

Contents

1 語法	1
1.1 c++	1
1.2 python	1
2 字串	2
2.1 KMP	2
3 數論	2
3.1 喵	2
3.2 Fibonaccimal	2
3.3 LCM	3
3.4 LCS	3
3.5 LPS	3
3.6 Pairty	3
4 圖論	3
4.1 最短路徑 dijkstra	3
4.2 DFS	4
4.3 merge sort	5
4.4 quick sort	5
4.5 二分搜	6
5 dp	9
5.1 階乘 1	10
5.2 階乘 2	11
5.3 階梯問題	12
5.4 極值問題 (格子有權重)	13

1 語法

1.1 c++

```

1 // c++ code
2 #include <bits/stdc++.h>
3 using namespace std;
4 int main(){
5     std::ios::sync_with_stdio(false); // 加速
6     return 0;
7 }
8
9 // struct宣告
10 struct s{
11     int x[100];
12     int y[100];
13 };
14
15 s num; //一組s
16 num.x[1]=1;
17 num.y[1]=2;
18
19 // sort
20 sort(v.begin(), v.end()) // array 不能用
21 sort(v, v+n)
22 sort(v, v+n, greater<int>()) // 大到小
23
24 sort(v, v+n, cmp) // 自己寫比較序列
25 bool cmp(型態 a, 型態 b){
26     return a > b; // 大到小
27 }
28
29 // set
30 s.insert(x) //將x插入s中 O(log(n))
31 s.count(x) //回傳x是否存在於s中() O(log(n))
32 s.erase(x) //刪除在s中的x O(log(n))
33 s.clear() //刪除s中所有元素 O(n)
34 s.empty() //回傳是否為空 O(1)
35 s.size() //回傳共有幾個元素 O(1)
36
37 map
38 insert(x) //將x這個pair插入map中 O(log(n))
39 count(x) //回傳x這個key是否在map中 O(log(n))
40 erase(x) //刪除在map中key為x的 O(log(n))
41
42 // vector

```

```

43 vector<int> v // 宣告
44 v.push_back(1) // 推入數字
45 v.pop_back() // 拔出尾端數字

```

1.2 python

```

1 # EOF
2 while True:
3     try:
4         '''
5             你要執行的程式碼
6         '''
7         except EOFError:
8             break
9
10 # 有規定終止條件
11 while True:
12     if a==0:
13         break
14
15 # 數學符號
16 a//=10 # 整除
17 a**b # a^b
18
19 # 邏輯
20 a=True
21 b=False
22 print(a and b) #False
23 print(a or b) #True
24
25 # scan
26 a = int(input())
27 n=list(input().split(' ')) #
28     連續輸入一串用空格隔開的數字
29
30 for i in range(a):
31     c, d = map(int, input().split()) # 連續輸入兩個數
32
33 # print
34 print('for is not a multiple of 11.'.format(a))
35 print(a+" and "+b+" sitting in the tree")
36 print('The parity of ',a,' is ',count,' (mod 2).')
37
38 # 標頭檔math
39 import math
40 math.gcd(a, b, c, d, e) # 最大公約數
41 math.lcm(a, b, c, d, e) # 最小公倍數
42 math.fabs(x) # 絕對值
43 math.isqrt(n) # 整數平方根
44 math.sqrt(x) # 平方根
45 math.pow(x, y) # x^y
46
47 # count
48 c+=b.count("商店") # 用在要計算好幾個字串時
49 c=b.count('1') # 一次算出一串字串有幾個 '1'
50
51 # 進制轉換
52 a = bin(a)[2:] # 10 to 2
53 a = hex(a)[2:] # 10 to 16
54 a = oct(a)[2:] # 10 to 8
55
56 # 大小寫轉換
57 a.lower()
58 a.upper()
59
60 # 取長度
61 a.len()

