**EECE 7205 Fundamentals of Computer Engineering**

**Report of Project1**

Liangshe Li

**Problem Description:**

You are given an input array 𝐴[1, … , 𝑁]. A grouping of the array 𝐴 is described by an array 𝐺[1, … , 𝑀], where the array 𝐴 is partitioned into 𝑀 groups, the 1st group consists of the first 𝐺[1] elements of array 𝐴, the 2nd group consists of the next 𝐺[2] elements, and so forth. Define array 𝐵[1, … , 𝑀] such that 𝐵[𝑗] is the summation of the elements in the 𝑗-th group of array 𝐴. Use a dynamic programming algorithm to find a grouping of array 𝐴 with 𝑀 groups such that we maximize the minimum element of array 𝐵.

**Max-min-grouping(𝐴, 𝑁, 𝑀)**

{

return 𝐺[1, … , 𝑀]

}

**Part1 Pseudo Codes**

First, I will show the pseudo codes as below:

**Max\_min\_grouping (A, G, m, n)** // A is input. G is an empty array and G will be output, m is array A’s size and n is the number of groups //

**if**

c[n,m], p[n,m] to be new tables

**for**

**for**

**for**

**for**

**else if**

**else**

print ”The factors input are not correct! ”

**Part2 Analysis of Running Time**

This part I will calculate the running time of this algorithm. Note that m is the size of array A needed to be grouped, and n is the number of groups.

First, the algorithm will use “if” sentence to find the proper case. The running time of this is . The algorithm will consider three cases. I find that the latter two cases are not common cases and their running time is not large, so next step I will mainly consider the first case: if .

Then, I find that the first “for” loop in the first case needs to sum up all the numbers in the array A and it will repeat m times, so in this operation the running time is .

Next, there are two “for” loops between which one (from 1 to m-1) is nested in another (from 1 to n-1). And the operation needs to find the maximum number from an array whose range is from j-1 to i. And the worst case we need to sum up the array’s numbers whose range are from 0 to m-1(can be nearly the total array). So this part the running time is .

At last, the “for” loop is from n-2 to 0, so the running time is .

Totally the sum of four sections’ running time can be . If n is smaller enough than m, the running time can be . But n is a randomized number, so ultimately the running time of the algorithm is .

**Part3 Results**

This part I will show how to use six examples below to test my codes. These examples contain almost every kind of case, which is illustrated below:

**Example1:**

**Input:** A1= {3,9,7,8,2,6,5,10,1,7,6,4}

M1=3

**Output:** G1= {3,4,5}

**Note:** The input is the same with the project request.

**Example2:**

**Input:** A2= {5,4,15,13,2,4,20,9,2,4}

M2=10

**Output:** G2= {1,1,1,1,1,1,1,1,1,1}

**Note:** The number of groups is the equal to the size of array A2.

**Example3:**

**Input:** A3= {5,19,5,71,20,13}

M3=1

**Output:** G3= {6}

**Note:** The number of groups is 1.

**Example4:**

**Input:** A4= {3,8,50,1,3,44,12,5,9,32,9,7}

M4=6

**Output:** G4= {3,3,1,2,1,2}

**Example5:**

**Input:** A5= {5,7,3,4,8,1,20,4,5,13,8,10,6,9,12,3,6,14,11,5}

M5=5

**Output:** G5= {6,3,3,4,4}

**Example6:**

**Input:** A6= {2,5,4,8}

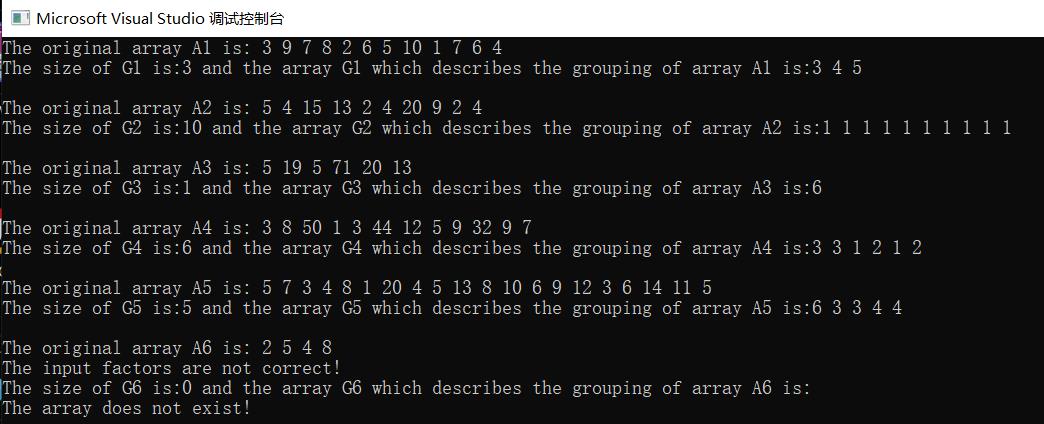
M6=0

**Output:** The input factors are not correct!

The array does not exist!

