컴퓨터 알고리즘과 실습 1주차 보고서

2016112158 김희수

문제1번

<1-1> 유사코드 작성

accumulate\_average(array):

    sum = 0

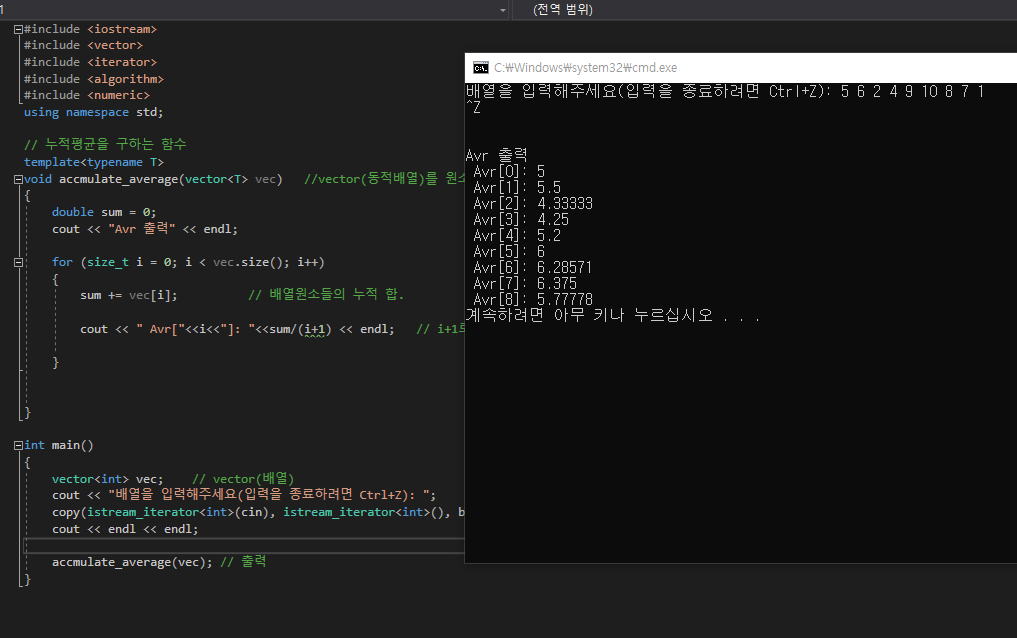
    for i = 1 to Length(array):

        sum+=array[i]

        return sum/(i+1)

<1-2> 소스코드 첨부했음

<1-3실행화면>



문제2번

<2-1 GreedyBetterChange 유사코드>

GreedBetterChange(Money, coin, d):

    for i := d to 1

        k[i] := M/c[i] //k is array. k1, k2, ... , kd

        M := M - c[i] \* k[i]

    return k

<2-2>

<2-3 BruteForceChange 유사코드>

BruteForceChange(coinValueList, total, numCoins, coinDict):

    for i := 0 to length(coinValueList):

        coinDict[coinValueList[i]]

    if total == 0:

        return (numCoins, coinDict)

    bestCoins := -1

    bestDict := make\_dictionary()   // make empty dictionary

    for i := 0 to length(coinValueList):

        dictCopy := make\_dictionary()

        for i:=0 to length(coinValueList):

            dictCopy[coin] = coinDict[coin]

        coin := conValueList[i]

        if coin <= total:

            dictCopy[coin] += 1

            (subCoins, subDict) := BruteForceChange(coinValueList, totla-coin, numCoins+1, dictCopy)

            if bestCoins == -1 && subCoins < bestCoins:

                bestCoins := subCoins

                bestDict := subDict

    return (bestCoins, bestDict)

<2-2, 2-4 소스코드와 실행화면>

#include <iostream>

#include <vector>

#include <algorithm>

#include <iterator>

#include <iomanip>

#include <map>

#include <tuple>

using namespace std;

// define GreedyBetterChange

// M : total Money

// c : coinList

// d : length of coinList

void GreedBetterChange(int M, vector<int> c, int d)

{

    vector<int> k(d, 0);    // k1,k2,...kd

    sort(c.begin(), c.end());   // 혹시 coinList가 정렬되지 않았다면 먼저 정렬한다

    //알고리즘 연산

    for (int i = d - 1; i >= 0; i--)