```

2 字串

2.1 KMP

```

1 #include <iostream>
2 using namespace std;
3
4 void KMP(string text, string pattern)
5 {
6     int m = text.length();
7     int n = pattern.length();
8
9     // 如果模組沒東東
10    if (n == 0)
11    {
12        cout << "The pattern occurs with shift 0";
13        return;
14    }
15
16    // 如果文本的長度小於模組的長度
17    if (m < n)
18    {
19        cout << "Pattern not found";
20        return;
21    }
22
23    // next[i] 存儲下一個最佳部分匹配的索引
24    int next[n + 1];
25
26    for (int i = 0; i < n + 1; i++) {
27        next[i] = 0;
28    }
29
30    for (int i = 1; i < n; i++)
31    {
32        int j = next[i];
33
34        while (j > 0 && pattern[j] != pattern[i]) {
35            j = next[j];
36        }
37
38        if (j > 0 || pattern[j] == pattern[i]) {
39            next[i + 1] = j + 1;
40        }
41    }
42
43    for (int i = 0, j = 0; i < m; i++)
44    {
45        if (text[i] == pattern[j]) // 一樣如果 +1j 下一個檢查
46        {
47            if (++j == n) {
48                cout << "The pattern occurs with shift " << i - j + 1 << endl;
49            }
50        }
51        else if (j > 0)
52        {
53            j = next[j]; // 把她休崩變回來
54            i--; // 還要回去啾啾
55        }
56    }
57 }
58
59 int main()
60 {
61     string text = "ABCABAABCABAC";
62     string pattern = "CAB";
63
64     KMP(text, pattern);
65
66     return 0;
67 }

```

3 數論

3.1 喵

```

1 #include <iostream>
2 using namespace std;
3 int a[20];
4
5 int main() {
6     int cases, target, sticks, num, tmp, i;
7     bool flag;
8     while (cin >> cases){
9         while (cases--){
10             cin >> target;
11             cin >> sticks;
12             for (int i = 0; i < sticks; i++){
13                 cin >> a[i];
14             }
15             num = 1;
16             for (int i = 1; i < sticks; i++){
17                 num <= 1;
18                 num++;
19             }
20             flag = false;
21             for (int _i = 0; _i <= num; _i++){
22                 tmp = 0;
23                 i = _i;
24                 for (int j = 0; j < sticks; j++){
25                     if (i & 1) tmp += a[j];
26                     i >>= 1;
27                 }
28                 if (tmp == target){
29                     flag = true;
30                     break;
31                 }
32             }
33             if (flag) cout << "YES\n";
34             else cout << "NO\n";
35         }
36     }
37 }

```

3.2 Fibonaccimal

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5
6     int N;
7     int Fibonacci[40] = {0, 1}; // 開始的兩個數
8     int i;
9
10    for(i = 2; i < 40; i++){
11        Fibonacci[i] = Fibonacci[i - 1] + Fibonacci[i - 2];
12    }
13
14    scanf("%d", &N);
15
16    while(N--){
17
18        int num;
19        bool flag = false;
20
21        scanf("%d", &num);
22        printf("%d = ", num);
23
24        for(i = 39; i >= 2; i--){
25            if(num >= Fibonacci[i]){
26                num = num - Fibonacci[i];
27                flag = true;
28                printf("%1");
29            }

```

```

30         else if(flag){
31             printf("0");
32         }
33     }
34     printf(" (fib)\n");
35 }
36 }
37
38 return 0;
39 }

```

3.3 LCM

```

1 int GCD(int num1,int num2)
2 {
3     if(num2==0)
4     {
5         return num1;
6     }
7     return GCD(num2,num1%num2);
8 }
9
10 int LCM(int num1,int num2) //2個最小公倍數
11 {
12     return((num1*num2)/GCD(num1,num2));
13 }
14
15 int LCM2(int num1,int num2,int num3) //3個最小公倍數
16 {
17     return((num1*num2)/GCD((num1,num2),num3));
18 }
19
20 int main()
21 {
22     cout<<GCD(6,3);
23     cout<<LCM(6,3);
24     cout<<LCM2(6,3,3);
25
26     return 0;
27 }
28 }

```

3.4 LCS

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int main()
4 {
5     string str1, str2;
6     short lcs[2][1000];
7     while (cin >> str1 >> str2)
8     {
9         lcs[0][0] = str1[0] == str2[0];
10        for (int j=1; j<str2.length(); j++) lcs[0][j]
11            = max(lcs[0][j-1],
12                short{str1[0]==str2[j]});
13        for (int i=1; i<str1.length(); i++)
14        {
15            bool r = i & 1;
16            lcs[r][0] = max(lcs[r^1][0],
17                            short{str1[i]==str2[0]});
18            for (int j=1; j<str2.length(); j++)
19                lcs[r][j] = str1[i]==str2[j] ?
20                    lcs[r^1][j-1] + 1 :
21                    max(lcs[r^1][j], lcs[r][j-1]);
22        }
23        cout <<
24            lcs[(str1.length()-1)&1][str2.length()-1]
25            << '\n';
26    }
27    return 0;
28 }