**Note:** The number of groups is 0 (It is not a correct factor).

And my result of codes is as following:



**Part4 Source codes**

This part, I will list the source codes as below:

#include "Matrix.h"

#include<iostream>

using namespace Numeric\_lib;

using namespace std;

// the fuction of summing up an array which begins with A[p] and ends with A[q]

int sumup(Matrix<int,1>&A, int p, int q)

{

if (q >= p)

{

int i;

int sum;

sum = 0;

for (i = p; i <= q; i++)

sum = sum + A(i);

return sum;

}

}

// find the minimum in two numbers

int min\_in2(int p, int q)

{

if (p <= q)

return p;

else

return q;

}

// find the serial number of the maximum number in an array

int max\_inarray(Matrix<int,1>&A, int p, int q)

{

if (q > p)

{

int i;

int max;

max = p;

for (i = p; i <= q; i++)

if (A(i) > A(max))

max = i;

return max;

}

else if (p == q)

return p;

}

void max\_min\_grouping(Matrix<int,1>&A, Matrix<int,1>&G, int m, int n)

{

if ((m >= n) && (n > 1))

{

Matrix<int, 2>c(n, m);

Matrix<int, 2>p(n, m);

Matrix<int, 1>b(m);

int i, j, k;

for (i = 0; i < m; i++)

{

c(0, i) = sumup(A, 0, i);

p(0, i) = 0;

}

for (j = 1; j < n; j++)

{

for (i = j; i < m; i++)

{

for (k = 0; k < i; k++)

b(k) = min\_in2(c(j - 1, k), sumup(A, k + 1, i));

c(j, i) = b(max\_inarray(b, 0, i - 1));

p(j, i) = 1 + max\_inarray(b, 0, i - 1);

}

}

G(n - 1) = m - p(n - 1, m - 1);

for (j = n - 2; j >= 0; j--)

G(j) = m - sumup(G, j + 1, n - 1) - p(j, m - 1 - sumup(G, j + 1, n - 1));

}

else if (n == 1)

G[0] = m;

else

{

cerr << "The input factors are not correct!";

cout << endl;

}

}

// print all numbers of an array

void print\_all\_array1(Matrix<int, 1>&A)

{

if (A.size() > 0)

{

int i;

for (i = 0; i < A.size(); i++)

cout << A(i) << " ";

cout << endl;

}

else

{

cout << endl;

cerr << "The array does not exist!";

cout << endl;

}

}

int main()

{

int C1[12] = { 3,9,7,8,2,6,5,10,1,7,6,4 };

int C2[10] = { 5,4,15,13,2,4,20,9,2,4 };

int C3[6] = { 5,19,5,71,20,13 };

int C4[12] = { 3,8,50,1,3,44,12,5,9,32,9,7 };

int C5[20] = { 5,7,3,4,8,1,20,4,5,13,8,10,6,9,12,3,6,14,11,5 };

int C6[4] = { 2,5,4,8 };

int M1 = 3, M2 = 10, M3 = 1, M4 = 6, M5 = 5, M6 = 0;

Matrix<int, 1>A1=C1;

Matrix<int, 1>G1(M1);

cout << "The original array A1 is: ";

print\_all\_array1(A1);

max\_min\_grouping(A1, G1, A1.size(), G1.size());

cout << "The size of G1 is:"<<M1<<" and the array G1 which describes the grouping of array A1 is:";

print\_all\_array1(G1);

cout << endl;

Matrix<int, 1>A2 = C2;

Matrix<int, 1>G2(M2);

cout << "The original array A2 is: ";

print\_all\_array1(A2);

max\_min\_grouping(A2, G2, A2.size(), G2.size());

cout << "The size of G2 is:" << M2 << " and the array G2 which describes the grouping of array A2 is:";

print\_all\_array1(G2);

cout << endl;

Matrix<int, 1>A3 = C3;

Matrix<int, 1>G3(M3);

cout << "The original array A3 is: ";

print\_all\_array1(A3);

max\_min\_grouping(A3, G3, A3.size(), G3.size());

cout << "The size of G3 is:" << M3 << " and the array G3 which describes the grouping of array A3 is:";

print\_all\_array1(G3);

cout << endl;

Matrix<int, 1>A4 = C4;

Matrix<int, 1>G4(M4);

cout << "The original array A4 is: ";

print\_all\_array1(A4);

max\_min\_grouping(A4, G4, A4.size(), G4.size());

cout << "The size of G4 is:" << M4 << " and the array G4 which describes the grouping of array A4 is:";

print\_all\_array1(G4);

cout << endl;

Matrix<int, 1>A5 = C5;

Matrix<int, 1>G5(M5);

cout << "The original array A5 is: ";

print\_all\_array1(A5);

max\_min\_grouping(A5, G5, A5.size(), G5.size());

cout << "The size of G5 is:" << M5 << " and the array G5 which describes the grouping of array A5 is:";

print\_all\_array1(G5);

cout << endl;

Matrix<int, 1>A6 = C6;

Matrix<int, 1>G6(M6);

cout << "The original array A6 is: ";

print\_all\_array1(A6);

max\_min\_grouping(A6, G6, A6.size(), G6.size());

cout << "The size of G6 is:" << M6 << " and the array G6 which describes the grouping of array A6 is:";

print\_all\_array1(G6);

}

The head file “Matrix.h” is in the file “First Submission.zip”, and I will not list the codes here for that the codes are too long.

**Part5 Summary**

At last, I successfully design an algorithm and codes to group one array as requests. As the results show, the codes can work very well and fit the bill.

The problem is that I do not design a beautiful interface for users, like GUI. This is because my C++’s skill is not so good. I believe in several days’ study, I can address this problem.