NORTHEASTERN UNIVERSITY

PROJECT

EECE 7205

SECTION V30

PROFESSOR Xue Lin

BY

ANNA DEVRIES

24 OCTOBER 2019

Table of Contents

Project

Project 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.

Max-min-grouping(A, N, M) {

return $G[1,\ldots,M]$ }

Hint:

- The optimal subproblem property: suppose the optimal solution to Max-min-grouping(A,N,M) is $G[1,\ldots,M]$=[n_1, n_2,…, n_{M-1},n_M]. Then $G[1,\ldots,M-1]$ is the optimal solution to the subproblem Max-min-grouping(A,N-n_m,M-1).
- See Algorithm 2 in the paper – $I_L(G) = \frac{G}{G_{STC}} * I_L(G_{STC})$

Pseudo Code

```
Max-min-grouping(A[], N, M):
     for j <- 1 to M:
          for i <- 1 to N:
               C[i][j] = 0
     for j <- 1 to M:
          G[j] = 0

     C[0][0] = A[0]
     for i <- 1 to N:
          C[i][0] <- C[i-1][0] + A[i]

     for i <- 1 to N:
          for j <- 1 to M:
               if i < j:
                    C[i][j] = 0
               else:
                    for x <- 0 to i:
                         B[x] = 0
                    for k <- j-1 to i:
                         B[k] = min(C[k][j-1], C[i][0]-C[k][0])
                    C[i][j] = max(B)

     j <- M - 1
     for i <- N-1 to 0:
          largest = getnextlargest(C[N][M-1])
          if C[i][j] == largest:
               G[j + 1] = counter
               counter = 0
               j—

          if C[i][j] > largest:
               if largest == 0:
                    print “No Solution”
                    break
          counter++

     return G
```

Analysis of Running Time Asymptotically

Worst Case Analysis

Worst case running time refers to the longest running time for any input size n. In this algorithm, the worst-case running time occurs every time. Therefore, the worst-case running time equates to both average-case and best-case running times. The reason for this analysis is the table construction. Independent of the values in array A[], table C[i][j] and array B[i] are created and initialized to 0 every run. In this algorithm, these three nested-for-loops are the most time-consuming steps.

```
for i <- 1 to N:                                  T(n) = θ(N)
  for j <- 1 to M:                                T(n) = θ(M)
    if i < j:                                     T(n) = θ(1)
      C[i][j] = 0
    else:
      for x <- 0 to i:                            T(n) = θ(N)
        B[x] = 0
      for k <- j-1 to i:                          T(n) = θ(N)
        B[k] = min(C[k][j-1], C[i][0]-C[k][0])    T(n) = θ(1)
      C[i][j] = max(B)                            T(n) = θ(1)
```

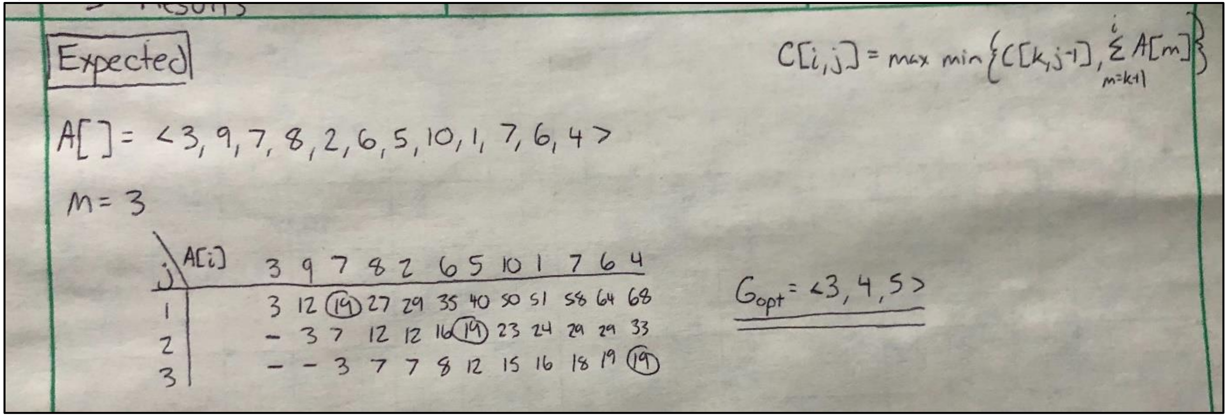
Since “`for x <- 0 to i`” and “`for k <- j-1 to i`” are executed in line, the time complexity overall will only be T(n) = θ(N) for these for-loops. Therefore, the overall time complexity is $T(n) = \theta(N) * \theta(N) * \theta(M) = \theta(M * N^2)$.

