profit of the store by selling only these number of objects. With the help of the program, the owner can give inputs according to the requirements of the store and can get a combination of products that will give maximum profits. The main data structure used is a structure defined as item that contains all the details of an item that is sold or purchased with the name item. The structure item contains and stores the following fields:

1. String name: The name of the item
2. Int Index: used when sorting the items by the value ratio
3. Int Id: used for the identification of the item
4. Weight: If the item is purchased in terms of weights rather than units, then this will be used. Else, it will be put to zero.
5. Unit: If the item is purchased in terms of units rather than weights. Then this will be used. Else, it will be put to zero.
6. Float purchase_price: The price at which the item is purchased.
7. Float selling_price: The price at which the item is sold.
8. Int month: The month in which the past sales record was made.
9. Int year: The year in which the past sales record was made.
10. Float ValueRatio: The ratio of the profit earned to the amount of the item sold.
11. Average: The average number of items sold during a month. Calculated using all the past records entered.
12. Int counter: To keep a record of the number of records entered for the particular item during all the past entries.
13. Int tunits: The total number of units of the item sold during all the past records, used to calculate the average number of the units for that particular item having a unique ID.
14. Int tweight: The total weight of the items sold during all the past records, used to calculate the average weight sold for that particular item having a unique ID.

Along with this, some other global variables also have been declared as follows:

1. Int n: This takes the number of the items for whose the previous records are to be entered in the program.
2. Total_weights: This takes in the total amount that can be stored at maximum in the store in the form of weights.
3. Total_units: This takes in the total amount that can be stored at maximum in the store in the form of units.
4. Units: This variable is initialized to zero and is increment according to the conditions until it becomes equal to total_units.

5. Weights: This variable is also initialized to zero and is incremented according to the conditions until it becomes equal to total_weights.
6. vector<item> itm: This is a variable of structures of the type item that is used to store all the past items and the sales records.

Apart from these, three functions have also been declared globally;

1. int checkinteger(int num): It checks whether the number entered by the user is less than zero or not. If it is less than zero, then the number entered is required to be entered again and again using recursion until the user enters the correct number. It is used to check fields like the item id, the number of units, amount sold, etc.
2. float checkfloat(float decimal): It checks whether the decimal value entered by the user is less than zero or not. If it is less than zero, then the value is required to be entered again and again using recursion until the user enters the correct value. It is used to check fields like the selling price, the purchasing price, etc.
3. void add_records: In this function the user is required to enter the past records and then these records are stored in the vector named itm. The past records are used to calculate the average amount of items of a particular type sold in a month. The idea behind this is that a store will sell only a certain quantity of a item in a month in the area that it operates.

The rest of the operations that are used in the program are defined as follows in the main function :

At first, the function to all the past records is called and these are stored in the vector itm.

Then, using bubble sort, the vector is sorted on the basis of the ValueRatio in decreasing order.

After this, the Knapsack algorithm is applied to calculate the order and the number of items to be purchased on the basis of the maximum profit.

Two flag variables flag1 and flag2 are used to keep a count of the items and check when the purchased amount has reached the capacity. The items are first decremented on the basis of the average amount sold and if the capacity of the store falls short, then only the leftover amount is added to the list of the items to be purchased. All these details are further stored in the vector objects which is composed of structure variables of the type item.

At last, the dotted lines represent the amount contributed in the total profit by a particular item in the constructed list, provided that all the items are sold. It is calculated on the basis of the percentage and then the dots are represented on the basis of the length of the fraction calculated multiplied by 100.

The worst-case time complexity of the overall program will be $O(n^2)$ and the best-case time complexity of the overall program will be O(n).

