

# 1 Basic

## 1.1 data range

```
1 int (-2147483648 to 2147483647)
2 unsigned int(0 to 4294967295)
3 long(-2147483648 to 2147483647)
4 unsigned long(0 to 4294967295)
5 long long(-9223372036854775808 to
   9223372036854775807)
6 unsigned long long (0 to
   18446744073709551615)
```

## 1.2 IO\_fast

```
1 ios_base::sync_with_stdio(0);
2 cin.tie(0);
```

# 2 DP

## 2.1 KMP

```
1 void ComputePrefix(string s, int next[])
2 {
3     int n = s.length();
4     int q, k;
5     next[0] = 0;
6     for (k = 0, q = 1; q < n; q++)
7     {
8         while (k > 0 && s[k] != s[q])
9             k = next[k];
10        if (s[k] == s[q])
11            k++;
12        next[q] = k;
13    }
14 }
15 void KMPMatcher(string text, string pattern)
16 {
17     int n = text.length();
18     int m = pattern.length();
19     int next[pattern.length()];
20     ComputePrefix(pattern, next);
21
22     for (int i = 0, q = 0; i < n; i++)
23     {
24         while (q > 0 && pattern[q] != text[i])
25             q = next[q];
26         if (pattern[q] == text[i])
27             q++;
28         if (q == m)
29         {
30             cout << "Pattern occurs with
31                 shift " << i - m + 1 << endl;
32             ;
33         }
```

```
31         q = 0;
32     }
33 }
34 }
35 string s = "abdcabcdebc";
36 string p = "bcd";
37 KMPMatcher(s, p);
38 cout << endl;
39 return 0;
```

## 2.2 LCS

```
1 int LCS(vector<int> Ans, vector<int> num) //
   Ans 跟 num 都要 index 從1開始放
2 {
3     vector<vector<int>> LCS(N + 1, vector<
4         int>(N + 1, 0));
5     for (int i = 1; i <= N; ++i)
6     {
7         for (int j = 1; j <= N; ++j)
8         {
9             if (Ans[i] == num[j])
10                LCS[i][j] = LCS[i - 1][j -
11                    1] + 1;
12            else
13                LCS[i][j] = max(LCS[i - 1][j
14                    ], LCS[i][j - 1]);
15        }
16    }
17    // printf("%d\n", LCS[N][N]);
18    return LCS[N][N];
19    //列印 LCS
20    vector<int> k;
21    while (n && m)
22    {
23        if (LCS[n][m] != max(LCS[n - 1][m],
24            LCS[n][m - 1]))
25        {
26            k.push_back(arr1[n]);
27            n--;
28            m--;
29        }
30        else if (LCS[n][m] == LCS[n - 1][m])
31            n--;
32        else if (LCS[n][m] == LCS[n][m - 1])
33            m--;
34    }
35    reverse(k.begin(), k.end());
36 }
```

## 2.3 LIC

```
1 void getMaxElementAndPos(vector<int> &LISTbl
   , vector<int> &LISLen, int tNum,
2 int tlen, int tStart, int &num, int &pos
3 )
4 {
5     int max = numeric_limits<int>::min();
```

```
5     int maxPos;
6     for (int i = tStart; i >= 0; i--)
7     {
8         if (LISLen[i] == tlen && LISTbl[i] <
9             tNum)
10        {
11            if (LISTbl[i] > max)
12            {
13                max = LISTbl[i];
14                maxPos = i;
15            }
16        }
17    }
18    num = max;
19    pos = maxPos;
20 }
21 int LIS(vector<int> &LISTbl)
22 {
23     if (LISTbl.size() == 0)
24         return 0;
25     vector<int> LISLen(LISTbl.size(), 1);
26     for (int i = 1; i < LISTbl.size(); i++)
27     {
28         for (int j = 0; j < i; j++)
29         {
30             if (LISTbl[j] < LISTbl[i])
31                 LISLen[i] = max(LISLen[i],
32                     LISLen[j] + 1);
33         }
34     }
35     int maxlen = *max_element(LISLen.begin()
36         , LISLen.end());
37     int num, pos;
38     vector<int> buf;
39     getMaxElementAndPos(LISTbl, LISLen,
40         numeric_limits<int>::max(),
41         maxlen, LISTbl.size() - 1, num, pos);
42     buf.push_back(num);
43     for (int len = maxlen - 1; len >= 1; len
44         --)
45     {
46         int tnum = num;
47         int tpos = pos;
48         getMaxElementAndPos(LISTbl, LISLen,
49             tnum, len, tpos
50             - 1, num, pos);
51         buf.push_back(num);
52     }
53     reverse(buf.begin(), buf.end());
54     for (int k = 0; k < buf.size(); k++) //
55         列印
56     {
57         if (k == buf.size() - 1)
58             cout << buf[k] << endl;
59         else
60             cout << buf[k] << ", ";
61     }
62     return maxlen;
63 }
```

