Contents

1 基本

1.1 型態大小

- **int**:
  - $-2,147,483,648$ to $2,147,483,647$ (10 digits)
  - -2^{31} to $2^{31}-1$
  - -10^{9} to 10^{9}
- **unsigned long long int**:
  - Begins with 9, and has a total of 19 digits
  - $2^{63}-1$
  - 10^{18}
- **array**:
  - Do not declare with a size larger than 30,005.

2 語法

2.1 c++

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef unsigned long long int ll;
int main(){
    std::ios::sync_with_stdio(false);
    return 0;
}
```

2.2 c++ 函式庫

```cpp
// <string>

// 查找substr第一次出現的位置
str.find(substr);
// 返回substr最后一次出现的位置
str.rfind(substr);

s1 = s1+s2 // 連接兩個字串
s1 + s2 // 跟上面一樣
s1.append(s2) // s2插在s1的屁股
if(s1 == s2) // 比較兩字串

// <ctype>

// 檢查系列
isalpha(c) // 字母
isdigit(c) // 數字
isalnum(c) // 字母or數字
isspace(c) // 空格or換行
isupper(c) // 大寫
islower(c) // 小寫
ispunct(c) // 標點符號
toupper(c) // to大寫
tolower(c) // to小寫


// <algorithm>

// 酷東西
reverse(v, v+n)
find(v, v+n, 3) //查找3是否在v中
count(v, v+n, 3) // 算3在v裡出現幾次(只能算字元or數字)

// sort
sort(v.begin(), v.end())
sort(v, v+n)
sort(v, v+n, greater<int>())

sort(v, v+n, cmp)
bool cmp(型態 a, 型態 b){
    return a > b; // 大到小
}

// <numeric>

// 返回鄰近數值的差
int arr[10]={1,2,3,4,5,6,7,8,9,10};
int a[9] = {0};
adjacent_difference(arr, arr+10, a);
for(int i= 0; i < 9; i++){
  cout<<a[i]<<' ';
}

// <cmath>
round(x) // 返回最接近x的整數
```

2.3 宣告法

```cpp
// <vector>
vector<int> v;
vector<int> v = {1,2,3,4};
vector<int> v(5); // v={0,0,0,0,0}
vector<int> v(5,1); // v={1,1,1,1,1}
vector<vector<int>> v; //二維
// v[2][3] v[樓][層]
vector<vector<int>> v(2, vector<int>(3));
// v = {(1,1),(1,1),(1,1)}
vector<vector<int>> v(2, vector<int>(3, 1));

v.push_back(1) // 推入數字
```

```cpp
v.pop_back() // 拔出尾端數字

// 在二維陣列中插入元素
vector<vector<int> > arr(5, vector<int>(3, 1));
arr[1].push_back(2);
for(size_t i= 0; i < arr.size();i++){
  for(size_t j= 0; j < arr[i].size();j++){
    cout<<arr[i][j]<<' ';
  }
  cout<<endl;
}
/*
Output
1 1 1
1 1 1 2
1 1 1
1 1 1
1 1 1
*/

// struct
struct s{
  int x[100];
  int y[100];
};
s num; //一 組s
num.x[1]=1; num.y[1]=2;

// set
set<int> s;

s.insert(x)
s.count(x) // x是否存在於set中
s.erase(x)

s.clear()
s.empty()
s.size()

// stack
stack<int> s;

s.push(1); // 把1推到尾巴
s.pop(); // 刪掉尾巴
s.top(); // 返回尾巴

// queue
queue<int> q;
q.pop(); // 刪掉前
q.front(); // 返回前
q.back(); // 返回尾
q.push(1); // 把1推到前
```

2.4 強制轉型

```cpp
// 數字to字元(串)
str = to_string(num)
c = num + '0';

// 字符串流轉型法
stringstream ss;
ss << num; // 把num塞進字符串流
string str;
ss >> str; // 把字符串留丟進str

// 字串to數字
int num = stoi(str) //整數
double num = stod(str) //小數
```

2.5 python

```python
# EOF
while True:
    try:
        '''
        你要執行的程式碼
        '''
    except EOFError:
        break

# 有規定終止條件
while True:
    if a==0:
        break

# 數學符號
a//=10 # 整除
a**b   # a^b

# 邏輯
a=True
b=False
print(a and b) #False
print(a or b)  #True

# scan
a = int(input())
n=list(input().split(' '))           #
    連續輸入一串用空格隔開的數字

for i in range(a):
    c, d = map(int, input().split()) # 連續輸入兩個數

# print
print('for is not a multiple of 11.'.format(a))
print(a+" and "+b+" sitting in the tree")
print('The parity of ',a,' is ',count,' (mod 2).')