Program Code:

```
/*This project has been prepared by Abhishek Bhati (Enrollment No. 9917103139 Batch:F5) and

Shivam Goel (Enrollment No. 9917103158 Batch:F5. It uses fractional knapsack problem algorithm and optimizes the purchase of the

items by taking the amount of values that were sold in the past and taking their average.*/


#include<bits/stdc++.h>

#include<algorithm>

#include<iomanip>


using namespace std;


struct item

{

    string name;

    int index;

    int id;

    int weight;

    int unit;

    float purchase_price;

    float selling_price;

    float profit;

    float loss;

    int amount_sold;

    int month;

    int year;

    float ValueRatio;

    int average;

    int counter;

    int tunits;

    int tweight;

};


int n;

int total_weights;
```

```
int total_units;
int units;
int weights;
vector<item> itm;

int checkinteger(int num)
{
    int d;
    if(num<0)
    {
        cout<<"\nWrong value entered. Enter again.";
        cin>>d;
        checkinteger(d);
    }
    else
    {
        return num;
    }
}

float checkfloat(float decimal)
{
    int d;
    if(decimal<0.0)
    {
        cout<<"\nWrong value entered. Enter again.";
        cin>>d;
        checkfloat(d);
    }
    else
    {
        return decimal;
    }
}

void add_records()
{
    cout<<"\nEnter the number of items that you want to enter:\n";
```

```
    cin>>n;
    int i;
    struct item obj;
    for(i=0; i<n; i++)
    {
        cout<<"\nEnter the month and the year of the sale:\n";
        cin>>obj.month>>obj.year;
        cout<<"\nEnter the name of the item:\n";
        cin>>obj.name;
        cout<<"\nEnter the id of the item:\n";
        cin>>obj.id;
        obj.id=checkinteger(obj.id);
        cout<<"\nEnter the weight (If the item is purchased in terms of the
units then enter 0 for weight)\n";
        cin>>obj.weight;
        obj.weight=checkinteger(obj.weight);
        cout<<"\nEnter the units (If the item is purchased in terms of the
weights then enter 0 for unit)\n";
        cin>>obj.unit;
        obj.unit=checkinteger(obj.unit);
        cout<<"\nEnter the purchasing price of the item per unit or unit
weight:\n";
        cin>>obj.purchase_price;
        obj.purchase_price=checkfloat(obj.purchase_price);
        cout<<"\nEnter the selling price of the item per unit or unit
weight:\n";
        cin>>obj.selling_price;
        obj.selling_price=checkfloat(obj.selling_price);
        cout<<"\nEnter the amount of the item sold during this month:\n";
        cin>>obj.amount_sold;
        if(obj.unit==0)
        {
            if(obj.amount_sold<obj.weight)
            {
                obj.loss=(obj.purchase_price*obj.weight)-
(obj.selling_price*obj.amount_sold);
                obj.profit=0.0;
            }
            else
            {
```

```
                obj.profit=(obj.weight)*(obj.selling_price-obj.purchase_price);
                obj.loss=0.0;
            }
        }
        if(obj.weight==0)
        {
            if(obj.amount_sold<obj.unit)
            {
                obj.loss=(obj.purchase_price*obj.unit)-
(obj.selling_price*obj.amount_sold);
                obj.profit=0.0;
            }
            else
            {
                obj.profit=(obj.unit)*(obj.selling_price-obj.purchase_price);
                obj.loss=0.0;
            }
        }
        obj.ValueRatio=obj.profit/amount_sold;
        obj.index=0;
        obj.counter=0;
        obj.average=0;
        obj.tunits=0;
        obj.weight=0;
        itm.push_back(obj);
    }

    cout<<"\nThese are the past records that are present in the records\n";
    struct item temp;
    for(i=0; i<n; i++)
    {
        temp=itm[i];
        cout<<"\nItem Name: "<<temp.name;
        cout<<"\nItem Id: "<<temp.id;
        cout<<"\nItem Weight: "<<temp.weight;
        cout<<"\nItem Unit: "<<temp.unit;
        cout<<"\nItem Purchase Month: "<<temp.month;
        cout<<"\nItem Purchase Year: "<<temp.year;
        cout<<"\nItem amount sold "<<temp.amount_sold;
```

```
        cout<<"\nItem Purchase Price: "<<temp.purchase_price;
        cout<<"\nItem Selling Price: "<<temp.selling_price;
        cout<<"\nItem Profit: "<<temp.profit;
        cout<<"\nItem Loss: "<<temp.loss;
    }
}

int main()
{
    cout<<"\nEnter the total weights and the units that your store can hold:\n";
    cin>>total_weights;
    cin>>total_units;
    char ch;
    cout<<"\nEnter the previous records for the data file";
    add_records();
    int j;
    int k;
    units=0;
    weights=0;
    struct item tmp;
    for(j=0; j<itm.size(); j++)
    {
        for(k=0; k<itm.size()-j-1; k++)
        {
            if(itm[k].ValueRatio<itm[k+1].ValueRatio)
            {
                tmp=itm[k];
                itm[k]=itm[k+1];
                itm[k+1]=itm[k];
            }
        }
    }
    cout<<"value Ratio";
    for(j=0; j<itm.size(); j++)
    {
        itm[j].index=j;
        cout<<itm[j].ValueRatio<<"\t";
    }
```