## 2.4 Max\_subarray

```
1 /*Kadane's algorithm*/
2 int maxSubArray(vector<int>& nums) {
3     int local_max = nums[0], global_max =
4     nums[0];
5     for(int i = 1; i < nums.size(); i++){
6         local_max = max(nums[i],nums[i]+
7             local_max);
8         global_max = max(local_max,
9             global_max);
10    }
```

## 2.5 MFlow

```
1 typedef long long ll;
2 struct MF
3 {
4     static const int N = 5000 + 5;
5     static const int M = 60000 + 5;
6     static const ll oo = 1000000000000LL;
7
8     int n, m, s, t, tot, tim;
9     int first[N], next[M];
10    int u[M], v[M], cur[N], vi[N];
11    ll cap[M], flow[M], dis[N];
12    int que[N + N];
13
14    void Clear()
15    {
16        tot = 0;
17        tim = 0;
18        for (int i = 1; i <= n; ++i)
19            first[i] = -1;
20    }
21    void Add(int from, int to, ll cp, ll flw)
22    {
23        u[tot] = from;
24        v[tot] = to;
25        cap[tot] = cp;
26        flow[tot] = flw;
27        next[tot] = first[u[tot]];
28        first[u[tot]] = tot;
29        ++tot;
30    }
31    bool bfs()
32    {
33        ++tim;
34        dis[s] = 0;
35        vi[s] = tim;
36
37        int head, tail;
38        head = tail = 1;
39        que[head] = s;
40        while (head <= tail)
41        {
42            for (int i = first[que[head]]; i
43                != -1; i = next[i])
44            {
```

```

44         if (vi[v[i]] != tim && cap[i]
45             ] > flow[i])
46         {
47             vi[v[i]] = tim;
48             dis[v[i]] = dis[que[head
49                 ]] + 1;
50             que[++tail] = v[i];
51         }
52         ++head;
53     }
54     return vi[t] == tim;
55 }
56 ll dfs(int x, ll a)
57 {
58     if (x == t || a == 0)
59         return a;
60     ll flw = 0, f;
61     int &i = cur[x];
62     for (i = first[x]; i != -1; i = next
63         [i])
64     {
65         if (dis[x] + 1 == dis[v[i]] && (
66             f = dfs(v[i], min(a, cap[i]
67                 - flow[i]))) > 0)
68         {
69             flow[i] += f;
70             flow[v[i]] -= f;
71             a -= f;
72             flw += f;
73             if (a == 0)
74                 break;
75         }
76     }
77     return flw;
78 }
79 ll MaxFlow(int s, int t)
80 {
81     this->s = s;
82     this->t = t;
83     ll flw = 0;
84     while (bfs())
85     {
86         for (int i = 1; i <= n; ++i)
87             cur[i] = 0;
88         flw += dfs(s, oo);
89     }
90     return flw;
91 }
92 // MF Net;
93 // Net.n = n;
94 // Net.Clear();
95 // a 到 b (注意從1開始!!!!)
96 // Net.Add(a, b, w, 0);
97 // Net.MaxFlow(s, d)
98 // s 到 d 的 MF

