Project 1_Report

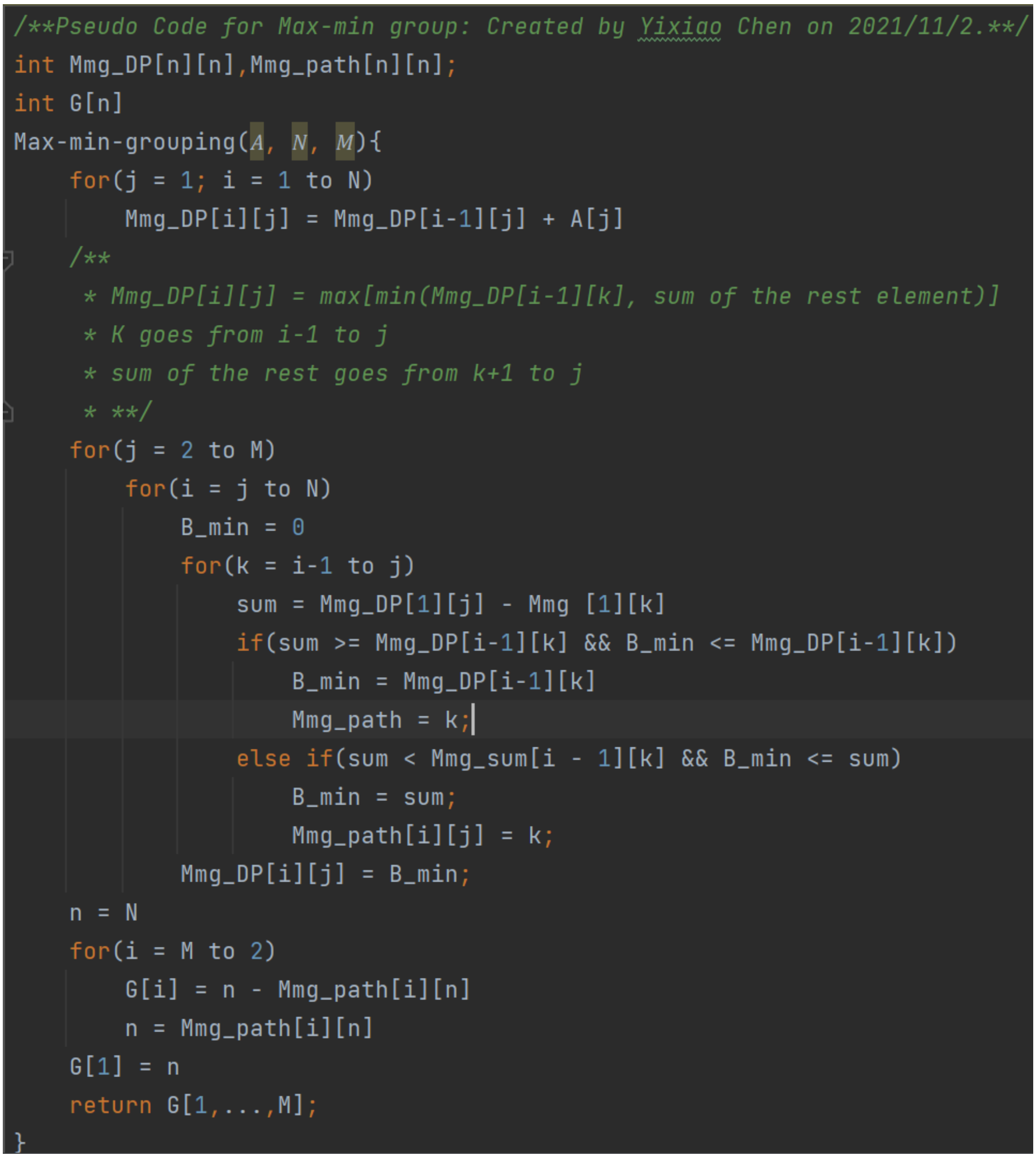
Yixiao Chen_002198256

Problem:

Problem Description:

You are given an input array $A[1,\ldots,N]$. A grouping of the array A is described by an array $G[1,\ldots,M]$, 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 $B[1,\ldots,M]$ 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.