# 標頭檔math
import math
math.gcd(a, b, c, d, e) # 最大公約數
math.lcm(a, b, c, d, e) # 最小公倍數
math.fabs(x)   # 絕對值
math.isqrt(n)  # 整數平方根
math.sqrt(x)   # 平方根
math.pow(x, y) # x^y

# count
c+=b.count("商店")  # 用在要計算好幾個字串時
c=b.count('1')     # 一次算出一串字串有幾個 '1'

# 進制轉換
a = bin(a)[2:] # 10 to 2
a = hex(a)[2:] # 10 to 16
a = oct(a)[2:] # 10 to 8

# 大小寫轉換
a.lower()
a.upper()

# 取長度
a.len()
```

3 字串

3.1 KMP

```cpp
#include <iostream>
using namespace std;

void KMP(string text, string pattern)
{
    int m = text.length();
```

```
    int n = pattern.length();

    // 如果模組沒東東
    if (n == 0)
    {
        cout << "The pattern occurs with shift 0";
        return;
    }

    // 如果文本的長度小於模組的長度
    if (m < n)
    {
        cout << "Pattern not found";
        return;
    }

    // next[i] 存儲下一個最佳部分匹配的索引
    int next[n + 1];

    for (int i = 0; i < n + 1; i++) {
        next[i] = 0;
    }

    for (int i = 1; i < n; i++)
    {
        int j = next[i];

        while (j > 0 && pattern[j] != pattern[i]) {
            j = next[j];
        }

        if (j > 0 || pattern[j] == pattern[i]) {
            next[i + 1] = j + 1;
        }
    }

    for (int i = 0, j = 0; i < m; i++)
    {
        if (text[i] ==
            pattern[j])//一樣如果+1j下一個檢查
        {
            if (++j == n) {
                cout << "The pattern occurs with
                    shift " << i - j + 1 << endl;
            }
        }
        else if (j > 0)
        {
            j = next[j];//把她休崩變回來
            i--;     // 還要回去啾啾
        }
    }
}

int main()
{
    string text = "ABCABAABCABAC";
    string pattern = "CAB";

    KMP(text, pattern);

    return 0;
}
```

4 數論

4.1 快速幕

```
long long binpow(long long a, long long b){
    if(b==0) return 1;
    int res = binpow(a, b/2);
    if(b%2==0) return res*res;
    else return res*res*a;
}
```

4.2 窮舉（選 or 不選）

```
 #include<bits/stdc++.h>
using namespace std;

int k, all = 0;
int Min = 9999999;
int arr[100] = {0};

void Find(int sum, int now){
    if(now == k) return;
    Min = min(abs(all-sum-sum), Min);

    Find(sum, now+1);
    Find(sum+arr[now], now+1);
    return;
}
int main(){
    cin >> k;
    for(int i = 0; i < k; i++){
        cin >> arr[i];
        all+=arr[i];
    }
    Find(0, 0);
    cout << Min;
}
```

4.3 喵

```
#include <iostream>
using namespace std;
int a[20];

int main() {
    int cases, target, sticks, num, tmp, i;
    bool flag;
    while (cin >> cases){
        while (cases--){
            cin >> target;
            cin >> sticks;
            for (int i = 0; i < sticks; i++){
                cin >> a[i];
            }
            num = 1;
            for (int i = 1; i < sticks; i++){
                num <<= 1;
                num++;
            }
            flag = false;
            for (int _i = 0; _i <= num; _i++){
                tmp = 0;
                i = _i;
                for (int j = 0; j < sticks; j++){
                    if (i & 1) tmp += a[j];
                    i >>= 1;
                }
                if (tmp == target){
                    flag = true;
                    break;
                }
            }
            if (flag) cout << "YES\n";
            else cout << "NO\n";
        }
    }
}
```

4.4 小費馬實踐

```
ll s(ll b, ll p, ll m){
  if(p == 0) return 1;
  else if(p%2!=0) return ((b%m)*(s(b, p-1, m)%m))%m;
  else{
```

```
    ll t = s(b, p/2, m);
    return ((t%m)*(t%m))%m;
  }
}
```

4.5 Fibonaccimal