```

## 3 Geometry

### 3.1 Convexhull 2D

```

1 bool same(double a, double b) { return abs(a
2     - b) < 0; }
3 struct P // 台大
4 {
5     double x, y;
6     P() : x(0), y(0) {}
7     P(double x, double y) : x(x), y(y) {}
8     P operator+(P b) { return P(x + b.x, y +
9         b.y); }
10    P operator-(P b) { return P(x - b.x, y -
11        b.y); }
12    P operator*(double b) { return P(x * b,
13        y * b); }
14    P operator/(double b) { return P(x / b,
15        y / b); }
16    double operator*(P b) { return x * b.x +
17        y * b.y; }
18    double operator^(P b) { return x * b.y -
19        y * b.x; }
20    double abs() { return hypot(x, y); }
21    P unit() { return *this / abs(); }
22    P spin(double o)
23    {
24        double c = cos(o), s = sin(o);
25        return P(c * x - s * y, s * x + c *
26            y);
27    }
28    double angle() { return atan2(y, x); }
29 };
30 bool operator<(const P &a, const P &b) {
31     return same(a.x, b.x) ? a.y < b.y : a.x
32     < b.x; }
33 bool operator>(const P &a, const P &b) {
34     return same(a.x, b.x) ? a.y > b.y : a.x
35     > b.x; }
36 #define crx(a, b, c) ((b - a) ^ (c - a)) //
37 (向量OA叉積向量OB) > 0 表示從OA到OB為
38 逆時針旋轉。
39 vector<P> convex(vector<P> ps) // Andrew's
40 Monotone Chain
41 {
42     vector<P> p;
43     sort(ps.begin(), ps.end(), [](P &a, P &b
44         )
45         { return a.y < b.y || (a.y == b.y
46             && a.x < b.x); });
47     for (int i = 0; i < ps.size(); ++i)
48     {
49         while (p.size() >= 2 && crx(p[p.size()
50             - 2], ps[i], p[p.size() - 1])
51             >= 0)
52             p.pop_back();
53         p.push_back(ps[i]);
54     }
55     int t = p.size();
56     for (int i = (int)p.size() - 2; i >= 0;
57         --i)
58     {

```

```

40         while (p.size() > t && crx(p[p.size()
41             - 2], ps[i], p[p.size() - 1])
42             >= 0)
43             p.pop_back();
44         p.push_back(ps[i]);
45     }
46     // p.pop_back(); //起點依照題目
47     return p;
48 }

```

## 4 Graph

### 4.1 Bellman-Ford

```

1 /*SPA - Bellman-Ford*/
2 #include<bits/stdc++.h>
3 #define inf 99999 //define by you maximum
4 edges weight
5 using namespace std;
6 vector<vector<int>> > edges;
7 vector<int> dist;
8 vector<int> ancestor;
9 void BellmanFord(int start, int node){
10     dist[start] = 0;
11     for(int it = 0; it < node-1; it++){
12         for(int i = 0; i < node; i++){
13             for(int j = 0; j < node; j++){
14                 if(edges[i][j] != -1){
15                     if(dist[i] + edges[i][j]
16                         < dist[j]){
17                         dist[j] = dist[i] +
18                             edges[i][j];
19                         ancestor[j] = i;
20                     }
21                 }
22             }
23         }
24     }
25     for(int i = 0; i < node; i++) //
26         negative cycle detection
27         for(int j = 0; j < node; j++){
28             if(dist[i] + edges[i][j] < dist[
29                 j]){
30                 cout<<"Negative cycle!"<<
31                     endl;
32                 return;
33             }
34         }
35 }
36 int main(){
37     int node;
38     cin>>node;
39     edges.resize(node, vector<int>(node, inf));
40     ;
41     dist.resize(node, inf);
42     vector<pair<int, int>> edges;
43     int a, b, d;
44     while(cin>>a>>b>>d){
45         /*input: source destination weight*/
46         if(a == -1 && b == -1 && d == -1)

```

### 4.2 BFS-queue

```

1 /*BFS - queue version*/
2 #include<bits/stdc++.h>
3 using namespace std;
4 void BFS(vector<int> &result, vector<pair<int
5     ,int> > edges, int node, int start){
6     vector<int> pass(node, 0);
7     queue<int> q;
8     queue<int> p;
9     q.push(start);
10    int count = 1;
11    vector<pair<int, int>> newedges;
12    while(!q.empty()){
13        pass[q.front()] = 1;
14        for (int i = 0; i < edges.size(); i
15            ++){
16            if(edges[i].first == q.front()
17                && pass[edges[i].second] ==
18                0){
19                p.push(edges[i].second);
20                result[edges[i].second] =
21                    count;
22            }
23            else if(edges[i].second == q.
24                front() && pass[edges[i].
25                    first] == 0){
26                p.push(edges[i].first);
27                result[edges[i].first] =
28                    count;
29            }
30            else
31                newedges.push_back(edges[i]);
32        }
33    }
34    edges = newedges;
35    newedges.clear();
36    q.pop();
37    if(q.empty() == true){
38        q = p;
39        queue<int> tmp;
40        p = tmp;
41        count++;
42    }
43 }
44 int main(){
45     int node;
46     cin >> node;
47     vector<pair<int, int>> edges;
48     int a, b;
49     while(cin>>a>>b){
50         /*a = b = -1 means input edges ended
51         */