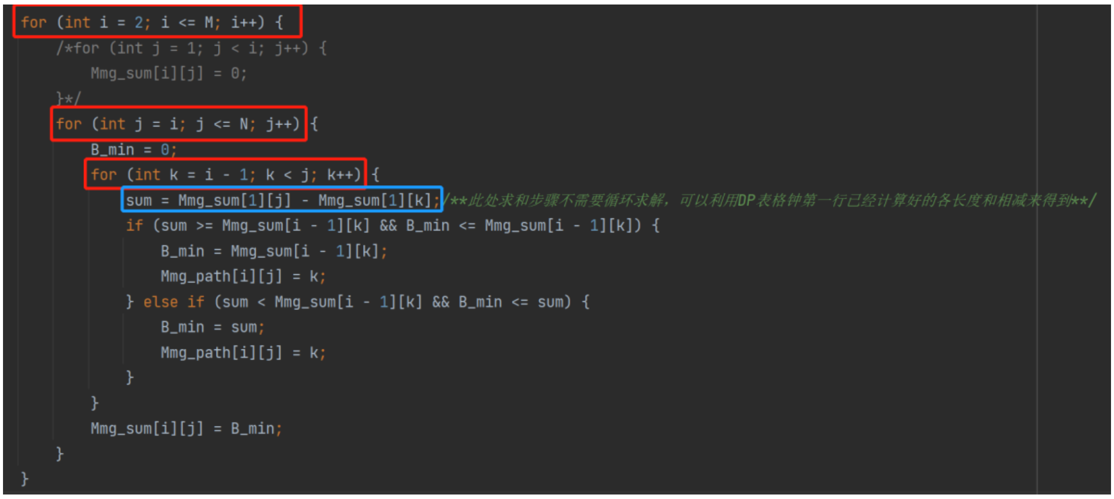
1. Pseudo codes of your dynamic programming algorithm.



```
/**Pseudo Code for Max-min group: Created by Yixiao Chen on 2021/11/2.**/
int Mmg_DP[n][n],Mmg_path[n][n];
int G[n]
Max-min-grouping(A, N, M){
    for(j = 1; i = 1 to N)
        Mmg_DP[i][j] = Mmg_DP[i-1][j] + A[j]
    /**
     * Mmg_DP[i][j] = max[min(Mmg_DP[i-1][k], sum of the rest element)]
     * K goes from i-1 to j
     * sum of the rest goes from k+1 to j
     * **/
    for(j = 2 to M)
        for(i = j to N)
            B_min = 0
            for(k = i-1 to j)
                sum = Mmg_DP[1][j] - Mmg [1][k]
                if(sum >= Mmg_DP[i-1][k] && B_min <= Mmg_DP[i-1][k])
                    B_min = Mmg_DP[i-1][k]
                    Mmg_path = k;
                else if(sum < Mmg_sum[i - 1][k] && B_min <= sum)
                    B_min = sum;
                    Mmg_path[i][j] = k;
            Mmg_DP[i][j] = B_min;
    n = N
    for(i = M to 2)
        G[i] = n - Mmg_path[i][n]
        n = Mmg_path[i][n]
    G[1] = n
    return G[1,...,M];
}
```

2. Analysis of the running time asymptotically.

Here below is the most important part of my source code. About how the DP matrix of this Max-min-group problem was formed during the process.