```

3.5 LPS

```

1 #include<bits/stdc++.h>
2 #include<iostream>
3 using namespace std;
4
5 int main(){
6     string s;
7     cin >> s;
8     int maxlen=0, l, r;
9     for(int i=0; i<s.length(); i++){
10         //奇
11         int x = 0;
12         while((s[i-x]==s[i+x]) && (i-x>=0) &&
13             (i+x<s.length())){ //當找到一個中心點以其為中間然後左右
14             x++;
15         }
16         x--;
17         if(2*x+1 >
18             maxlen){ //只有第一次會max==後後面就追不到那女孩
19             maxlen = 2*x+1; //最大的
20             l = i-x; //計算頭頭
21             r = i+x; //計算尾巴
22         }
23         //偶
24         x = 0;
25         while( (s[i-x]==s[i+1+x]) && (i-x>=0) &&
26             (i+1+x<s.length())) {
27             x++;
28         }
29         if(2*x > maxlen){
30             maxlen = 2*x;
31             l = i-x+1;
32             r = i+x;
33         }
34     }
35     cout << maxlen << '\n';
36     cout << l+1 << ' ' << r+1 << '\n';
37 }

```

3.6 Parity

```

1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4
5 int main() {
6     int I, n;
7     while (cin >> I) {
8         if (I == 0) break;
9         string B = "";
10        n = I;
11        int cnt = 0;
12        while (n){
13            cnt += (n & 1);
14            B += '0' + (n & 1);
15            n >>= 1;
16        }
17        reverse(B.begin(), B.end());
18        cout << "The parity of " << B << " is " <<
19            cnt << " (mod 2).\n";
20    }
21    return 0;
22 }

```

4 圖論

4.1 最短路徑 dijkstra

```

1 // 邊權重皆為正數時使用
2 // 1.
   輸入有總點、總邊，接著輸入點，點，距離（權重）時使用
3 #include <iostream>
4 #include <vector>
5 #include <climits>
6
7 using namespace std;
8
9 // 定義城市數量的上限
10 #define MAX_CITIES 100
11
12 // 定義無限大的距離
13 #define INF INT_MAX
14
15 // 城市數量、道路數量
16 int numCities, numRoads;
17
18 // 圖的鄰接矩陣表示法
19 vector<vector<int>> graph(MAX_CITIES,
   vector<int>(MAX_CITIES, INF));
20
21 //
   Dijkstra演算法，計算從指定城市出發到其他城市的最短路徑
22 void dijkstra(int startCity) {
23     vector<int> dist(numCities, INF);
24     vector<bool> visited(numCities, false);
25
26     dist[startCity] = 0;
27
28     for (int i = 0; i < numCities - 1; i++) {
29         int u = -1;
30         for (int j = 0; j < numCities; j++) {
31             if (!visited[j] && (u == -1 || dist[j] <
   dist[u])) {
32                 u = j;
33             }
34         }
35
36         visited[u] = true;
37
38         for (int v = 0; v < numCities; v++) {
39             if (!visited[v] && graph[u][v] != INF) {
40                 dist[v] = min(dist[v], dist[u] +
   graph[u][v]);
41             }
42         }
43     }
44
45     // 輸出最短路徑結果
46     cout << "從城市 " << startCity << "
   出發到其他城市的最短路徑如下：" << endl;
47     for (int i = 0; i < numCities; i++) {
48         if (i != startCity) {
49             cout << "到城市 " << i << " 的最短距離為
   " << dist[i] << endl;
50         }
51     }
52 }
53
54 int main() {
55     // 讀取城市數量和道路數量
56     cin >> numCities >> numRoads;
57
58     // 初始化圖的鄰接矩陣
59     for (int i = 0; i < numRoads; i++) {
60         int city1, city2, distance;
61         cin >> city1 >> city2 >> distance;
62         graph[city1][city2] = distance;
63         graph[city2][city1] = distance; //
   因為是雙向道路
64     }
65
66     // 選擇起始城市，這裡以城市0為例
67     int startCity = 0;
68