```

```

43     if(a == -1 && b == -1)
44         break;
45     edges.push_back(pair<int, int>(a, b)
46     );
47 }
48 vector<int> result(node, -1);
49 BFS(result, edges, node, 0);
50 return 0;
51 }

```

### 4.3 DFS-rec

```

1  /*DFS - Recursive version*/
2  #include<bits/stdc++.h>
3  using namespace std;
4  map<pair<int,int>,int> edges;
5  vector<int> pass;
6  vector<int> route;
7  void DFS(int start){
8      pass[start] = 1;
9      map<pair<int,int>,int>::iterator iter;
10     for(iter = edges.begin(); iter != edges.
11         end(); iter++){
12         if((*iter).first.first == start &&
13             (*iter).second == 0 && pass[(*)
14             iter).first.second] == 0){
15             route.push_back((*iter).first.
16                 second);
17             DFS((*iter).first.second);
18         }
19         else if((*iter).first.second ==
20             start && (*iter).second == 0 &&
21             pass[(*)iter).first.first] == 0){
22             route.push_back((*iter).first.
23                 first);
24             DFS((*iter).first.first);
25         }
26     }
27 }
28 int main(){
29     int node;
30     cin>>node;
31     pass.resize(node,0);
32     int a,b;
33     while(cin>>a>>b){
34         if(a == -1 && b == -1)
35             break;
36         edges.insert(pair<pair<int,int>,int>
37             >(pair<int,int>(a,b),0));
38     }
39     int start;
40     cin>>start;
41     route.push_back(start);
42     DFS(start);
43     return 0;
44 }

```

### 4.4 Dijkstra

```

1  /*SPA - Dijkstra*/
2  #include<bits/stdc++.h>
3  #define inf INT_MAX
4  using namespace std;
5  vector<vector<int>> weight;
6  vector<int> ancestor;
7  vector<int> dist;
8  void dijkstra(int start){
9      priority_queue<pair<int,int>,vector<
10         pair<int,int>,greater<pair<int,
11         int>>> pq;
12      pq.push(make_pair(0,start));
13      while(!pq.empty()){
14          int cur = pq.top().second;
15          pq.pop();
16          for(int i = 0; i < weight[cur].size
17              (); i++){
18              if(dist[i] > dist[cur] + weight[
19                  cur][i] && weight[cur][i] !=
20                  -1){
21                  dist[i] = dist[cur] + weight
22                      [cur][i];
23                  ancestor[i] = cur;
24                  pq.push(make_pair(dist[i],i)
25                      );
26              }
27          }
28      }
29      int main(){
30          int node;
31          cin>>node;
32          int a,b,d;
33          weight.resize(node,vector<int>(node,-1))
34          ;
35          while(cin>>a>>b>>d){
36              /*input: source destination weight*/
37              if(a == -1 && b == -1 && d == -1)
38                  break;
39              weight[a][b] = d;
40          }
41          ancestor.resize(node,-1);
42          dist.resize(node,inf);
43          int start;
44          cin>>start;
45          dist[start] = 0;
46          dijkstra(start);
47          return 0;
48      }
49  }

```

### 4.5 union\_find

```

1  int find(int x,vector<int> &union_set){
2      if(union_set[x] != x)
3          union_set[x] = find(union_set[x],
4              union_set); //compress path
5      return union_set[x];
6  }
7  void merge(int x,int y,vector<int> &
8      union_set,vector<int> &rank){
9      int rx, ry;
10     rx = find(x,union_set);
11     ry = find(y,union_set);

```

```

10     if(rx == ry)
11         return;
12     /*merge by rank -> always merge small
13         tree to big tree*/
14     if(rank[rx] > rank[ry])
15         union_set[ry] = rx;
16     else
17     {
18         union_set[rx] = ry;
19         if(rank[rx] == rank[ry])
20             ++rank[ry];
21     }
22 }
23 int main(){
24     int node;
25     cin>>node; //Input Node number
26     vector<int> union_set(node, 0);
27     vector<int> rank(node, 0);
28     for (int i = 0; i < node; i++)
29         union_set[i] = i;
30     int edge;
31     cin>>edge; //Input Edge number
32     for(int i = 0; i < edge; i++)
33     {
34         int a, b;
35         cin>>a>>b;
36         merge(a, b, union_set,rank);
37     }
38     /*build party*/
39     vector<vector<int>> party(node, vector<
40         int>(0));
41     for (int i = 0; i < node; i++)
42         party[find(i, union_set)].push_back(
43             i);
44 }

