

Contents

1	基本	1
1.1	型態大小	1
2	語法	1
2.1	c++	1
2.2	c++ 函式庫	1
2.3	宣告法	2
2.4	強制轉型	2
2.5	python	2
3	字串	3
3.1	KMP	3
4	數論	3
4.1	喵	3
4.2	Fibonacci	4
4.3	LCM	4
4.4	LCS	4
4.5	LPS	4
4.6	Pairty	5
5	圖論	5
5.1	最短路徑 dijkstra	5
5.2	DFS	5
5.3	merge sort	6
5.4	quick sort	7
5.5	二分搜	7
6	dp	8
6.1	階乘 1	8
6.2	階乘 2	8
6.3	階梯問題	8
6.4	極值問題 (格子有權重)	8

1 基本

1.1 型態大小

```

1 \documentclass{article}
2 \usepackage{amsmath}
3 \usepackage{geometry}
4
5 % Page layout
6 \geometry{a4paper, margin=1in}
7
8 \begin{document}
9 \title{Integer and Array Size Information}
10 \author{}
11 \date{}
12 \maketitle
13
14 \section*{Information}
15 \begin{itemize}
16 \item \texttt{int}:
17 \begin{itemize}
18 \item Range:  $-2,147,483,648$  to  $2,147,483,647$  (10 digits)
19 \item Using powers of 2:  $-2^{31}$  to  $2^{31}-1$ 
20 \item Using powers of 10:  $-10^9$  to  $10^9$ 
21 \end{itemize}
22
23 \item \texttt{unsigned long long int}:
24 \begin{itemize}
25 \item Begins with 9, and has a total of 19 digits
26 \item Using powers of 2:  $2^{63}-1$ 
27 \item Using powers of 10:  $10^{18}$ 
28 \end{itemize}
29
30 \item \texttt{array}:
31 \begin{itemize}
32 \item Do not declare with a size larger than 30,005.
33 \end{itemize}
34 \end{itemize}
35

```

36 \end{document}

2 語法

2.1 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

```

2.2 c++ 函式庫

```

1 // <string>
2
3 // 查找substr第一次出現的位置
4 str.find(substr);
5 // 返回substr最后一次出现的位置
6 str.rfind(substr);
7
8 s1 = s1+s2 // 連接兩個字串
9 s1 + s2 // 跟上面一樣
10 s1.append(s2) // s2插在s1的屁股
11 if(s1 == s2) // 比較兩字串
12
13 // <ctype>
14
15 // 檢查系列

```

```

16 isalpha(c) // 字母
17 isdigit(c) // 數字
18 isalnum(c) // 字母or數字
19 isspace(c) // 空格or換行
20 isupper(c) // 大寫
21 islower(c) // 小寫
22 ispunct(c) // 標點符號
23 toupper(c) // to大寫
24 tolower(c) // to小寫
25
26
27 // <algorithm>
28
29 // 酷東西
30 reverse(v, v+n)
31 find(v, v+n, 3) //查找3是否在v中
32 count(v, v+n, 3) // 算3在v裡出現幾次(只能算字元or數字)
33
34 // sort
35 sort(v.begin(), v.end())
36 sort(v, v+n)
37 sort(v, v+n, greater<int>())
38
39 sort(v, v+n, cmp)
40 bool cmp(型態 a, 型態 b){
41     return a > b; // 大到小
42 }
43
44 // <numeric>
45
46 // 返回鄰近數值的差
47 int arr[10]={1,2,3,4,5,6,7,8,9,10};
48 int a[9] = {0};
49 adjacent_difference(arr, arr+10, a);
50 for(int i= 0; i < 9; i++){
51     cout<<a[i]<< ' ';
52 }
53
54 // <cmath>
55 round(x) // 返回最接近x的整數

```

2.3 宣告法

```

1 // <vector>
2 vector<int> v;
3 vector<int> v = {1,2,3,4};
4 vector<int> v(5); // v={0,0,0,0,0}
5 vector<int> v(5,1); // v={1,1,1,1,1}
6 vector<vector<int>> v; //二維
7 // v[2][3] v[樓][層]
8 vector<vector<int>> v(2, vector<int>(3));
9 // v = {(1,1),(1,1),(1,1)}
10 vector<vector<int>> v(2, vector<int>(3, 1));
11
12 v.push_back(1) // 推入數字
13 v.pop_back() // 拔出尾端數字
14
15 // 在二維陣列中插入元素
16 vector<vector<int>> arr(5, vector<int>(3, 1));
17 arr[1].push_back(2);
18 for(size_t i= 0; i < arr.size();i++){
19     for(size_t j= 0; j < arr[i].size();j++){
20         cout<<arr[i][j]<< ' ';
21     }
22     cout<<endl;
23 }
24 /*
25 Output
26 1 1 1
27 1 1 1 2
28 1 1 1
29 1 1 1
30 1 1 1

```