```

```

69 // 執行Dijkstra演算法
70 dijkstra(startCity);
71
72 return 0;
73 }

```

4.2 DFS

```

1 // 印出最快路徑（座標）
2 #include <bits/stdc++.h>
3 #define N 100
4 using namespace std;
5
6 int map[N][N], visited[N][N]={0};
7 typedef pair<int, int> p;
8 int n,m,found=0;
9 deque<p> path;
10
11 void dfs(int x, int y){
12     if (found==1) return;
13     visited[x][y]=1;
14     path.push_back(make_pair(x,y));
15     if (x==n-1 && y==m-1){
16         found=1;
17         cout<<"Path: ";
18         while(!path.empty()){
19             cout<< "("<<path.front().first<<","<<path.front().second<<
20             path.pop_front();
21             cout<<((path.empty())?"\n":"->");
22         }
23         cout<<endl;
24         return;
25     }
26     if (x+1<n && visited[x+1][y]==0 && map[x+1][y]==0){
27         dfs(x+1,y);
28         path.pop_back();
29     }
30     if (y+1<m && visited[x][y+1]==0 && map[x][y+1]==0){
31         dfs(x,y+1);
32         path.pop_back();
33     }
34     if (x-1>=0 && visited[x-1][y]==0 && map[x-1][y]==0){
35         dfs(x-1,y);
36         path.pop_back();
37     }
38     if (y-1>=0 && visited[x][y-1]==0 && map[x][y-1]==0){
39         dfs(x,y-1);
40         path.pop_back();
41     }
42 }
43
44 int main(){
45     cin>>n>>m;
46     for (int i=0; i<n; i++)
47         for (int j=0; j<m; j++)
48             cin>>map[i][j];
49     dfs(0,0);
50     if (found==0){
51         cout<<"No routes accessible.\n";
52     }
53     return 0;
54 }
55
56 // 顯示最短距離
57 #include <iostream>
58 #include <utility>
59 #include <deque>
60 #define N 100
61 using namespace std;
62
63 int map[N][N], visited[N][N]={0};
64 typedef pair<int, int> p;
65 int n,m,dis=-2;
66 deque<p> path;
67
68 void dfs(int x, int y){

```

```

68 visited[x][y]=1;
69 path.push_back(make_pair(x,y));
70 if (x==n-1 && y==m-1){
71     if (dis==1){
72         dis=path.size()-1;
73     }
74     else {
75         if (path.size()-1<dis) dis=path.size()-1;
76     }
77 }
78 if (x+1<n && visited[x+1][y]==0 && map[x+1][y]==0){
79     dfs(x+1,y);
80     visited[x+1][y]=0;
81     path.pop_back();
82 }
83 if (y+1<m && visited[x][y+1]==0 && map[x][y+1]==0){
84     dfs(x,y+1);
85     visited[x][y+1]=0;
86     path.pop_back();
87 }
88 if (x-1>=0 && visited[x-1][y]==0 && map[x-1][y]==0){
89     dfs(x-1,y);
90     visited[x-1][y]=0;
91     path.pop_back();
92 }
93 if (y-1>=0 && visited[x][y-1]==0 && map[x][y-1]==0){
94     dfs(x,y-1);
95     visited[x][y-1]=0;
96     path.pop_back();
97 }
98 }
99
100 int main(){
101     cin>>n>>m;
102     for (int i=0; i<n; i++)
103         for (int j=0; j<m; j++)
104             cin>>map[i][j];
105     dfs(0,0);
106     if (dis==1)
107         cout<<"No routes accessible.\n";
108     else
109         cout<<"Shortest distance: "<<dis<<endl;
110     return 0;
111 }

```

4.3 merge sort

```

1 #include <iostream>
2 using namespace std;
3
4 //做比較大小的部分
5 void merge(int arr[], int l, int m, int r, int size)
6 {
7     int i = l;
8     int j = m + 1;
9     int k = l;
10
11     /* create temp array */
12     int temp[size];
13
14     while (i <= m && j <= r) {
15         if (arr[i] <= arr[j]) {
16             temp[k] = arr[i];
17             i++;
18             k++;
19         }
20         else {
21             temp[k] = arr[j];
22             j++;
23             k++;
24         }
25     }
26     / Copy the remaining elements of first half, if
27     there are any /
28     while (i <= m) {