```
#include<bits/stdc++.h>
using namespace std;

int main(){

    int N;
    int Fibonacci[40] = {0, 1};//開始的兩個數
    int i;

    for(i = 2; i < 40; i++){
        Fibonacci[i] = Fibonacci[i - 1] + Fibonacci[i
            - 2];
    }

    scanf("%d", &N);

    while(N--){

        int num;
        bool flag = false;

        scanf("%d", &num);
        printf("%d = ", num);

        for(i = 39; i >= 2; i--){
            if(num >= Fibonacci[i]){
                num = num - Fibonacci[i];
                flag = true;
                printf("1");
            }
            else if(flag){
                printf("0");
            }
        }

        printf(" (fib)\n");
    }

    return 0;
}
```

4.6 LCM

```
int GCD(int num1,int num2)
{
  if(num2==0)
  {
   return num1;
  }

  return GCD(num2,num1%num2);
}

int LCM(int num1,int num2) //2個最小公倍數
{
  return((num1*num2)/GCD(num1,num2));
}

int LCM2(int num1,int num2,int num3) //3個最小公倍數
{
  return((num1*num2)/GCD((num1,num2),num3));
}

int main()
{
    cout<<GCD(6,3);
    cout<<LCM(6,3);
    cout<<LCM2(6,3,3);

    return 0;
}
```

4.7 LCS

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string str1, str2;
    short lcs[2][1000];
    while (cin >> str1 >> str2)
    {
        lcs[0][0] = str1[0] == str2[0];
        for (int j=1; j<str2.length(); j++) lcs[0][j]
            = max(lcs[0][j-1],
            short{str1[0]==str2[j]});
        for (int i=1; i<str1.length(); i++)
        {
            bool r = i & 1;
            lcs[r][0] = max(lcs[r^1][0],
                short{str1[i]==str2[0]});
            for (int j=1; j<str2.length(); j++)
                lcs[r][j] = str1[i]==str2[j] ?
                    lcs[r^1][j-1] + 1 :
                    max(lcs[r^1][j], lcs[r][j-1]);
        }
        cout <<
            lcs[(str1.length()-1)&1][str2.length()-1]
            << '\n';
    }
    return 0;
}
```

4.8 LPS

```
#include<bits/stdc++.h>
#include<iostream>
using namespace std;

int main(){
    string s;
    cin >> s;
    int maxlen=0, l, r;
    for(int i=0; i<s.length(); i++){
        //奇
        int x = 0;
        while((s[i-x]==s[i+x]) && (i-x>=0) &&
            (i+x<s.length())){//當找到一個中心點以其為中間然後左
            x++;
        }
        x--;
        if(2*x+1 >
            maxlen){//只有第一次會max==後後面就追不到那女孩
            maxlen = 2*x+1;//最大的
            l = i-x;//計算頭頭
            r = i+x;//計算尾巴
        }
        //偶
        x = 0;
        while( (s[i-x]==s[i+1+x]) && (i-x>=0) &&
            (i+1+x<s.length()) ){
            x++;
        }
        if(2*x > maxlen){
            maxlen = 2*x;
            l = i-x+1;
            r = i+x;
        }
    }
```

```
    cout << maxlen << '\n';
    cout << l+1 << ' ' << r+1 << '\n';

}
```

4.9 Pairty

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    int I, n;
    while (cin >> I) {
        if (I == 0) break;
        string B = "";
        n = I;
        int cnt = 0;
        while (n){
            cnt += (n & 1);
            B += '0' + (n & 1);
            n >>= 1;
        }
        reverse(B.begin(), B.end());
        cout << "The parity of " << B << " is " <<
            cnt << " (mod 2).\n";
    }
    return 0;
}
```

5 圖論

5.1 最短路徑 dijkstra

```
// 邊權重皆為正數時使用
// 1.
    輸入有總點、總邊，接著輸入點,點,距離（權重）時使用
#include <iostream>
#include <vector>
#include <climits>

using namespace std;

// 定義城市數量的上限
#define MAX_CITIES 100

// 定義無限大的距離
#define INF INT_MAX

// 城市數量、道路數量
int numCities, numRoads;

// 圖的鄰接矩陣表示法
vector<vector<int> > graph(MAX_CITIES,
    vector<int>(MAX_CITIES, INF));