```

31 */
32
33 // struct
34 struct s{
35     int x[100];
36     int y[100];
37 };
38 s num; //一組s
39 num.x[1]=1; num.y[1]=2;
40
41 // set
42 set<int> s;
43
44 s.insert(x)
45 s.count(x) // x是否存在於set中
46 s.erase(x)
47
48 s.clear()
49 s.empty()
50 s.size()
51
52 // stack
53 stack<int> s;
54
55 s.push(1); // 把1推到尾巴
56 s.pop(); // 刪掉尾巴
57 s.top(); // 返回尾巴
58
59 // queue
60 queue<int> q;
61 q.pop(); // 刪掉前
62 q.front(); // 返回前
63 q.back(); // 返回尾
64 q.push(1); // 把1推到前

```

2.4 強制轉型

```

1 // 數字to字元(串)
2 str = to_string(num)
3 c = num + '0';
4
5 // 字符串流轉型法
6 stringstream ss;
7 ss << num; // 把num塞進字符串流
8 string str;
9 ss >> str; // 把字符串留丟進str
10
11 // 字串to數字
12 int num = stoi(str) //整數
13 double num = stod(str) //小數

```

2.5 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(' ')) #
    連續輸入一串用空格隔開的數字
28
29 for i in range(a):
30     c, d = map(int, input().split()) # 連續輸入兩個數
31
32 # print
33 print('for is not a multiple of 11.'.format(a))
34 print(a+" and "+b+" sitting in the tree")
35 print('The parity of ',a,' is ',count,' (mod 2).')
36
37 # 標頭檔math
38 import math
39 math.gcd(a, b, c, d, e) # 最大公約數
40 math.lcm(a, b, c, d, e) # 最小公倍數
41 math.fabs(x) # 絕對值
42 math.isqrt(n) # 整數平方根
43 math.sqrt(x) # 平方根
44 math.pow(x, y) # x*y
45
46 # count
47 c+=b.count("商店") # 用在要計算好幾個字串時
48 c=b.count('1') # 一次算出一串字串有幾個 '1'
49
50 # 進制轉換
51 a = bin(a)[2:] # 10 to 2
52 a = hex(a)[2:] # 10 to 16
53 a = oct(a)[2:] # 10 to 8
54
55 # 大小寫轉換
56 a.lower()
57 a.upper()
58
59 # 取長度
60 a.len()

```

```

25 for (int i = 0; i < n + 1; i++) {
26     next[i] = 0;
27 }
28
29 for (int i = 1; i < n; i++)
30 {
31     int j = next[i];
32
33     while (j > 0 && pattern[j] != pattern[i]) {
34         j = next[j];
35     }
36
37     if (j > 0 || pattern[j] == pattern[i]) {
38         next[i + 1] = j + 1;
39     }
40 }
41
42 for (int i = 0, j = 0; i < m; i++)
43 {
44     if (text[i] ==
45         pattern[j])//一樣如果+1j下一個檢查
46     {
47         if (++j == n) {
48             cout << "The pattern occurs with
49                 shift " << i - j + 1 << endl;
50         }
51     }
52     else if (j > 0)
53     {
54         j = next[j];//把她休崩變回來
55         i--;      // 還要回去啾啾
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 字符串

3.1 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];

```

4 數論

4.1 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 }

```

4.2 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
12        - 2];
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
35        printf(" (fib)\n");
36    }
37
38    return 0;
39 }

```

4.3 LCM

```

1 int GCD(int num1, int num2)
2 {
3     if(num2==0)
4     {
5         return num1;
6     }
7
8     return GCD(num2, num1%num2);
9 }
10
11 int LCM(int num1, int num2) //2個最小公倍數
12 {
13     return ((num1*num2)/GCD(num1, num2));
14 }

```

```

15
16 int LCM2(int num1, int num2, int num3) //3個最小公倍數
17 {
18     return ((num1*num2)/GCD((num1, num2), num3));
19 }
20
21 int main()
22 {
23     cout<<GCD(6,3);
24     cout<<LCM(6,3);
25     cout<<LCM2(6,3,3);
26
27     return 0;
28 }

```

4.4 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 }

```

4.5 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
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;

```

```

23     while( (s[i-x]==s[i+1+x]) && (i-x>=0) &&
24            (i+1+x<s.length())){
25         x++;
26     }
27     if(2*x > maxlen){
28         maxlen = 2*x;
29         l = i-x+1;
30         r = i+x;
31     }
32     cout << maxlen << '\n';
33     cout << l+1 << ' ' << r+1 << '\n';
34 }
35 }