```

```

28         temp[k] = arr[i];
29         i++;
30         k++;
31     }
32
33     / Copy the remaining elements of second half, if
34     there are any /
35     while (j <= r) {
36         temp[k] = arr[j];
37         j++;
38         k++;
39     }
40
41     / Copy the temp array to original array /
42     for (int p = l; p <= r; p++) {
43         arr[p] = temp[p];
44     }
45
46     //做分開陣列的部分
47 void mergeSort(int arr[], int l, int r, int size)
48 {
49     if (l < r) {
50         // 找中間點 ex:陣列五個元素0-4 2是中間點
51         //陣列分成兩組 0-2/3-4兩個部分
52         //舉0-2陣列來說 中間點是1
53         //陣列再分成 0-1/2兩個部分
54         int m = (l + r) / 2;
55
56         / 遞迴第一和第二部分*/
57         //(也就是不斷的分)
58         mergeSort(arr, l, m, size);
59         mergeSort(arr, m + 1, r, size);
60
61         // merge
62         //當我分到不能再分 比較陣列內數值 小的放前面
63         merge(arr, l, m, r, size);
64     }
65 }
66
67 int main()
68 {
69     cout << "Enter size of array: " << endl;
70     int size;
71     cin >> size;
72     int myarray[size];
73
74     cout << "Enter " << size << " integers in any
75     order: " << endl;
76     for (int i = 0; i < size; i++) {
77         cin >> myarray[i];
78     }
79     cout << "Before Sorting" << endl;
80     for (int i = 0; i < size; i++) {
81         cout << myarray[i] << " ";
82     }
83     cout << endl;
84     mergeSort(myarray, 0, (size - 1), size); //
85     mergesort(arr,left,right) called
86
87     cout << "After Sorting" << endl;
88     for (int i = 0; i < size; i++) {
89         cout << myarray[i] << " ";
90     }
91     return 0;
92 }

```

4.4 quick sort

```

1 include <iostream>
2 using namespace std;
3 // quick sort sorting algorithm
4 int Partition(int arr[], int s, int e)
5 {

```

```

6  int pivot = arr[e];
7  int pIndex = s;
8
9  for(int i = s; i < e; i++)
10 {
11     if(arr[i] < pivot)
12     {
13         int temp = arr[i];
14         arr[i] = arr[pIndex];
15         arr[pIndex] = temp;
16         //swapping 也就是說如果當前數值比指標小
17         //他就移到最前面
18         //也就是陣列0的位置
19         pIndex++;
20         //下一個比指標小的數值放進陣列1的位置
21     }
22 }
23 int temp = arr[e];
24 arr[e] = arr[pIndex];
25 arr[pIndex] = temp;
26 //比指標數值小的都去前面了
27 //將指標放到目前計數到的陣列位置
28 //那指標前都比她小 指標後都比他大
29 return pIndex; //回傳給p值
30 }
31
32 void QuickSort(int arr[], int s, int e)
33 //s stand for start index
34 //e stand for end index also (size-1)
35 {
36     if(s < e)
37     {
38         int p = Partition(arr, s, e);
39         QuickSort(arr, s, (p-1));
40         // recursive QS call for left partition
41         //做陣列前半部分 因為都比指標小 去進行內部排序
42         QuickSort(arr, (p+1), e);
43         // recursive QS call for right partition
44     }
45 }
46
47 int main()
48 {
49
50     int size=0;
51     cout<<"Enter Size of array: "<<endl;
52     cin>>size;
53     int myarray[size];
54
55     cout<<"Enter "<<size<<" integers in any order:
56         "<<endl;
57     for(int i=0; i<size; i++)
58     {
59         cin>>myarray[i];
60     }
61     cout<<"Before Sorting"<<endl;
62     for(int i=0; i<size; i++)
63     {
64         cout<<myarray[i]<<" ";
65     }
66     cout<<endl;
67
68     QuickSort(myarray, 0, (size-1)); // quick sort called
69
70     cout<<"After Sorting"<<endl;
71     for(int i=0; i<size; i++)
72     {
73         cout<<myarray[i]<<" ";
74     }
75
76     return 0;
77 }

```

4.5 二分搜

```

1  #include <iostream>
2  using namespace std;
3
4  int binarySearch(int arr[], int left, int right, int
5      x) {
6      while (left <= right) {
7          int mid = left + (right - left) / 2;
8
9          if (arr[mid] == x) {
10             return mid;
11         }
12         else if (arr[mid] < x) {
13             left = mid + 1;
14         }
15         else {
16             right = mid - 1;
17         }
18     }
19     return -1;
20 }
21
22 int main() {
23     int myarr[10];
24     int num;
25     int output;
26
27     cout << "Please enter 10 elements ASCENDING order"
28         << endl;
29     for (int i = 0; i < 10; i++) {
30         cin >> myarr[i];
31     }
32     cout << "Please enter an element to search" << endl;
33     cin >> num;
34
35     output = binarySearch(myarr, 0, 9, num);
36
37     if (output == -1) {
38         cout << "No Match Found" << endl;
39     } else {
40         cout << "Match found at position: " << output <<
41             endl;
42     }
43
44     return 0;
45 }
46
47 如果我們超過25頁我還可以再縮減程式區 這是比較完整的
48 Floyd
49 void floyd(){
50     for(int k=0; k<n; k++){ //中間點
51         for(int i=0; i<n; i++){
52             for(int j=0; j<n; j++){
53                 dp[i][j]=min(dp[i][j], dp[i][k]+dp[k][j]);
54                 //經過中間點k的路徑是否小於原始路徑
55                 //小於則更新 不小於則不變動
56                 //窮舉所有鬆弛的可能
57             }
58         }
59     }
60 }