//
    Dijkstra演算法，計算從指定城市出發到其他城市的最短路徑
void dijkstra(int startCity) {
    vector<int> dist(numCities, INF);
    vector<bool> visited(numCities, false);

    dist[startCity] = 0;

    for (int i = 0; i < numCities - 1; i++) {
        int u = -1;
        for (int j = 0; j < numCities; j++) {
            if (!visited[j] && (u == -1 || dist[j] <
                dist[u])) {
                u = j;
            }
        }

        visited[u] = true;

        for (int v = 0; v < numCities; v++) {
            if (!visited[v] && graph[u][v] != INF) {
                dist[v] = min(dist[v], dist[u] +
                    graph[u][v]);
            }
        }
    }

    // 輸出最短路徑結果
    cout << "從城市 " << startCity << "
        出發到其他城市的最短路徑如下：" << endl;
    for (int i = 0; i < numCities; i++) {
        if (i != startCity) {
            cout << "到城市 " << i << " 的最短距離為
                " << dist[i] << endl;
        }
    }
}

int main() {
    // 讀取城市數量和道路數量
    cin >> numCities >> numRoads;

    // 初始化圖的鄰接矩陣
    for (int i = 0; i < numRoads; i++) {
        int city1, city2, distance;
        cin >> city1 >> city2 >> distance;
        graph[city1][city2] = distance;
        graph[city2][city1] = distance; //
            因為是雙向道路
    }

    // 選擇起始城市，這裡以城市0為例
    int startCity = 0;

    // 執行Dijkstra演算法
    dijkstra(startCity);

    return 0;
}
```

5.2 DFS

```
// 印出最快路徑（座標）
#include<bits/stdc++.h>
#define N 100
using namespace std;

int map[N][N], visited[N][N]={0};
typedef pair<int, int> p;
int n,m,found=0;
deque<p> path;

void dfs(int x, int y){
  if (found==1) return;
  visited[x][y]=1;
  path.push_back(make_pair(x,y));
  if (x==n-1 && y==m-1){
    found=1;
    cout<<"Path: ";
    while(!path.empty()){
      cout<<"("<<path.front().first<<","<<path.front().second<<
      path.pop_front();
      cout<<((path.empty())?"\n":"→");
    }
    cout<<endl;
    return;
  }
  if (x+1<n && visited[x+1][y]==0 && map[x+1][y]==0){
    dfs(x+1,y);
```

```
    path.pop_back();
  }
  if (y+1<m && visited[x][y+1]==0 && map[x][y+1]==0){
    dfs(x,y+1);
    path.pop_back();
  }
  if (x-1>=0 && visited[x-1][y]==0 && map[x-1][y]==0){
    dfs(x-1,y);
    path.pop_back();
  }
  if (y-1>=0 && visited[x][y-1]==0 && map[x][y-1]==0){
    dfs(x,y-1);
    path.pop_back();
  }
}

int main(){
  cin>>n>>m;
  for (int i=0; i<n; i++)
    for (int j=0; j<m; j++)
      cin>>map[i][j];
  dfs(0,0);
  if (found==0){
    cout<<"No routes accessible.\n";
  }
  return 0;
}
// 顯示最短距離
#include <iostream>
#include <utility>
#include <deque>
#define N 100
using namespace std;

int map[N][N], visited[N][N]={0};
typedef pair<int, int> p;
int n,m,dis=-2;
deque<p> path;

void dfs(int x, int y){
  visited[x][y]=1;
  path.push_back(make_pair(x,y));
  if (x==n-1 && y==m-1){
    if (dis==-1){
      dis=path.size()-1;
    }
    else {
      if (path.size()-1<dis) dis=path.size()-1;
    }
  }
  if (x+1<n && visited[x+1][y]==0 && map[x+1][y]==0){
    dfs(x+1,y);
    visited[x+1][y]=0;
    path.pop_back();
  }
  if (y+1<m && visited[x][y+1]==0 && map[x][y+1]==0){
    dfs(x,y+1);
    visited[x][y+1]=0;
    path.pop_back();
  }
  if (x-1>=0 && visited[x-1][y]==0 && map[x-1][y]==0){
    dfs(x-1,y);
    visited[x-1][y]=0;
    path.pop_back();
  }
  if (y-1>=0 && visited[x][y-1]==0 && map[x][y-1]==0){
    dfs(x,y-1);
    visited[x][y-1]=0;
    path.pop_back();
  }
}

int main(){
  cin>>n>>m;
  for (int i=0; i<n; i++)
    for (int j=0; j<m; j++)
      cin>>map[i][j];
  dfs(0,0);
  if (dis==-2)
    cout<<"No routes accessible.\n";
  else
    cout<<"Shortest distance: "<<dis<<endl;
  return 0;
}
```

5.3 merge sort