Results and Examples

Example 1:

$$A = \{3, 9, 7, 8, 2, 6, 5, 10, 1, 7, 6, 4\}\ and\ M = 3$$
$$Expected\ G_{opt} = \{3,4,5\}$$
$$Actual\ G_{opt} = \{3,4,5\}$$

Figure 1: Calculations for expected G.



Expected

$C[i,j] = max\ min\left\{C[k,j-1], \sum_{m=k+1}^{i} A[m]\right\}$

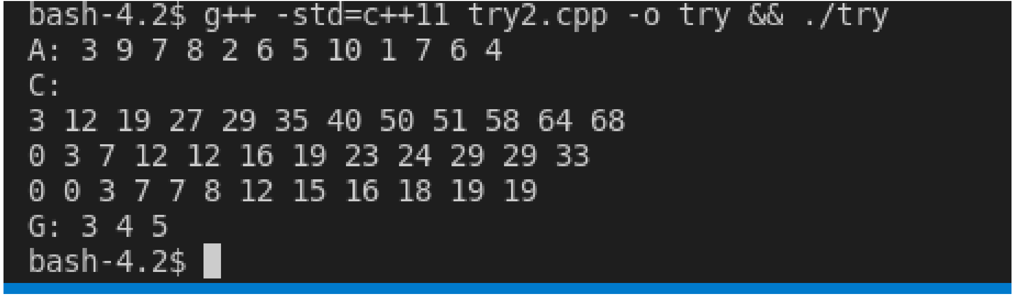
A[ ] = <3, 9, 7, 8, 2, 6, 5, 10, 1, 7, 6, 4>

m = 3

j \ A[i]	3	9	7	8	2	6	5	10	1	7	6	4
1	3	12	19	27	29	35	40	50	51	58	64	68
2	–	3	7	12	12	16	19	23	24	29	29	33
3	–	–	3	7	7	8	12	15	16	18	19	19

$G_{opt} = <3, 4, 5>$

Figure 2: Program output.

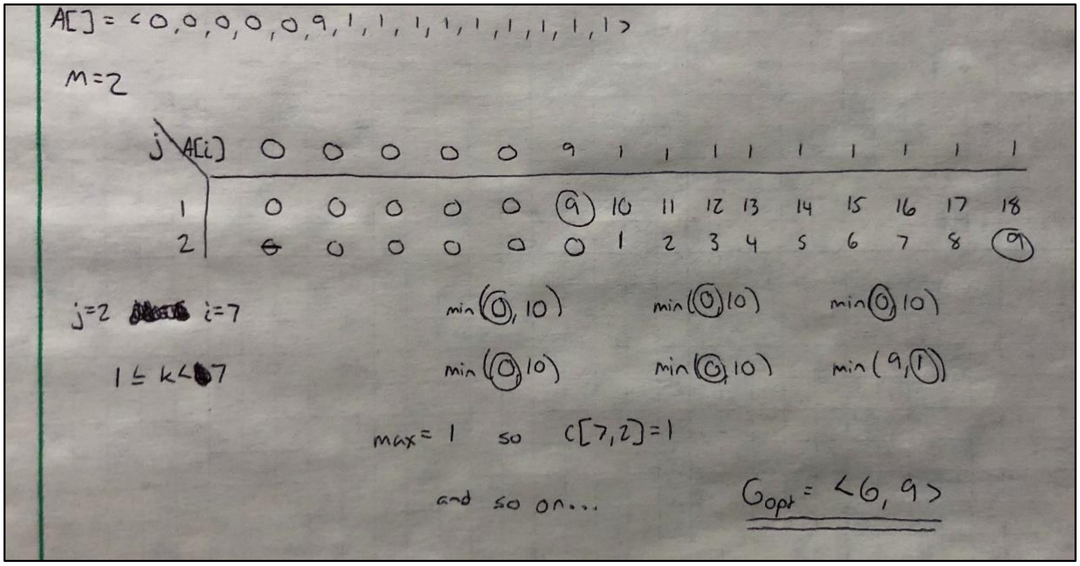


```
bash-4.2$ g++ -std=c++11 try2.cpp -o try && ./try
A: 3 9 7 8 2 6 5 10 1 7 6 4
C:
3 12 19 27 29 35 40 50 51 58 64 68
0 3 7 12 12 16 19 23 24 29 29 33
0 0 3 7 7 8 12 15 16 18 19 19
G: 3 4 5
bash-4.2$
```

Example 2:

$$A = \{0,0,0,0,0,9,1,1,1,1,1,1,1,1,1\}\ and\ M = 2$$
$$Expected\ G_{opt} = \{6,9\}$$
$$Actual\ G_{opt} = \{6,9\}$$

Figure 3: Calculations for expected G.



