Contents

1 語法

1.1 c++

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

// struct宣告
struct s{
  int x[100];
  int y[100];
};

s num; //一組s
num.x[1]=1;
num.y[1]=2;

// sort
sort(v.begin(), v.end())      // array 不能用
sort(v, v+n)
sort(v, v+n, greater<int>()) // 大到小

sort(v, v+n, cmp)             // 自己寫比較序列
bool cmp(型態 a, 型態 b){
  return a > b;               // 大到小
}

// set
s.insert(x) //將x插入s中   O(log(n))
s.count(x)   //回傳x是否存在於s中()   O(log(n))
s.erase(x)   //刪除在s中的x   O(log(n))
s.clear()    //刪除s中所有元素  O(n)
s.empty()    //回傳是否為空   O(1)
s.size()     //回傳共有幾個元素   O(1)

map
insert(x) //將x這個pair插入map中   O(log(n))
count(x)   //回傳x這個key是否在map中  O(log(n))
erase(x)   //刪除在map中key為x的  O(log(n))

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

1.2 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()
```

2 字串

2.1 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;
}
```

3 數論

3.1 喵

```cpp
#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";
        }
    }
}
```

3.2 Fibonaccimal

```cpp
#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");
            }
        }
    }
}
```

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

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]
                = max(lcs[0][j-1],
                short{str1[0]==str2[j]});
11         for (int i=1; i<str1.length(); i++)
12         {
13             bool r = i & 1;
14             lcs[r][0] = max(lcs[r^1][0],
                    short{str1[i]==str2[0]});
15             for (int j=1; j<str2.length(); j++)
16                 lcs[r][j] = str1[i]==str2[j] ?
                        lcs[r^1][j-1] + 1 :
                        max(lcs[r^1][j], lcs[r][j-1]);
17         }
18         cout <<
                lcs[(str1.length()-1)&1][str2.length()-1]
                << '\n';
19     }
20     return 0;
21 }
```

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) &&
                (i+x<s.length())){//當找到一個中心點以其為中間然後左
13             x++;
14         }
15         x--;
16         if(2*x+1 >
                maxlen){//只有第一次會max==後後面就追不到那女孩
17             maxlen = 2*x+1;//最大的
18             l = i-x;//計算頭頭
19             r = i+x;//計算尾巴
20         }
21         //偶
22         x = 0;
23         while( (s[i-x]==s[i+1+x]) && (i-x>=0) &&
                (i+1+x<s.length()) ){
24             x++;
25         }
26         if(2*x > maxlen){
27             maxlen = 2*x;
28             l = i-x+1;
29             r = i+x;
30         }
31     }
32     cout << maxlen << '\n';
33     cout << l+1 << ' ' << r+1 << '\n';
34
35 }
```

3.6 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 " <<
                cnt << " (mod 2).\n";
19     }
20     return 0;
21 }
```

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  // 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 }
```

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
          there are any /
27     while (i <= m) {
28         temp[k] = arr[i];
29         i++;
30         k++;
```

```
31     }
32
33     / Copy the remaining elements of second half, if
          there are any /
34     while (j <= r) {
35         temp[k] = arr[j];
36         j++;
37         k++;
38     }
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
          order: " << endl;
75     for (int i = 0; i < size; i++) {
76         cin >> myarray[i];
77     }
78     cout << "Before Sorting" << endl;
79     for (int i = 0; i < size; i++) {
80         cout << myarray[i] << " ";
81     }
82     cout << endl;
83     mergeSort(myarray, 0, (size - 1), size); //
          mergesort(arr,left,right) called
84
85     cout << "After Sorting" << endl;
86     for (int i = 0; i < size; i++) {
87         cout << myarray[i] << " ";
88     }
89
90     return 0;
91 }
```

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
            他就移到最前面
17      //也就是陣列0的位置
18      pIndex++;
19      //下一個比指標小的數值放進陣列1的位置
20      }
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:
        "<<endl;
56   for(int i=0;i<size;i++)
57   {
58   cin>>myarray[i];
59   }
60   cout<<"Before Sorting"<<endl;
61   for(int i=0;i<size;i++)
62   {
63   cout<<myarray[i]<<" ";
64   }
65   cout<<endl;
66
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  }
```

4.5 二分搜

```
1  #include < iostream >
2  using namespace std;
3
4  int binarySearch(int arr[], int left, int right, int
       x) {
5    while (left <= right) {
6      int mid = left + (right - left) / 2;
7
8      if (arr[mid] == x) {
9        return mid;
10     }
11      else if (arr[mid] < x) {
12        left = mid + 1;
13     }
14      else {
15        right = mid - 1;
16     }
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(intk=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  }
```

5 dp

5.1 階乘