```
#include <iostream>
using namespace std;

//做比較大小的部分
void merge(int arr[], int l, int m, int r, int size)
{
    int i = l;
    int j = m + 1;
    int k = l;

    /* create temp array */
    int temp[size];

    while (i <= m && j <= r) {
        if (arr[i] <= arr[j]) {
            temp[k] = arr[i];
            i++;
            k++;
        }
        else {
            temp[k] = arr[j];
            j++;
            k++;
        }
    }
    / Copy the remaining elements of first half, if
        there are any /
    while (i <= m) {
        temp[k] = arr[i];
        i++;
        k++;
    }

    / Copy the remaining elements of second half, if
        there are any /
    while (j <= r) {
        temp[k] = arr[j];
        j++;
        k++;
    }

    / Copy the temp array to original array /
    for (int p = l; p <= r; p++) {
        arr[p] = temp[p];
    }
}

//做分開陣列的部分
void mergeSort(int arr[], int l, int r, int size)
{
    if (l < r) {
        // 找中間點 ex:陣列五個元素0-4 2是中間點
        //陣列分成兩組 0-2/3-4兩個部分
        //舉0-2陣列來說 中間點是1
        //陣列再分成 0-1/2兩個部分
        int m = (l + r) / 2;

        / 遞迴第一和第二部分*/
        //(也就是不斷的分)
        mergeSort(arr, l, m, size);
        mergeSort(arr, m + 1, r, size);

        // merge
        //當我分到不能再分 比較陣列內數值 小的放前面
        merge(arr, l, m, r, size);
```

```
    }
}

int main()
{
    cout << "Enter size of array: " << endl;
    int size;
    cin >> size;
    int myarray[size];

    cout << "Enter " << size << " integers in any
        order: " << endl;
    for (int i = 0; i < size; i++) {
        cin >> myarray[i];
    }
    cout << "Before Sorting" << endl;
    for (int i = 0; i < size; i++) {
        cout << myarray[i] << " ";
    }
    cout << endl;
    mergeSort(myarray, 0, (size - 1), size); //
        mergesort(arr,left,right) called

    cout << "After Sorting" << endl;
    for (int i = 0; i < size; i++) {
        cout << myarray[i] << " ";
    }

    return 0;
}
```

5.4 quick sort

```
include <iostream>
using namespace std;
// quick sort sorting algorithm
int Partition(int arr[], int s, int e)
{
 int pivot = arr[e];
 int pIndex = s;

 for(int i = s;i<e;i++)
 {
     if(arr[i]<pivot)
     {
     int temp = arr[i];
     arr[i] = arr[pIndex];
     arr[pIndex] = temp;
    //swapping 也就是說如果當前數值比指標小
        他就移到最前面
    //也就是陣列0的位置
     pIndex++;
    //下一個比指標小的數值放進陣列1的位置
     }
 }

 int temp = arr[e];
 arr[e] = arr[pIndex];
 arr[pIndex] = temp;
 //比指標數值小的都去前面了
 //將指標放到目前計數到的陣列位置
 //那指標前都比她小 指標後都比他大
 return pIndex; //回傳給p值
}

void QuickSort(int arr[], int s, int e)
//s stand for start index
//e stand for end index also (size-1)
{
 if(s<e)
 {
 int p = Partition(arr,s, e);
 QuickSort(arr, s, (p-1));
// recursive QS call for left partition
//做陣列前半部分 因為都比指標小 去進行內部排序
 QuickSort(arr, (p+1), e);
// recursive QS call for right partition
 }
}

int main()
{

 int size=0;
 cout<<"Enter Size of array: "<<endl;
 cin>>size;
 int myarray[size];

 cout<<"Enter "<<size<<" integers in any order:
     "<<endl;
 for(int i=0;i<size;i++)
 {
 cin>>myarray[i];
 }
 cout<<"Before Sorting"<<endl;
 for(int i=0;i<size;i++)
 {
 cout<<myarray[i]<<" ";
 }
 cout<<endl;