```

5 dp

5.1 階乘 1

```

1  '''
2  !注意! long long 存到21!就會爆掉
3
4  1. 要你輸出階乘
5  好懶，請你直接用python
6  '''

```

```

7 | a = 0
8 | while True:
9 |     try:
10 |         a = int(input())
11 |         sum = 1
12 |         for i in range(1, a+1):
13 |             sum *= i
14 |         a = str(a)
15 |         print(a + '!')
16 |         print(sum)
17 |     except EOFError:
18 |         break

```

5.2 階乘 2

```

1 | /*
2 | 2. 要你輸出階乘最後一個非0的數字
3 | 用dp表格先存1-10000數字的階乘，
4 | 同時因為我們只關心最後一個非0的數字，
5 | 所以可以每次乘完一階就讓他進while迴圈裡%10，
6 | 把0都去掉，到while迴圈外後再把arr[i]%10000，
7 | 只留下剩下可能會影響結果的數值部分。
8 | */
9 | typedef long long ll;
10 | ll arr[10000];
11 | void s(){
12 |     arr[0]=1;
13 |     for(ll i = 1; i <= 10000; i++){
14 |         arr[i] = arr[i-1]*(i+1);
15 |         while (arr[i] % 10 == 0) {
16 |             arr[i] /= 10;
17 |         }
18 |         arr[i] %= 1000000;
19 |     }
20 | }

```

5.3 階梯問題

```

1 | /*
2 | 1. 問從左上角走到右下角有幾種解法
3 | - 此問題可分為(1)往下(2)往右，兩個走法。
4 | */
5 | const int H = 8, W = 8;
6 | int f[2][W]; //
7 |     兩條陣列，儲存最近算出來的問題答案。
8 | void staircase_walk()
9 | {
10 |     // [Initial States]
11 |     for (int j=0; j<W; ++j) f[0][j] = 1;
12 |
13 |     // [Computation]
14 |     for (int i=1; i<H; i++)
15 |         for (int j=1; j<W; j++)
16 |             // 只是多了 mod 2，
17 |             // 外觀看起來就像兩條陣列輪替使用。
18 |             f[i % 2][j] = f[(i-1) % 2][j] + f[i %
19 |                 2][j-1];
20 |
21 |     // 輸出結果
22 |     cout << "由(0,0)走到(7,7)有" << f[7 % 2][7] <<
23 |         "種走法";
24 |     // cout << "由(0,0)走到(7,7)有" << f[(H-1) %
25 |         2][W-1] << "種走法";
26 | }

```

5.4 極值問題（格子有權重）

```

1 | const int H = 8, W = 8;
2 | int a[H][W];

```

```

3 | int f[H][W];
4 |
5 | void staircase_walk()
6 | {
7 |     // [Initial States]
8 |     f[0][0] = a[0][0];
9 |     for (int i=1; i<H; i++)
10 |         f[i][0] = f[i-1][0] + a[i][0];
11 |     for (int j=1; j<W; j++)
12 |         f[0][j] = f[0][j-1] + a[0][j];
13 |
14 |     // [Computation]
15 |     for (int i=1; i<H; i++)
16 |         for (int j=1; j<W; j++)
17 |             f[i][j] = max(f[i-1][j], f[i][j-1]) + a[i][j];
18 |
19 |     // 輸出結果
20 |     cout << "由(0,0)走到(7,7)的最小總和" << f[7][7];
21 |     // cout << "由(0,0)走到(7,7)的最小總和" <<
22 |         f[H-1][W-1];
23 |
24 |     int h, w;
25 |     while (cin >> h >> w)
26 |         cout << "由(0,0)走到(h,w)的最小總和" << f[h][w];

```