```

## 5 Mathematics

### 5.1 Combination

```

1  /*input type string or vector*/
2  for (int i = 0; i < (1 << input.size()); ++i
3      )
4  {
5      string testCase = "";
6      for (int j = 0; j < input.size(); ++j)
7          if (i & (1 << j))
8              testCase += input[j];
9  }

```

### 5.2 Extended Euclidean

```

1  // ax + by = gcd(a,b)
2  pair<long long, long long> extgcd(long long
3      a, long long b)
4  {
5      if (b == 0)
6          return {1, 0};

```

```

6      long long k = a / b;
7      pair<long long, long long> p = extgcd(b,
8          a - k * b);
9      //cout << p.first << " " << p.second <<
10         endl;
11      //cout << "商數(k)= " << k << endl <<
12         endl;
13      return {p.second, p.first - k * p.second
14          };
15  }
16  int main()
17  {
18      int a, b;
19      cin>>a>>b;
20      pair<long long, long long> xy = extgcd(a
21          , b); //(x0,y0)
22      cout << xy.first << " " << xy.second <<
23          endl;
24      cout << xy.first << " * " << a << " + "
25          << xy.second << " * " << b << endl;
26      return 0;
27  }
28  // ax + by = gcd(a,b) * r
29  /*find |x|+|y| -> min*/
30  int main()
31  {
32      long long r, p, q; /*px+qy = r*/
33      int cases;
34      cin>>cases;
35      while (cases-->0)
36      {
37          cin>>r>>p>>q;
38          pair<long long, long long> xy =
39              extgcd(q, p); //(x0,y0)
40          long long ans = 0, tmp = 0;
41          double k, k1;
42          long long s, s1;
43          k = 1 - (double)(r * xy.first) / p;
44          s = round(k);
45          ans = llabs(r * xy.first + s * p) +
46              llabs(r * xy.second - s * q);
47          k1 = -(double)(r * xy.first) / p;
48          s1 = round(k1);
49          /*cout << k << endl << k1 << endl;
50              cout << s << endl << s1 << endl;
51              */
52          tmp = llabs(r * xy.first + s1 * p) +
53              llabs(r * xy.second - s1 * q);
54          ans = min(ans, tmp);
55          cout << ans << endl;
56      }
57      return 0;
58  }

```

### 5.3 Hex to Dec

```

1  int HextoDec(string num) //16 to 10
2  {
3      int base = 1;
4      int temp = 0;

```

```

5 for (int i = num.length() - 1; i = 0; i --) // if(!prime[數字])
6 { // 我是質數
7     if (num[i] = '0' && num[i] = '9')
8     {
9         temp += (num[i] - 48) base;
10        base = base 16;
11    }
12    else if (num[i] = 'A' && num[i] = 'F')
13    {
14        temp += (num[i] - 55) base;
15        base = base 16;
16    }
17 }
18 return temp;
19 }
20 void DecToHex(int p_intValue) //10 to 16
21 {
22     char l_pCharRes = new (char);
23     sprintf(l_pCharRes, %X, p_intValue);
24     int l_intResult = stoi(l_pCharRes);
25     cout << l_pCharRes << endl;
26     return l_intResult;
27 }

```

## 5.7 二分逼近法

```

1 #define eps 1e-14
2 void half_interval()
3 {
4     double L = 0, R = /*區間*/, M;
5     while (R - L >= eps)
6     {
7         M = (R + L) / 2;
8         if (/*函數*/ > /*方程式目標*/)
9             L = M;
10        else
11            R = M;
12    }
13    printf("%.3lf\n", R);
14 }