 QuickSort(myarray,0,(size-1));  // quick sort called

 cout<<"After Sorting"<<endl;
 for(int i=0;i<size;i++)
 {
 cout<<myarray[i]<<" ";
 }

 return 0;
}
```

5.5 二分搜

```
#include < iostream >
using namespace std;

int binarySearch(int arr[], int left, int right, int
    x) {
  while (left <= right) {
    int mid = left + (right - left) / 2;

    if (arr[mid] == x) {
      return mid;
    }
     else if (arr[mid] < x) {
      left = mid + 1;
    }
     else {
      right = mid - 1;
    }
  }

  return -1;
}

int main() {
  int myarr[10];
  int num;
  int output;

  cout << "Please enter 10 elements ASCENDING order"
      << endl;
  for (int i = 0; i < 10; i++) {
    cin >> myarr[i];
  }
  cout << "Please enter an element to search" << endl;
  cin >> num;

  output = binarySearch(myarr, 0, 9, num);

```

```
  if (output == -1) {
    cout << "No Match Found" << endl;
  } else {
    cout << "Match found at position: " << output <<
        endl;
  }

  return 0;
}
```

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Floyd

```
void floyd(){
        for(intk=0;k<n;k++){ //中間點
            for(int i=0;i<n;i++){
                for(int j=0;j<n;j++){
                    dp[i][j]=min(dp[i][j],dp[i][k]+dp[k][j
                    //經過中間點k的路徑是否小於原始路徑
                    //小於則更新 不小於則不變動
                    //窮舉所有鬆弛的可能
                }
            }
        }
    }
```

6 dp

6.1 階乘 1

```
'''
!注意！long long 存到21!就會爆掉

1.  要你輸出階乘
好懶，請你直接用python
'''
a = 0
while True:
    try:
        a = int(input())
        sum = 1
        for i in range(1,a+1):
            sum*=i
        a = str(a)
        print(a+'!')
        print(sum)
    except EOFError:
        break
```

6.2 階乘 2

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

6.3 階梯問題

```
/*
1.  問從左上角走到右下角有幾種解法
- 此問題可分為(1)往下(2)往右，兩個走法。
*/
const int H = 8, W = 8;
int f[2][W];    //
    兩條陣列，儲存最近算出來的問題答案。

void staircase_walk()
{     // [Initial States]
    for (int j=0; j<W; ++j) f[0][j] = 1;

    // [Computation]
    for (int i=1; i<H; i++)
       for (int j=1; j<W; j++)
           // 只是多了 mod 2，
          // 外觀看起來就像兩條陣列輪替使用。
         f[i % 2][j] = f[(i-1) % 2][j] + f[i %
             2][j-1];

      // 輸出結果
      cout << "由(0,0)走到(7,7)有" << f[7 % 2][7] <<
          "種走法";
  //  cout << "由(0,0)走到(7,7)有" << f[(H-1) %
      2][W-1] << "種走法";
}
```

6.4 極值問題（格子有權重）

```
const int H = 8, W = 8;
int a[H][W];
int f[H][W];

void staircase_walk()
{
  // [Initial States]
  f[0][0] = a[0][0];
  for (int i=1; i<H; i++)
    f[i][0] = f[i-1][0] + a[i][0];
  for (int j=1; j<W; j++)
    f[0][j] = f[0][j-1] + a[0][j];

  // [Computation]
  for (int i=1; i<H; i++)
    for (int j=1; j<W; j++)
      f[i][j] = max(f[i-1][j], f[i][j-1]) + a[i][j];

  // 輸出結果
  cout << "由(0,0)走到(7,7)的最小總和" << f[7][7];
//  cout << "由(0,0)走到(7,7)的最小總和" <<
    f[H-1][W-1];