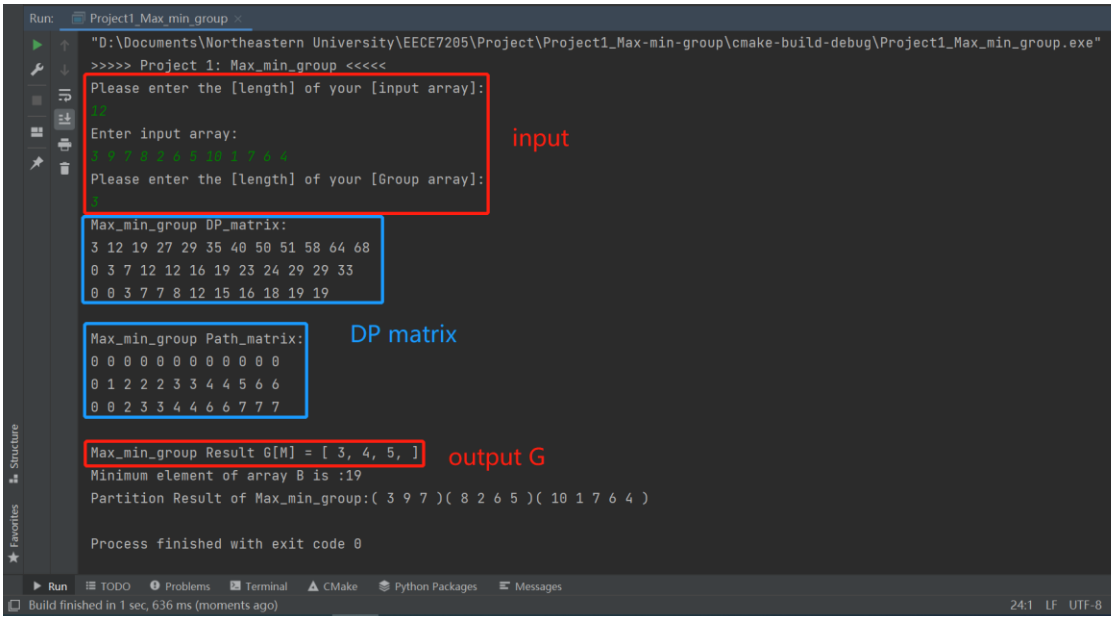
```
for (int i = 2; i <= M; i++) {
    /*for (int j = 1; j < i; j++) {
        Mmg_sum[i][j] = 0;
    }*/
    for (int j = i; j <= N; j++) {
        B_min = 0;
        for (int k = i - 1; k < j; k++) {
            sum = Mmg_sum[1][j] - Mmg_sum[1][k];/**此处求和步骤不需要循环求解，可以利用DP表格钟第一行已经计算好的各长度和相减来得到**/
            if (sum >= Mmg_sum[i - 1][k] && B_min <= Mmg_sum[i - 1][k]) {
                B_min = Mmg_sum[i - 1][k];
                Mmg_path[i][j] = k;
            } else if (sum < Mmg_sum[i - 1][k] && B_min <= sum) {
                B_min = sum;
                Mmg_path[i][j] = k;
            }
        }
        Mmg_sum[i][j] = B_min;
    }
}
```

According to my Source Code (or according to my Pseudo Code). The running time of this algorithm is mainly affected by this ‘triple loop’ which is **O(M*N*N)**.M is the number of groups, and N is the number of the elements in the input array.

From another point of view. To fill in the DP matrix. There’s going to be **M*N elements** in total. While calculating each element, we need to run a for loop of **k** from i-1 to j (#groups & #elements). The upper bound of this k would be the total number of the input elements, which is **N**. So the asymptotic analysis of running time would be **O(M*N*N)**.