```

4.6 Pairty

```

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 }

```

5 圖論

5.1 最短路徑 dijkstra

```

1 // 邊權重皆為正數時使用
2 // 1.
3 // 輸入有總點、總邊，接著輸入點、點、距離（權重）時使用
4 #include <iostream>
5 #include <vector>
6 #include <limits>
7
8 using namespace std;
9
10 // 定義城市數量的上限
11 #define MAX_CITIES 100
12
13 // 定義無限大的距離
14 #define INF INT_MAX
15
16 // 城市數量、道路數量
17 int numCities, numRoads;
18
19 // 圖的鄰接矩陣表示法
20 vector<vector<int>> > graph(MAX_CITIES,
21                             vector<int>(MAX_CITIES, INF));
22
23 //
24 // Dijkstra演算法，計算從指定城市出發到其他城市的最短路徑
25 void dijkstra(int startCity) {
26     vector<int> dist(numCities, INF);
27     vector<bool> visited(numCities, false);
28 }

```

```

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] <
32             dist[u])) {
33             u = j;
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] +
41                 graph[u][v]);
42         }
43     }
44 }
45
46 // 輸出最短路徑結果
47 cout << "從城市 " << startCity << "
48     出發到其他城市的最短路徑如下：" << endl;
49 for (int i = 0; i < numCities; i++) {
50     if (i != startCity) {
51         cout << "到城市 " << i << " 的最短距離為
52             " << dist[i] << endl;
53     }
54 }
55
56 int main() {
57     // 讀取城市數量和道路數量
58     cin >> numCities >> numRoads;
59
60     // 初始化圖的鄰接矩陣
61     for (int i = 0; i < numRoads; i++) {
62         int city1, city2, distance;
63         cin >> city1 >> city2 >> distance;
64         graph[city1][city2] = distance;
65         graph[city2][city1] = distance; //
66         因為是雙向道路
67     }
68
69     // 選擇起始城市，這裡以城市0為例
70     int startCity = 0;
71
72     // 執行Dijkstra演算法
73     dijkstra(startCity);
74
75     return 0;
76 }

```

5.2 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     }
19 }

```

```

18 while(!path.empty()){
19     cout<<"("<<path.front().first<<" "<<path.front().second<<endl;
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 // 顯示最短距離
56 #include <iostream>
57 #include <utility>
58 #include <deque>
59 #define N 100
60 using namespace std;
61
62 int map[N][N], visited[N][N]={0};
63 typedef pair<int, int> p;
64 int n,m,dis=-2;
65 deque<p> path;
66
67 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==-2)
107         cout<<"No routes accessible.\n";
108     else
109         cout<<"Shortest distance: "<<dis<<endl;
110     return 0;
111 }

```

5.3 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) {
29         temp[k] = arr[i];
30         i++;
31         k++;
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     / Copy the temp array to original array /
41     for (int p = l; p <= r; p++) {
42         arr[p] = temp[p];
43     }
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 }

```

5.4 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     QuickSort(myarray,0,(size-1)); // quick sort called
68
69     cout<<"After Sorting"<<endl;
70     for(int i=0;i<size;i++)
71     {
72         cout<<myarray[i]<<" ";
73     }
74
75     return 0;
76 }

```

5.5 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"
    << endl;
28 for (int i = 0; i < 10; i++) {
29     cin >> myarr[i];
30 }
31 cout << "Please enter an element to search" << endl;
32 cin >> num;
33
34 output = binarySearch(myarr, 0, 9, num);
35
36 if (output == -1) {
37     cout << "No Match Found" << endl;
38 } else {
39     cout << "Match found at position: " << output <<
        endl;
40 }
41
42 return 0;
43 }
44 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
45 Floyd
46 void floyd(){
47     for(int k=0;k<n;k++){ //中間點
48         for(int i=0;i<n;i++){
49             for(int j=0;j<n;j++){
50                 dp[i][j]=min(dp[i][j],dp[i][k]+dp[k][j]);
51                 //經過中間點k的路徑是否小於原始路徑
52                 //小於則更新 不小於則不變動
53                 //窮舉所有鬆弛的可能
54             }
55         }
56     }
57 }

```

6 dp

6.1 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
8 a = 0
9 while True:
10     try:
11         a = int(input())
12         sum = 1
13         for i in range(1,a+1):
14             sum*=i
15         a = str(a)
16         print(a+'!')
17         print(sum)
18     except EOFError:
19         break

```

6.2 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 }

```

6.3 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]; //
    兩條陣列，儲存最近算出來的問題答案。
7
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 %
                2][j-1];
19
20     // 輸出結果
21     cout << "由(0,0)走到(7,7)有" << f[7 % 2][7] <<
        "種走法";
22     // cout << "由(0,0)走到(7,7)有" << f[(H-1) %
        2][W-1] << "種走法";
23 }

```

6.4 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" <<
        f[H-1][W-1];
22
23     int h, w;
24     while (cin >> h >> w)
25         cout << "由(0,0)走到(h,w)的最小總和" << f[h][w];
26 }

```