  int h, w;
  while (cin >> h >> w)
    cout << "由(0,0)走到(h,w)的最小總和" << f[h][w];
}
```

7 數學

7.1 理論

- **三角形邊長定理**
  - $a+b>c$
  - 三角形形狀判定：
  - 直角$a^2+b^2=c^2$
  - 銳角$a^2+b^2>c^2$
  - 鈍角$a^2+b^2<c^2$

7.2 公式

- **積**
  - $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
  - $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
  - $\sum_{i=1}^{n} i^3 = \frac{(n^2(n+1)^2)}{4}$
  - $\sum_{i=1}^{n} i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$
  - $\sum_{i=1}^{n} i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$
  - $\sum_{i=1}^{n} i^6 = \frac{n(n+1)(2n+1)(3n^4+6n^3-3n+1)}{42}$
  - $\sum_{k=1}^{n} (k-1)(k-1)! = n! - 1$
- **log**
  - $\log \frac{a}{b} = \log a - \log b$
  - $\log_a b = \frac{\log a}{\log b}$
  - $\log ab = \log a + \log b$
  - $\log_a 1 = 0$
- **三角形面積**
  - $\sqrt{s(s-a)(s-b)(s-c)}$, $s = \frac{a+b+c}{2}$
- **圓形面積**
  - 面積：
  - $\pi * r^2$
  - 周長：
  - $2\pi * r$
- **圓形方程**
  - $(x-h)^2 + (y-k)^2 = r^2$, $(h,k) = \texttt{point}$
  - $x^2 + y^2 + Dx + Ey + F = 0$
  - $\texttt{point} = \left(-\frac{D}{2}, -\frac{E}{2}\right)$
  - $r = \sqrt{\left(\frac{D^2}{4} + \frac{E^2}{4} - F\right)}$
- **座標幾何**
  - 兩點距離：
  - $\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$
  - 斜率：
  - $\frac{y_2-y_1}{x_2-x_1}$
  - 中點：
  - $(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$
  - 平行：
  - $m_1 = m_2$
  - 垂直：
  - $m_1 * m_2 = -1$
- **二元方程式**
  - $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$
  - 判別式：
  - $x = b^2 - 4ac$
  - `x>0`: 相異根, `x=0`: 重根, `x<0`: 無實根
  - 根和：
  - $r_1 + r_2 = -\frac{b}{a}$
  - 根積：
  - $r_1 * r_2 = \frac{c}{a}$
  - 頂點：
  - $(-\frac{b}{2a}, -\frac{b^2-4ac}{4a})$
- **等差級數**
  - $S_n = \frac{(a_1+a_n)n}{2} = a_1 n + \frac{(n-1)nd}{2}$
  - $a_n = a_1 + (n-1)d$
  - $\sum_{i=1}^{n} a_n = \frac{(2a_1+(n-1)d)n}{2}$
- **等比級數**
  - $S_n = \frac{a_1(1-r^n)}{1-r} = \frac{a_1(r^n-1)}{r-1}, (r \neq 1)$
  - $\frac{a_n}{a_{n-1}} = r$
- **恆等式**
  - $a^3 + b^3 = (a+b)(a^2 - ab + b^2)$
  - $a^3 - b^3 = (a-b)(a^2 + ab + b^2)$
  - $(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$
  - $(a+b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$
  - $(a+b)^5 = a^5 + 5a^4b + 10a^3b^2 + 10a^2b^3 + 5ab^4 + b^5$
  - `1`
  - `1 1`
  - `1 2 1`
  - `1 3 3 1`
  - `1 4 6 4 1`
  - `1 5 10 10 5 1`
  - `1 6 15 20 15 6 1`
  - `1 7 21 35 35 21 7 1`
  - `1 8 28 56 70 56 28 8 1`
  - `1 9 36 84 126 126 84 36 9 1`
- **離散**
  - $(1+x)^n = \binom{n}{0} + \binom{n}{1}x^1 + \binom{n}{2}x^2 + ... + \binom{n}{n}x^n$
  - $\frac{1}{(1+x)^n} = \sum_{r=0}^{\infty} \binom{n+r-1}{r}(-1)^r(x)^r$
  - $\frac{1}{(1+x)^n} = \sum_{r=0}^{\infty} \binom{n+r-1}{r}(x)^r$
  - $\frac{1}{1-x} = 1 + x + x^2 + x^3 + ...$
  - $\frac{1}{(1-x)^2} = 1 + 2x + 3x^3 + 4x^4 + ... + nx^{n-1}$
- **線代**
  - $A_x = [a_1 a_2 ... a_n][\binom{x_1}{x_n}] = x_1 * a_1 + x_2 * a_2 + x_n * a_n$
  - $u = [], v = [], A(u+v) = A_u + A_v$
- **指數**
  - $a^n a^m = a^{n+m}$
  - $\frac{a^n}{a^m} = a^{n-m}$
  - $(a^n)^m = a^{n*m}$
- **小費馬**
  - $(a+b)\%n = (a\%n + b\%n)\%n$
  - $(a*b)\%n = (a\%n * b\%n)\%n$
  - $a * (b\%m) = (a*b)\%m$