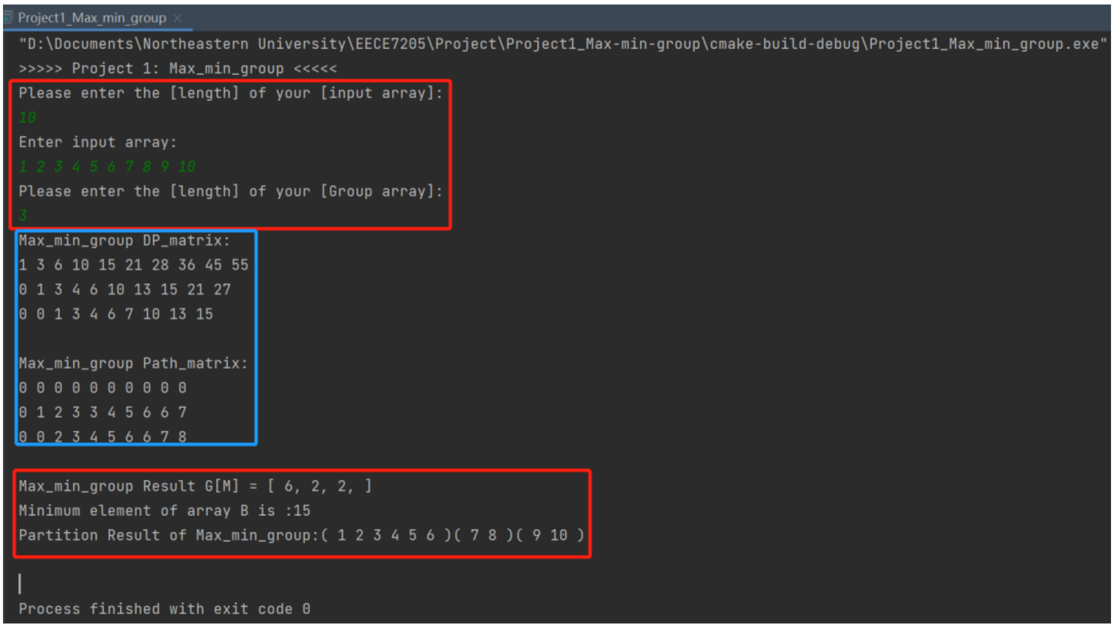
3. Grouping results of several input examples including the one that A={3,9,7,8,2,6,5,10,1,7,6,4} and M=3.

[Result #1:] A={3,9,7,8,2,6,5,10,1,7,6,4} and M=3



```
"D:\Documents\Northeastern University\EECE7205\Project\Project1_Max-min-group\cmake-build-debug\Project1_Max_min_group.exe"
>>>>> Project 1: Max_min_group <<<<<
Please enter the [length] of your [input array]:
12
Enter input array:
3 9 7 8 2 6 5 10 1 7 6 4
Please enter the [length] of your [Group array]:
3
Max_min_group DP_matrix:
3 12 19 27 29 35 40 50 51 58 64 68
0 3 7 12 12 16 19 23 24 29 29 33
0 0 3 7 7 8 12 15 16 18 19 19

Max_min_group Path_matrix:
0 0 0 0 0 0 0 0 0 0 0 0
0 1 2 2 2 3 3 4 4 5 6 6
0 0 2 3 3 4 4 6 6 7 7 7

Max_min_group Result G[M] = [ 3, 4, 5, ]
Minimum element of array B is :19
Partition Result of Max_min_group:( 3 9 7 )( 8 2 6 5 )( 10 1 7 6 4 )

Process finished with exit code 0
```