```

## 5.8 四則運算

```

1 string s = ""; //開頭是負號要補0
2 long long int DFS(int le, int ri) // (0,
3     string final index)
4 {
5     int c = 0;
6     for (int i = ri; i >= le; i--)
7     {
8         if (s[i] == ')')
9             c++;
10        if (s[i] == '(')
11            c--;
12        if (s[i] == '+' && c == 0)
13            return DFS(le, i - 1) + DFS(i + 1, ri);
14        if (s[i] == '-' && c == 0)
15            return DFS(le, i - 1) - DFS(i + 1, ri);
16    }
17    for (int i = ri; i >= le; i--)
18    {
19        if (s[i] == ')')
20            c++;
21        if (s[i] == '(')
22            c--;
23        if (s[i] == '*' && c == 0)
24            return DFS(le, i - 1) * DFS(i + 1, ri);
25        if (s[i] == '/' && c == 0)
26            return DFS(le, i - 1) / DFS(i + 1, ri);
27        if (s[i] == '%' && c == 0)
28            return DFS(le, i - 1) % DFS(i + 1, ri);
29    }
30    if ((s[le] == '(' && (s[ri] == ')'))
31        return DFS(le + 1, ri - 1); //去除刮
32        號

```

## 5.4 Permutation

```

1 // 全排列要先 sort !!!
2 // num -> vector or string
3 next_permutation(num.begin(), num.end());
4 prev_permutation(num.begin(), num.end());

```

## 5.5 PI

```

1 #define PI acos(-1)
2 #define PI_M_PI

```

## 5.6 Prime table

```

1 // 埃拉托斯特尼篩法
2 const int maxn = 10000000;
3 bitset<maxn> prime;
4 void sieve()
5 {
6     for (int i = 2; i * i < maxn; ++i)
7     {
8         if (prime[i] == 0)
9         {
10            for (int j = i * i; j < maxn; j += i)
11                prime[j] = 1;
12        }
13    }
14 }
15 /* 0跟1要寫if過濾掉 */

```

```

31 if (s[le] == ' ' && s[ri] == ' ')
32     return DFS(le + 1, ri - 1); //去除左
33     右兩邊空格
34 if (s[le] == ' ')
35     return DFS(le + 1, ri); //去除左邊空
36     格
37 if (s[ri] == ' ')
38     return DFS(le, ri - 1); //去除右邊空
39     格
40 long long int num = 0;
41 for (int i = le; i <= ri; i++)
42     num = num * 10 + s[i] - '0';
43 return num;

```

## 5.9 數字乘法組合

```

1 void toans(vector<vector<int>> &ans, vector<
2     int> com)
3 {
4     // sort(com.begin(), com.end());
5     ans.push_back(com);
6     // for (auto i : com)
7     //     cout << i << ' ';
8     // cout << endl;
9 }
10 void finds(int j, int old, int num, vector<
11     int> com, vector<vector<int>> &ans)
12 {
13     for (int i = j; i <= sqrt(num); i++)
14     {
15         if (old == num)
16             com.clear();
17         if (num % i == 0)
18         {
19             vector<int> a;
20             a = com;
21             a.push_back(i);
22             finds(i, old, num / i, a, ans);
23             a.push_back(num / i);
24             toans(ans, a);
25         }
26     }
27 }
28 int main()
29 {
30     vector<vector<int>> ans;
31     vector<int> zero;
32     finds(2, num, num, zero, ans);
33     // num 為 input 數字
34     for (int i = 0; i < ans.size(); i++)
35     {
36         for (int j = 0; j < ans[i].size() - 1; j++)
37             cout << ans[i][j] << " ";
38         cout << ans[i][ans[i].size() - 1] << endl;
39     }

```

## 5.10 數字加法組合

```

1 void printCombination(vector<int> const &out
2     , int m, vector<vector<int>> &ans)
3 {
4     for (int i : out)
5         if (i > m)
6             return;
7     ans.push_back(out);
8 }
9 void recur(int i, int n, int m, vector<int>
10     &out, vector<vector<int>> &ans)
11 {
12     if (n == 0)
13         printCombination(out, m, ans);
14     for (int j = i; j <= n; j++)
15     {
16         out.push_back(j);
17         recur(j, n - j, m, out, ans);
18         out.pop_back();
19     }
20 }
21 int main()
22 {
23     vector<vector<int>> ans;
24     vector<int> zero;
25     recur(1, num, num, zero, ans);
26     // num 為 input 數字
27     for (int i = 0; i < ans.size(); i++)
28     {
29         for (int j = 0; j < ans[i].size() - 1; j++)
30             cout << ans[i][j] << " ";
31         cout << ans[i][ans[i].size() - 1] << endl;
32     }