A[] = <0,0,0,0,0,9,1,1,1,1,1,1,1,1,1>

M=2

j \ A[i]	0	0	0	0	0	9	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	9	10	11	12	13	14	15	16	17	18
2	~~0~~	0	0	0	0	0	1	2	3	4	5	6	7	8	9

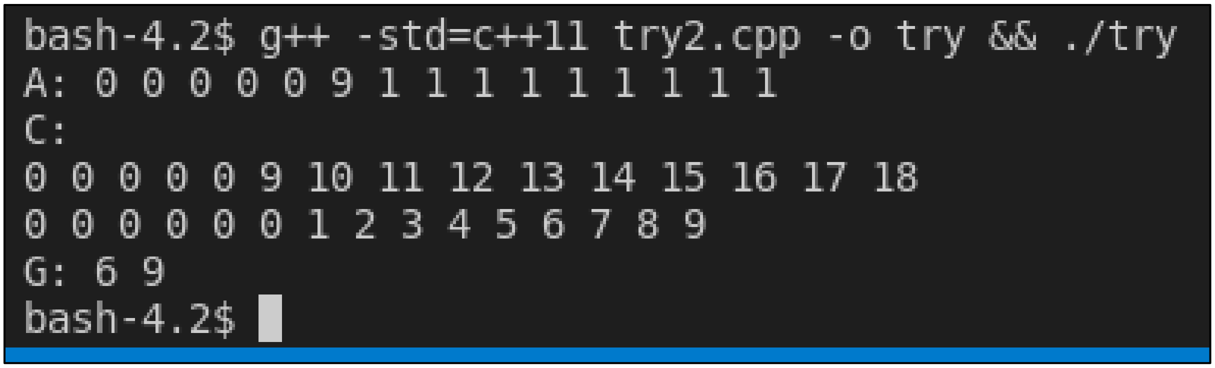
j=2 i=7 min(0, 10) min(0,10) min(0,10)

$1 \leq k < 7$ min(0,10) min(0,10) min(9,1)

max= 1 so C[7,2]=1

and so on... G_{opt} = <6, 9>

Figure 4: Program output.

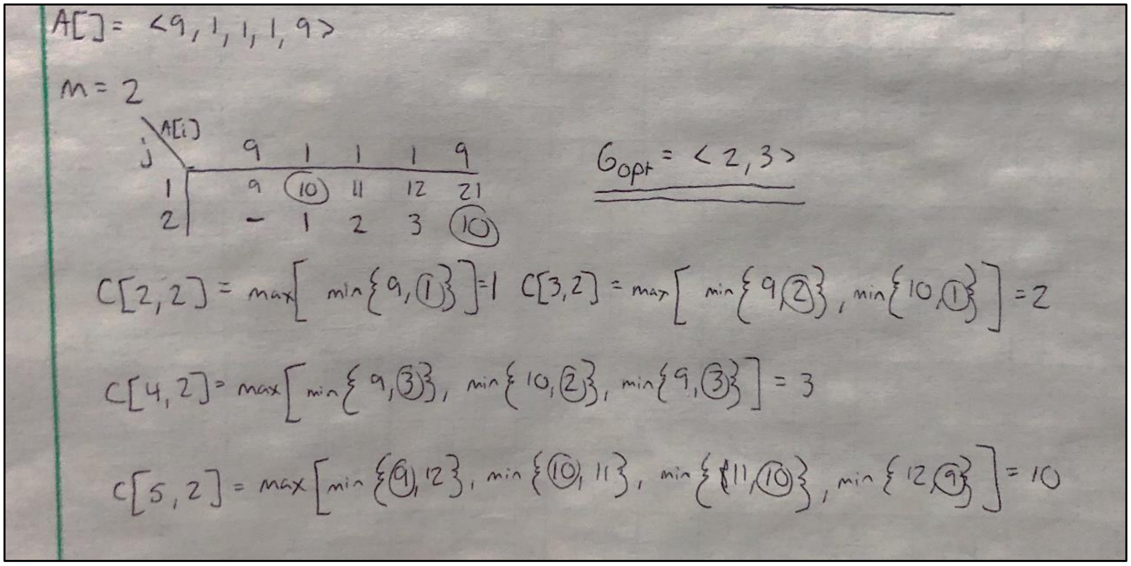


```
bash-4.2$ g++ -std=c++11 try2.cpp -o try && ./try
A: 0 0 0 0 0 9 1 1 1 1 1 1 1 1 1
C:
0 0 0 0 0 9 10 11 12 13 14 15 16 17 18
0 0 0 0 0 0 1 2 3 4 5 6 7 8 9
G: 6 9
bash-4.2$ 
```