[Result #2:] A={1,2,3,4,5,6,7,8,9,10} and M=3



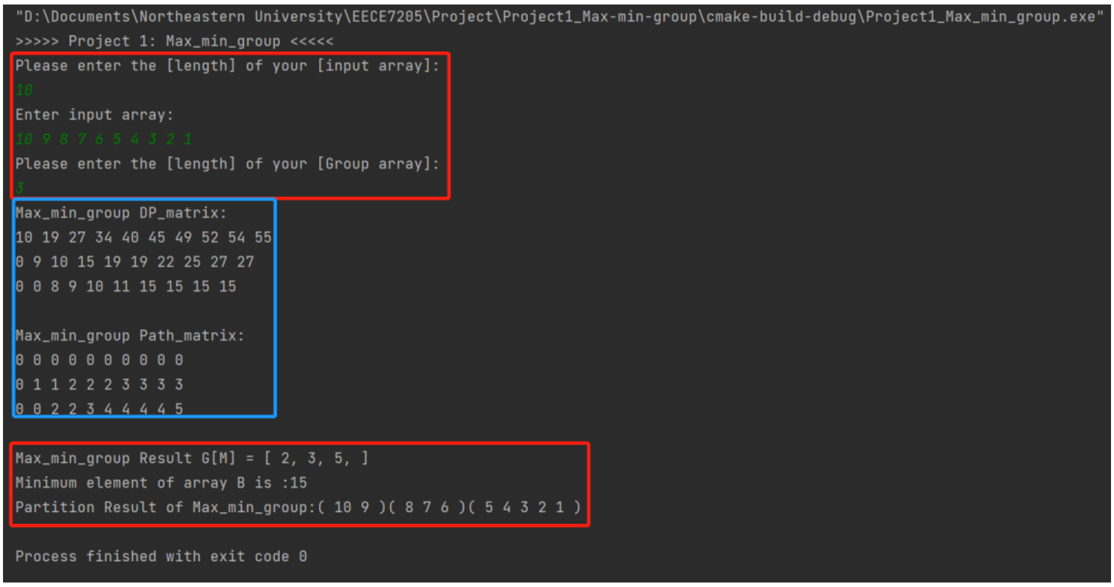
```
"D:\Documents\Northeastern University\EECE7205\Project\Project1_Max-min-group\cmake-build-debug\Project1_Max_min_group.exe"
>>>>> Project 1: Max_min_group <<<<<
Please enter the [length] of your [input array]:
10
Enter input array:
1 2 3 4 5 6 7 8 9 10
Please enter the [length] of your [Group array]:
3
Max_min_group DP_matrix:
1 3 6 10 15 21 28 36 45 55
0 1 3 4 6 10 13 15 21 27
0 0 1 3 4 6 7 10 13 15

Max_min_group Path_matrix:
0 0 0 0 0 0 0 0 0 0
0 1 2 3 3 4 5 6 6 7
0 0 2 3 4 5 6 6 7 8

Max_min_group Result G[M] = [ 6, 2, 2, ]
Minimum element of array B is :15
Partition Result of Max_min_group:( 1 2 3 4 5 6 )( 7 8 )( 9 10 )

|
Process finished with exit code 0
```

Note:

There are two possible correct answers for this input G=[5,3,2]/G=[6,2,2].

Both have the same result of B_min = 15.

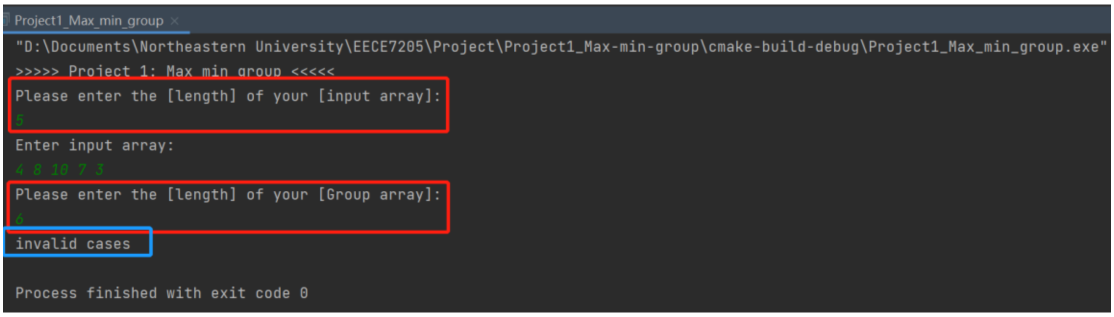
[Result #3:] A={10,9,8,7,6,5,4,3,2,1} and M=3



```
"D:\Documents\Northeastern University\EECE7205\Project\Project1_Max-min-group\cmake-build-debug\Project1_Max_min_group.exe"
>>>>> Project 1: Max_min_group <<<<<
Please enter the [length] of your [input array]:
10
Enter input array:
10 9 8 7 6 5 4 3 2 1
Please enter the [length] of your [Group array]:
3
Max_min_group DP_matrix:
10 19 27 34 40 45 49 52 54 55
0 9 10 15 19 19 22 25 27 27
0 0 8 9 10 11 15 15 15 15

Max_min_group Path_matrix:
0 0 0 0 0 0 0 0 0 0
0 1 1 2 2 2 3 3 3 3
0 0 2 2 3 4 4 4 4 5

Max_min_group Result G[M] = [ 2, 3, 5, ]
Minimum element of array B is :15
Partition Result of Max_min_group:( 10 9 )( 8 7 6 )( 5 4 3 2 1 )

Process finished with exit code 0
```