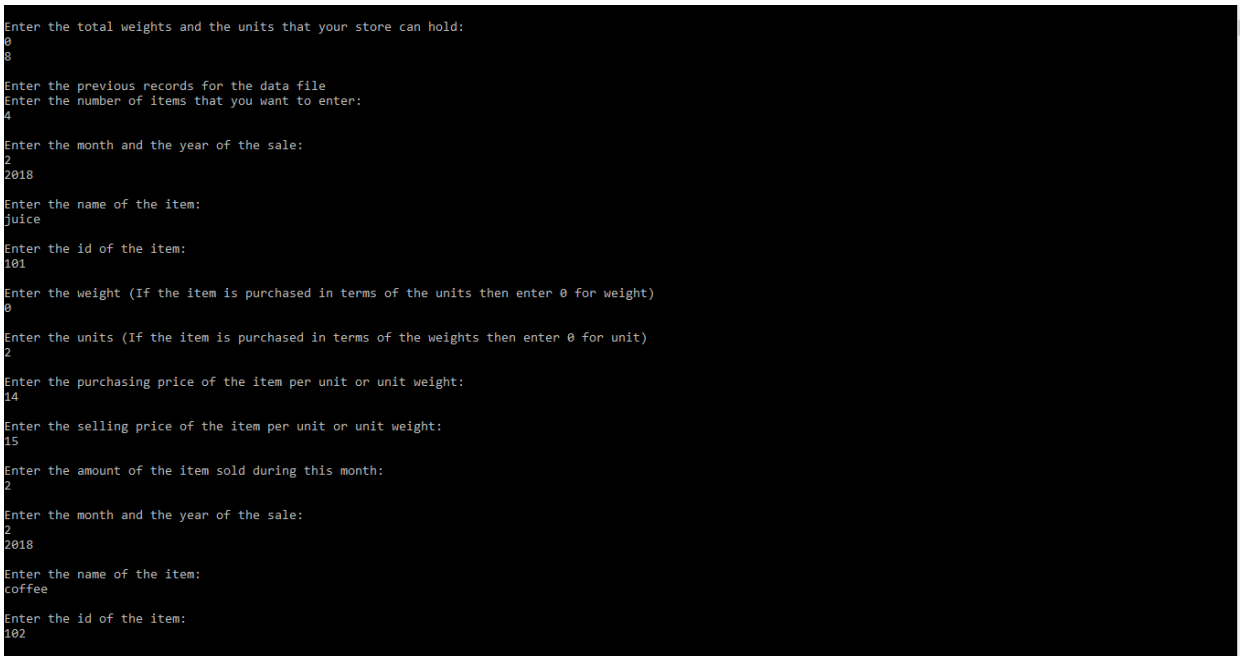
```
for(j=0; j<itm.size(); j++)
{
    tmp=itm[j];
    for(k=j; k<itm.size(); k++)
    {
        if(itm[j].id==itm[k].id)
        {
            itm[j].counter=itm[j].counter+1;
            if(itm[j].weight==0)
            {
                itm[j].tunits+=itm[k].unit;
                itm[j].average=itm[j].tunits/itm[j].counter;
            }
            else if(itm[j].unit==0)
            {
                itm[j].tweight+=itm[k].weight;
                itm[j].average=itm[j].tweight/itm[j].counter;
            }
        }
    }
    cout<<"\nItem Average:"<<itm[j].average;
}
vector <item> objects;
int flag1=0;
int flag2=0;
while(flag1==0 && flag2==0)
{
    for(j=0; j<itm.size(); j++)
    {
        if(itm[j].weight==0)
        {
            if(itm[j].unit<=total_units-units)
            {
                tmp=itm[j];
                tmp.unit=itm[j].average;
                units+=tmp.unit;
                if(units==total_units)
                {
```

```
                    flag1=1;
                }
                tmp.profit=(itm[j].selling_price-
itm[j].purchase_price)*tmp.unit;
                objects.push_back(tmp);
            }
            else
            {
                tmp=itm[j];
                tmp.unit=total_units-units;
                units+=tmp.unit;
                if(units==total_units)
                {
                    flag1=1;
                }
                tmp.profit=(itm[j].selling_price-
itm[j].purchase_price)*tmp.unit;
                objects.push_back(tmp);
            }
            if(weights==total_weights)
            {
                flag2=1;
            }
        }
        else if(itm[j].unit==0)
        {
            if(itm[j].weight<total_weights-weights)
            {
                tmp=itm[j];
                tmp.weight=itm[j].average;
                weights+=tmp.weight;
                if(weights==total_weights)
                {
                    flag2=1;
                }
                tmp.profit=(itm[j].selling_price-
itm[j].purchase_price)*tmp.weight;
                objects.push_back(tmp);
            }
            else
```

```
                {
                    tmp=itm[j];
                    tmp.weight=total_weights-weights;
                    weights+=tmp.weight;
                    if(weights==total_weights)
                    {
                        flag2=1;
                    }
                    tmp.profit=(itm[j].selling_price-
itm[j].purchase_price)*tmp.weight;
                    objects.push_back(tmp);
                }
                if(units==total_units)
                {
                    flag1=1;
                }
            }
        }
    }
    int ctr;
    int total_profit=0.0;
    for(ctr=0; ctr<objects.size(); ctr++)
    {
        tmp=objects[ctr];
        total_profit+=tmp.profit;
        cout<<"\nItem Name: "<<tmp.name;
        cout<<"\nItem Id: "<<tmp.id;
        cout<<"\nItem Weight: "<<tmp.weight;
        cout<<"\nItem Unit: "<<tmp.unit;
        cout<<"\nItem Purchase Month: "<<tmp.month;
        cout<<"\nItem Purchase Year: "<<tmp.year;
        cout<<"\nItem Purchase Price: "<<tmp.purchase_price;
        cout<<"\nItem Selling Price: "<<tmp.selling_price;
        cout<<"\nItem Profit: "<<tmp.profit;
        cout<<"\nItem Loss: "<<tmp.loss;
    }
    cout<<"\nThe total profit expected from such a sale is "<<total_profit;
    float fraction;
    int len;
```

```
    int c;
    for(ctr=0; ctr<objects.size(); ctr++)
    {
        cout<<"\n";
        cout<<"Item ID: "<<objects[ctr].id;
        fraction=objects[ctr].profit/total_profit;
        len=(int)(fraction*100);
        for(c=0; c<len; c++)
        {
            cout<<".";
        }
    }
    return 0;
}
```