Example 3:

$$A = \{9,1,1,1,9\}\ and\ M = 2$$
$$Expected\ G_{opt} = \{2,3\}$$
$$Actual\ G_{opt} = \{2,3\}$$

Figure 5: Calculations for expected G.



A[] = <9,1,1,1,9>

m = 2

j \ A[i]	9	1	1	1	9
1	9	10	11	12	21
2	–	1	2	3	10

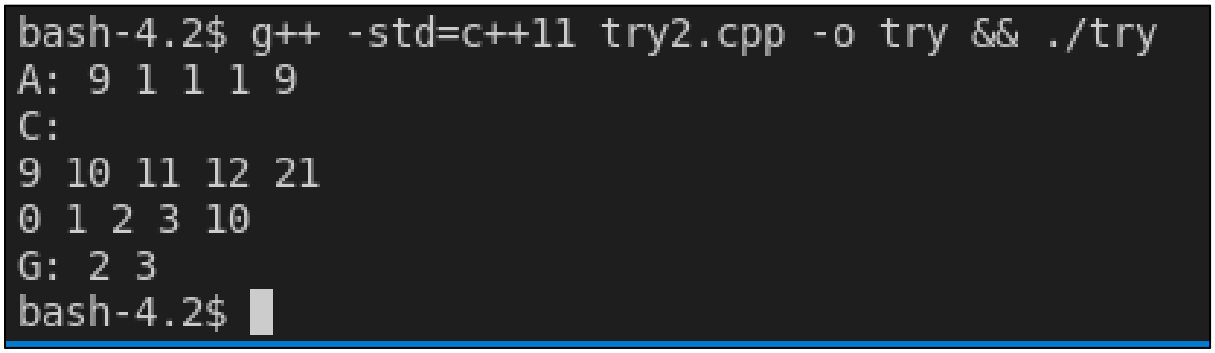
$G_{opt} = <2,3>$

$C[2,2] = max[\ min\{9,1\}\ ] = 1 \quad C[3,2] = max[\ min\{9,2\},\ min\{10,1\}\ ] = 2$

$C[4,2] = max[min\{9,3\},\ min\{10,2\},\ min\{9,3\}] = 3$

$C[5,2] = max[min\{9,12\},\ min\{10,11\},\ min\{11,10\},\ min\{12,9\}] = 10$

Figure 6: Program output.