Illustration of result 2

[Result #4:] A={4,8,10,7,3} and M=6



```
"D:\Documents\Northeastern University\EECE7205\Project\Project1_Max-min-group\cmake-build-debug\Project1_Max_min_group.exe"
>>>>> Project 1: Max_min_group <<<<<
Please enter the [length] of your [input array]:
5
Enter input array:
4 8 10 7 3
Please enter the [length] of your [Group array]:
6
invalid cases

Process finished with exit code 0
```

Special cases When M>N (invalid input)

4. Source codes.

```
#include <iostream>

using namespace std;
const int n = 50;/**此处根据实际情况预设 DP 二维数组 大小**/
int Mmg_sum[n][n], Mmg_path[n][n];
int sum, B_min, number;

void Mmg_output_G(int G[], int N, int M) {
    //G[0] = 1;
    number = N;
    for (int i = M; i >= 1; i--) {
        G[i] = number - Mmg_path[i][number];
        number = Mmg_path[i][number];
        if (i == 2) {
            G[i - 1] = number;/**输出 G 的第一位无需继续循环，可以直接由当前情况得出。**/
            break;
        }
    }
}

void print_partition(int a[], int G[], int M, int N) {
    int count1 = 0, count2 = 0;/**此处采用了两个变量分别划定每组输出的起始和结尾下标；也可以只用一个变量实现。**/
    for (int i = 1; i <= M; i++) {
        count1 = count1 + G[i];
        cout << "( ";
        for (int j = count2 + 1; j <= count1; j++) {
            cout << a[j] << " ";
        }
        count2 = count2 + G[i];
        cout << ")";
    }
    cout << endl;
}

/**此处 Max_min_km 功能函数通过递归调用实现 k 自增的比较 算法时间复杂
```

```
度不会因为递归或者循环而改变**/
/*void Max_min_km(int a[], int k, int i, int j) {
    if (k >= j) {
        Mmg_sum[i][j] = B_min;
        return;
    } else {
        //sum = 0;
        //for (int m = k + 1; m <= j; m++) {
        //    sum = sum + a[m];
        //}
        sum = Mmg_sum[1][j] - Mmg_sum[1][k];//此处求和步骤不需要循环求解，
可以利用 DP 表格钟第一行已经计算好的各长度和相减来得到
        if (sum >= Mmg_sum[i - 1][k] && B_min <= Mmg_sum[i - 1][k]) {
            B_min = Mmg_sum[i - 1][k];
            Mmg_path[i][j] = k;
        } else if (sum < Mmg_sum[i - 1][k] && B_min <= sum) {
            B_min = sum;
            Mmg_path[i][j] = k;
        }
        Max_min_km(a, k + 1, i, j);
    }
}*/

void Max_min_group(int a[], int N, int M) {/**此处可以选择给数组 0 行 0 列赋值
为 0，不进行此赋值也不影响过程和结果。**/
    /*for (int i = 0; i <= M; i++) {
        Mmg_sum[i][0] = 0;
    }
    for (int j = 0; j <= N; j++) {
        Mmg_sum[0][j] = 0;
    }*/
    Mmg_sum[1][0] = 0;
    for (int j = 1; j <= N; j++) {
        Mmg_sum[1][j] = Mmg_sum[1][j - 1] + a[j];/**此处稍微注意是否需要将
[1][0]提前赋 0**/
        /*sum = 0;
        for (int s = 0; s <= j; s++) {
            sum = sum + a[s];
        }
        Mmg_sum[1][j] = sum;*/
    }
    /**
     *  此处以下代码部分可以通过 Max_min_km 功能函数进行递归调用实现
     *  也可以通过三层循环解决 两者算法时间复杂度一样
     * **/
    for (int i = 2; i <= M; i++) {
        /*for (int j = 1; j < i; j++) {
            Mmg_sum[i][j] = 0;
        }*/
        for (int j = i; j <= N; j++) {
            B_min = 0;
            for (int k = i - 1; k < j; k++) {