```

## 5.11 羅馬數字

```

1 int romanToInt(string s)
2 {
3     unordered_map<char, int> T;
4     T['I'] = 1;
5     T['V'] = 5;
6     T['X'] = 10;
7     T['L'] = 50;
8     T['C'] = 100;
9     T['D'] = 500;
10    T['M'] = 1000;
11
12    int sum = T[s.back()];
13    for (int i = s.length() - 2; i >= 0; --i)
14    {
15        if (T[s[i]] < T[s[i + 1]])
16            sum -= T[s[i]];
17        else
18            sum += T[s[i]];
19    }
20    return sum;

```

21 | }

## 5.12 質因數分解

```

1 void cal(int in)
2 {
3     for (long long x = 2; x <= in; x++)
4     {
5         while (in % x == 0)
6         {
7             cout << x << " ";
8             in /= x;
9         }
10    }
11 }
```

## 6 Other

### 6.1 Weighted Job Scheduling

```

1 struct Job
2 {
3     int start, finish, profit;
4 };
5 bool jobComparataor(Job s1, Job s2)
6 {
7     return (s1.finish < s2.finish);
8 }
9 int latestNonConflict(Job arr[], int i)
10 {
11     for (int j = i - 1; j >= 0; j--)
12     {
13         if (arr[j].finish <= arr[i].start)
14             return j;
15     }
16     return -1;
17 }
18 int findMaxProfit(Job arr[], int n)
19 {
20     sort(arr, arr + n, jobComparataor);
21     int *table = new int[n];
22     table[0] = arr[0].profit;
23     for (int i = 1; i < n; i++)
24     {
25         int inclProf = arr[i].profit;
26         int l = latestNonConflict(arr, i);
27         if (l != -1)
28             inclProf += table[l];
29         table[i] = max(inclProf, table[i - 1]);
30     }
31     int result = table[n - 1];
32     delete[] table;
33     return result;
34 }
35 }
```

## 6.2 數獨解法

```

1 int getSquareIndex(int row, int column, int n)
2 {
3     return row / n * n + column / n;
4 }
5
6 bool backtracking(vector<vector<int>> &board,
7     vector<vector<bool>> &rows, vector<
8     vector<bool>> &cols,
9     vector<vector<bool>> &boxes,
10    int index, int n)
11 {
12     int n2 = n * n;
13     int rowNum = index / n2, colNum = index % n2;
14     if (index >= n2 * n2)
15         return true;
16
17     if (board[rowNum][colNum] != 0)
18         return backtracking(board, rows, cols, boxes, index + 1, n);
19
20     for (int i = 1; i <= n2; i++)
21     {
22         if (!rows[rowNum][i] && !cols[colNum][i] && !boxes[getSquareIndex(rowNum, colNum, n)][i])
23         {
24             rows[rowNum][i] = true;
25             cols[colNum][i] = true;
26             boxes[getSquareIndex(rowNum, colNum, n)][i] = true;
27             board[rowNum][colNum] = i;
28             if (backtracking(board, rows, cols, boxes, index + 1, n))
29                 return true;
30             board[rowNum][colNum] = 0;
31             rows[rowNum][i] = false;
32             cols[colNum][i] = false;
33             boxes[getSquareIndex(rowNum, colNum, n)][i] = false;
34         }
35     }
36     return false;
37 }
38
39 /*用法 main*/
40 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
41 vector<vector<int>> board(n * n + 1, vector<int>(n * n + 1, 0));
42 vector<vector<bool>> isRow(n * n + 1, vector<bool>(n * n + 1, false));
43 vector<vector<bool>> isColumn(n * n + 1, vector<bool>(n * n + 1, false));
44 vector<vector<bool>> isSquare(n * n + 1, vector<bool>(n * n + 1, false));
45
46 for (int i = 0; i < n * n; ++i)
47 {
48     for (int j = 0; j < n * n; ++j)
49     {
50         int number;
51         cin >> number;
52         board[i][j] = number;
53     }
54 }
```

```

49     if (number == 0)
50         continue;
51     isRow[i][number] = true;
52     isColumn[j][number] = true;
53     isSquare[getSquareIndex(i, j, n)][number] = true;
54 }
55 }
56 if (backtracking(board, isRow, isColumn, isSquare, 0, n))
57     /*有解答*/
58 else
59     /*解答*/
60
61 string minWindow(string s, string t) {
62     unordered_map<char, int> letterCnt;
63     for (int i = 0; i < t.length(); i++)
64         letterCnt[t[i]]++;
65     int minLength = INT_MAX, minStart = -1;
66     int left = 0, matchCnt = 0;
67     for (int i = 0; i < s.length(); i++)
68     {
69         if (--letterCnt[s[i]] >= 0)
70             matchCnt++;
71         while (matchCnt == t.length())
72         {
73             if (i - left + 1 < minLength)
74             {
75                 minLength = i - left + 1;
76                 minStart = left;
77             }
78             if (++letterCnt[s[left]] > 0)
79                 matchCnt--;
80             left++;
81         }
82     }
83     return minLength == INT_MAX ? "" : s.substr(minStart, minLength);
84 }
```