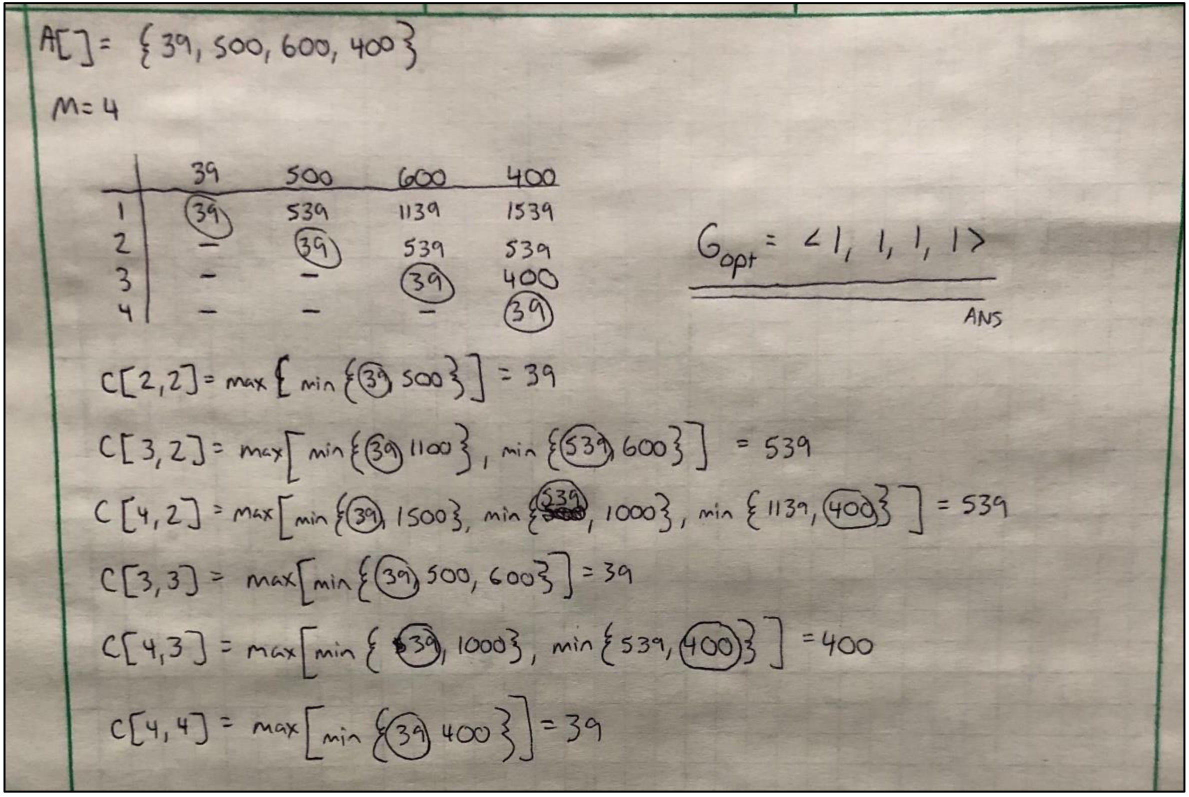


```
bash-4.2$ g++ -std=c++11 try2.cpp -o try && ./try
A: 9 1 1 1 9
C:
9 10 11 12 21
0 1 2 3 10
G: 2 3
bash-4.2$ 
```

Example 4:

$$A = \{39, 500, 600, 400\}\ and\ M = 4$$
$$Expected\ G_{opt} = \{1,1,1,1\}$$
$$Actual\ G_{opt} = \{\}$$

Figure 7: Calculations for expected G.



A[] = {39, 500, 600, 400}

M = 4

	39	500	600	400
1	39	539	1139	1539
2	–	39	539	539
3	–	–	39	400
4	–	–	–	39

$G_{opt} = \langle 1, 1, 1, 1 \rangle$

ANS

C[2,2] = max{ min{39, 500}] = 39

C[3,2] = max[min{39, 1100}, min{539, 600}] = 539

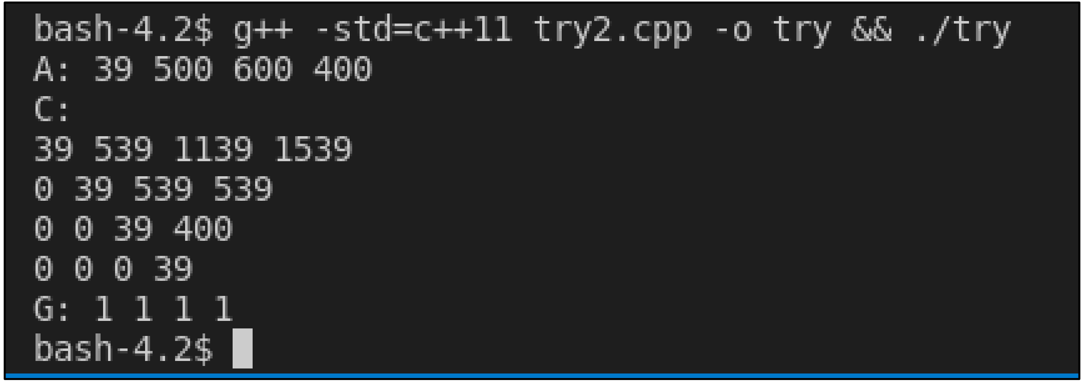
C[4,2] = max[min{39, 1500}, min{539, 1000}, min{1139, 400}] = 539

C[3,3] = max[min{39, 500, 600}] = 39

C[4,3] = max[min{39, 1000}, min{539, 400}] = 400

C[4,4] = max[min{39, 400}] = 39

Figure 8: Program output.



```
bash-4.2$ g++ -std=c++11 try2.cpp -o try && ./try
A: 39 500 600 400
C:
39 539 1139 1539
0 39 539 539
0 0 39 400
0 0 0 39
G: 1 1 1 1
bash-4.2$
```