    {

        k[i] = M / c[i];        // 큰 코인부터 사용

        M = M - c[i] \* k[i];    // 기존의 money에서 사용된 코인만큼의 돈을 공제

    }

    // 출력

    for (int i = 0; i < d; i++)

    {

        cout << c[i] << ": " << k[i] << endl;

    }

}

// dicionary초기화를 위한 함수

void build\_Dict(vector<int> coinValueList, map<int, int> coinDict)

{

    for (auto& coin : coinValueList)

        coinDict[coin] = 0;

}

// define BruteForceChange

// coinValueList : coinlist. ex) (25 20 10 5 1)

// numCoins : 코인개수

// coinDict : 코인종류(딕셔너리)

tuple<int, map<int, int>> BruteForceChange(vector<int> coinValueList, int total, int numCoins, map<int, int> coinDict, bool build)

{

    sort(coinValueList.begin(), coinValueList.end());       // 혹시 coinValueList가 정렬되지 않았다면 먼저 정렬

    if (build == true)

        build\_Dict(coinValueList, coinDict);        // 딕셔너리 초기화

    if (total == 0)         // total money가 0이 되면

        return tuple<int, map<int, int>>(numCoins, coinDict);   // 사용된 코인 개수와 코인종류를 반환

    // 찾고자하는 코인 개수와 코인종류 초기화

    int bestCoins = -1;

    map<int, int> bestDict;

    // coinValueList만큼 반복

    for (size\_t i = 0; i < coinValueList.size(); i++)

    {

        map<int, int> dictCopy;         //임시 딕셔너리

        for (auto& coin : coinValueList)

            dictCopy[coin] = coinDict[coin];

        int coin = coinValueList[i];

        if (coin <= total)

        {

            dictCopy[coin] += 1;        // 해당코인에 해당하는 딕셔너리의 value를 1증가. 예를 들어 dictCopy[25]가 1증가 됬다 >> 25를 key로 가지는 value가 1증가 됬다는 뜻.

            tuple<int, map<int, int>> temp\_tuple = BruteForceChange(coinValueList, total - coin, numCoins + 1, dictCopy, false);    // recursion

            int subCoins = get<0>(temp\_tuple);

            map<int, int> subDict = get<1>(temp\_tuple);

            if (bestCoins == -1 || subCoins < bestCoins)

            {

                bestCoins = subCoins;

                bestDict = subDict;

            }

        }

    }

    tuple<int, map<int, int>> result\_tuple = make\_tuple(bestCoins, bestDict);

    return result\_tuple;

}

int main()

{

    // 금액 입력

    int total;

    cout << "총 금액을 입력하시오: ";

    cin >> total;

    getchar();

    // 코인 종류 입력

    vector<int> coinValueList;

    cout << "코인의 종류를 입력하시오(입력을 종료하려면 Ctrl+Z): ";

    copy(istream\_iterator<int>(cin), istream\_iterator<int>(), back\_inserter(coinValueList));

    int d = coinValueList.size();

    map<int, int> coinDict;

    //Greed Better Change 출력

    cout << "Greed Better Change: " << endl;

    GreedBetterChange(total, coinValueList, d);

    cout << "=========================" << endl;

    // Brute Force Change 출력

    tuple<int, map<int, int>> bfc = BruteForceChange(coinValueList, total, 0, coinDict, true);

    int result\_coins = get<0>(bfc);

    map<int, int> result\_dict = get<1>(bfc);

    cout << "Brute Force Change: " << endl;

    for (auto iter = result\_dict.begin(); iter != result\_dict.end(); iter++)

    {

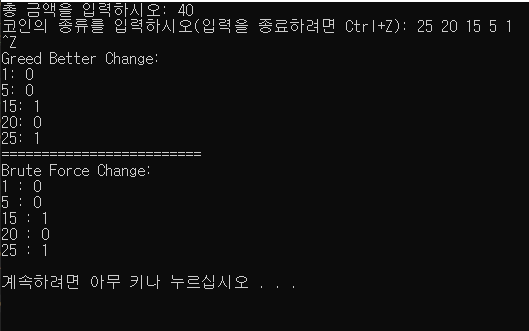
        cout << iter->first << " : " << iter->second << endl;

    }

    cout << endl;

    return 0;

}



<2-5 GreedyBetter과 BruteForce가 값이 다른경우>