                sum = Mmg_sum[1][j] - Mmg_sum[1][k];/**此处求和步骤不需要循环求解，可以利用 DP 表格钟第一行已经计算好的各长度和相减来得到**/
                if (sum >= Mmg_sum[i - 1][k] && B_min <= Mmg_sum[i - 1][k]) {
                    B_min = Mmg_sum[i - 1][k];
                    Mmg_path[i][j] = k;
                } else if (sum < Mmg_sum[i - 1][k] && B_min <= sum) {
                    B_min = sum;
                    Mmg_path[i][j] = k;
                }
            }
            Mmg_sum[i][j] = B_min;
        }
    }
    /*for (int i = 2; i <= M; i++) {
        for (int j = 1; j < i; j++) {
            Mmg_sum[i][j] = 0;
        }
        for (int j = i; j <= N; j++) {
            int k = i - 1;
            B_min = 0;
            Max_min_km(a, k, i, j);
        }
    }*/
```

```
}

void print_Mmg(int a[n][n], int N, int M) {
	for (int i = 1; i <= M; i++) {
		for (int j = 1; j <= N; j++) {
			cout << a[i][j] << " ";
		}
		cout << endl;
	}
	cout << endl;
}

/**代码编辑初期  调试所用主函数**/
/*int main() {
	cout << ">>>>> Project 1: Max_min_group <<<<<" << endl;
	int N = 12;///输入目标数组的长度
	int a[13] = {0, 3, 9, 7, 8, 2, 6, 5, 10, 1, 7, 6, 4};///int a[N+1];循环 cin
	int M = 3;
	int G[4];///int G[M+1]
	Max_min_group(a, N, M);
	cout << "Max_min_group DP_matrix:" << endl;
	print_Mmg(Mmg_sum, N, M);
	cout << "Max_min_group Path_matrix:" << endl;
	print_Mmg(Mmg_path, N, M);
	cout << "Max_min_group Result G[M] = [ ";
	Mmg_output_G(G, N, M);
	for (int i = 1; i <= M; i++) {
		cout << G[i] << ", ";
	}
	cout << "]" << endl;
	cout << "Minimum element of array B is :" << Mmg_sum[M][N] << endl;///输出
DP 表格 C[i][j]的最后一个元素
	cout << "Partition Result of Max_min_group:";
	print_partition(a, G, M, N);
	getchar();
	return 0;
}*/
```

```
/**可自定义输入的程序所对应主函数**/
int main() {
    int N, M;
    cout << ">>>>> Project 1: Max_min_group <<<<<" << endl;
    cout << "Please enter the [length] of your [input array]:" << endl;
    cin >> N;
    int a[N + 1];
    a[0] = 0;///为处理后续数组下标以及 DP 二维数组下标问题  将输入输入改为由 0 开始
    cout << "Enter input array:" << endl;
    for (int i = 1; i <= N; i++) {
        cin >> a[i];
    }
    cout << "Please enter the [length] of your [Group array]:" << endl;
    cin >> M;
    if(M>N){
        cout<< "invalid cases"<<endl;///处理特殊情况
        return 0;
    }
    int G[M+1];///此处同上  也是为了统一各数组下标
    Max_min_group(a, N, M);
    cout << "Max_min_group DP_matrix:" << endl;
    print_Mmg(Mmg_sum, N, M);
    cout << "Max_min_group Path_matrix:" << endl;
    print_Mmg(Mmg_path, N, M);
    cout << "Max_min_group Result G[M] = [ ";
    Mmg_output_G(G, N, M);
    for (int i = 1; i <= M; i++) {
        cout << G[i] << ", ";
    }
    cout << "]" << endl;
    cout << "Minimum element of array B is :" << Mmg_sum[M][N] << endl;/**输出 DP 表格 C[i][j]的最后一个元素**/
    cout << "Partition Result of Max_min_group:";
    print_partition(a, G, M, N);
    getchar();
    return 0;
}
```