Program Output:



```
102

Enter the weight (If the item is purchased in terms of the units then enter 0 for weight)
0

Enter the units (If the item is purchased in terms of the weights then enter 0 for unit)
1

Enter the purchasing price of the item per unit or unit weight:
19

Enter the selling price of the item per unit or unit weight:
20

Enter the amount of the item sold during this month:
1

Enter the month and the year of the sale:
3
2018

Enter the name of the item:
coffee

Enter the id of the item:
102

Enter the weight (If the item is purchased in terms of the units then enter 0 for weight)
0

Enter the units (If the item is purchased in terms of the weights then enter 0 for unit)
2

Enter the purchasing price of the item per unit or unit weight:
19

Enter the selling price of the item per unit or unit weight:
23

Enter the amount of the item sold during this month:
2

Enter the month and the year of the sale:
2
```

```
Enter the name of the item:
tea

Enter the id of the item:
103

Enter the weight (If the item is purchased in terms of the units then enter 0 for weight)
0

Enter the units (If the item is purchased in terms of the weights then enter 0 for unit)
4

Enter the purchasing price of the item per unit or unit weight:
19

Enter the selling price of the item per unit or unit weight:
20

Enter the amount of the item sold during this month:
4

These are the past records that are present in the records

Item Name: juice
Item Id: 101
Item Weight: 0
Item Unit: 2
Item Purchase Month: 2
Item Purchase Year: 2018
Item amount sold 2
Item Purchase Price: 14
Item Selling Price: 15
Item Profit: 2
Item Loss: 0
Item Name: coffee
Item Id: 102
Item Weight: 0
Item Unit: 1
Item Purchase Month: 2
Item Purchase Year: 2018
Item amount sold 1
Item Purchase Price: 19
Item Selling Price: 20
```

```
Item Purchase Price: 19
Item Selling Price: 20
Item Profit: 1
Item Loss: 0
Item Name: coffee
Item Id: 102
Item Weight: 0
Item Unit: 2
Item Purchase Month: 3
Item Purchase Year: 2018
Item amount sold 2
Item Purchase Price: 19
Item Selling Price: 23
Item Profit: 8
Item Loss: 0
Item Name: tea
Item Id: 103
Item Weight: 0
Item Unit: 4
Item Purchase Month: 2
Item Purchase Year: 2018
Item amount sold 4
Item Purchase Price: 19
Item Selling Price: 20
Item Profit: 4
Item Loss: 0
 The value ratio of the items are as follows:4  4       4       1

The list of items to be purchased is as follows:


Item Name: coffee
Item Id: 102
Item Weight: 0
Item Unit: 2
Item Purchase Month: 3
Item Purchase Year: 2018
Item Purchase Price: 19
Item Selling Price: 23
Item Profit: 8
Item Loss: 0

Item Name: coffee
Item Id: 102
```

```
Item Name: coffee
Item Id: 102
Item Weight: 0
Item Unit: 2
Item Purchase Month: 3
Item Purchase Year: 2018
Item Purchase Price: 19
Item Selling Price: 23
Item Profit: 8
Item Loss: 0

Item Name: coffee
Item Id: 102
Item Weight: 0
Item Unit: 2
Item Purchase Month: 3
Item Purchase Year: 2018
Item Purchase Price: 19
Item Selling Price: 23
Item Profit: 8
Item Loss: 0

Item Name: tea
Item Id: 103
Item Weight: 0
Item Unit: 2
Item Purchase Month: 2
Item Purchase Year: 2018
Item Purchase Price: 19
Item Selling Price: 20
Item Profit: 2
Item Loss: 0
The total profit expected from such a sale is 26
Item ID: 102.............................
Item ID: 102.............................
Item ID: 102.............................
Item ID: 103.......
Process returned 0 (0x0)   execution time : 100.609 s
Press any key to continue.
```