Source Code

```cpp
#include <iostream>
#include <stdio.h>
#include <algorithm>
#include <stack>

// Comparison function.
// Determines max element.
bool comp(int a, int b){
        return (a < b);
}

// Max-Min-Grouping function.
// Optimize grouping of elements into subarrays such that Bmin is maximized.
int* max_min_grouping(int A[],int N,int M){
// Initialize arrays and variables.
    int* G = (int*)malloc(sizeof(int) * M);
    int C[N][M];
    int* maxi;
    int i, j, k, x, sum;

// Print A[].
    std::cout<<"A: ";
    for(i = 0;i < N; i++){
        std::cout<<A[i]<<" ";
    }
    std::cout<<std::endl;

// Initialize C[] and set all elements to 0.
    for(j = 0; j < M; j++){
        for(i = 0; i < N; i++){
            C[i][j] = 0;
        }
    }

// Initialize G[] and set all elements to 0.
    for(j = 0; j < M; j++){
        G[j] = 0;
    }

// Fill in row 1 of table C.
    C[0][0] = A[0];
    for(i = 1; i < N; i++){
// For current index, sum value index in A[] and value index-1 in C[].
        C[i][0] = C[i - 1][0] + A[i];
    }

// Fill in the rest of the table, beginning at row 2 of table C.
    for (i = 0; i < N; i++){
        for (j = 1; j < M; j++){

// If index value is less than the amount of subarrays, then this state is impossible and 0 is placed into table C.
            if(i < j){
                C[i][j] = 0;
            }

// Otherwise, case is functional.
// Initialize B[] as a temporary holding array of values.
            else{
                int B[i + 1];
                for(x = 0; x <= i; x++){
                     B[x] = 0;
                }

// Sum values of k using previous table entries.
// Compare sum and value at C[index]. Store min value into temporary array B[].
                for(k = j - 1; k < i; k++){
                    sum = C[i][0] - C[k][0];
                    B[k] = std::min(C[k][j - 1], sum);
                }
// Find the max element from temporary array B[] and add to table C.
                maxi = std::max_element(B, B + i, comp);
                C[i][j] = *maxi;
            }
        }
    }

// Print table C[].
    std::cout<<"C: \n";
    for(j = 0;j < M; j++){
        for(i = 0;i < N; i++){
            std::cout<<C[i][j]<<" ";
        }
        std::cout << std::endl;
    }

// Pushes last row values 0 to len(A[]) onto stack and places largest values on top.
    std::stack<int> values;
    for(i = 0; i < N; i++){
        values.push(C[i][M-1]);
    }

// Initialize variables and pops last result.
    int largest = values.top();
    values.pop();

    int counter = 1;
    int last = N;
    j = M-1;

// Compare last result of the last row with values in other rows.
    for(i = N - 1; i >= 0; i--){
// If the next value equals largest, counter is added to G[] and restarted for next row.
        if(C[i][j] == largest){
            G[j + 1] = counter;
            last -= counter;
            counter = 0;
            j--;
        }
// If the next value is less than largest, the next largest value in the last row is popped.
        if(C[i][j] < largest){
            largest = values.top();
            if(largest == 0){
                if(j >= 0){
                    std::cout << "No Solution!\n";
                }
                break;
            }
            values.pop();
        }
        counter++;
    }
    G[0] = last+1;

    return G;

}

int main(){
// Example 1.
    int A[] = {3, 9, 7, 8, 2, 6, 5, 10, 1, 7, 6, 4};
    int M = 3;
// Expected: {3,4,5}

// Example 2.
    // int A[] = {0, 0, 0, 0, 0, 9, 1, 1, 1, 1, 1, 1, 1, 1, 1};
    // int M = 2;
// Expected: {6, 9}

// Example 3.
    // int A[] = {9, 1, 1, 1, 9};
    // int M = 2;
// Expected: {2, 3}

// Example 4.
    // int A[] = {39, 500, 600, 400};
    // int M = 4;
// Expected: {1, 1, 1, 1}

// Example 5.
//     int A[] = {3, 9, 7, 8, 2, 6, 5, 10, 1, 7, 6, 4};
//     int M = 11;
// Expected: No Solution

    int N = sizeof(A)/sizeof(A[0]);
    int* ptr = max_min_grouping(A, N, M);

// Print G[] optimal.
    std::cout<<"G: ";
    for(size_t i = 0; i < M; i++){
        std::cout<<ptr[i]<<" ";
    }
    std::cout<<std::endl;

    return 0;
}
```