## 7 String

### 7.1 sliding window

```

1 string minWindow(string s, string t) {
2     unordered_map<char, int> letterCnt;
3     for (int i = 0; i < t.length(); i++)
4         letterCnt[t[i]]++;
5     int minLength = INT_MAX, minStart = -1;
6     int left = 0, matchCnt = 0;
7     for (int i = 0; i < s.length(); i++)
8     {
9         if (--letterCnt[s[i]] >= 0)
10             matchCnt++;
11         while (matchCnt == t.length())
12         {
13             if (i - left + 1 < minLength)
14             {
15                 minLength = i - left + 1;
16                 minStart = left;
17             }
18             if (++letterCnt[s[left]] > 0)
19                 matchCnt--;
20             left++;
21         }
22     }
23     return minLength == INT_MAX ? "" : s.substr(minStart, minLength);
24 }
```

### 7.2 split

```

1 vector<string> mysplit(const string& str,
2     const string& delim)
3 {
4     vector<string> res;
5     if (" " == str)
6         return res;
7     char *strs = new char[str.length() + 1];
8     strcpy(strs, str.c_str());
9
10    char *d = new char[delim.length() + 1];
```

```

11    strcpy(d, delim.c_str());
12
13    char *p = strtok(strs, d);
14    while (p)
15    {
16        string s = p;
17        res.push_back(s);
18        p = strtok(NULL, d);
19    }
20    return res;
21 }
```

## 8 data structure

### 8.1 Bigint

```

1 //台大
2 struct Bigint{
3     static const int LEN = 60;
4     static const int BIGMOD = 10000;
5     int s;
6     int vl, v[LEN];
7     // vector<int> v;
8     Bigint() : s(1) { vl = 0; }
9     Bigint(long long a) {
10         s = 1; vl = 0;
11         if (a < 0) { s = -1; a = -a; }
12         while (a) {
13             push_back(a % BIGMOD);
14             a /= BIGMOD;
15         }
16     }
17     Bigint(string str) {
18         s = 1; vl = 0;
19         int stPos = 0, num = 0;
20         if (!str.empty() && str[0] == '-') {
21             stPos = 1;
22             s = -1;
23         }
24         for (int i = SZ(str) - 1, q = 1; i >= stPos; i--) {
25             num += (str[i] - '0') * q;
26             if ((q *= 10) >= BIGMOD) {
27                 push_back(num);
28                 num = 0; q = 1;
29             }
30         }
31         if (num) push_back(num);
32         n();
33     }
34     int len() const {
35         return vl; //return SZ(v);
36     }
37     bool empty() const { return len() == 0; }
38     void push_back(int x) {
39         v[vl++] = x; //v.PB(x);
40     }
41     void pop_back() {
42         vl--; //v.pop_back();
43     }
44 }
```



```

44 int back() const {
45     return v[vl-1]; //return v.back();
46 }
47 void n() {
48     while (!empty() && !back()) pop_back
49         ();
50 }
51 void resize(int nl) {
52     vl = nl; //v.resize(nl);
53     fill(v, v+vl, 0); //fill(ALL(v), 0);
54 }
55 void print() const {
56     if (empty()) { putchar('0'); return;
57     }
58     if (s == -1) putchar('-');
59     printf("%d", back());
60     for (int i=len()-2; i>=0; i--)
61         printf("%.4d", v[i]);
62 }
63 friend ostream& operator << (std::
64     ostream& out, const Bigint &a) {
65     if (a.empty()) { out << "0"; return
66     out; }
67     if (a.s == -1) out << "-";
68     out << a.back();
69     for (int i=a.len()-2; i>=0; i--) {
70         char str[10];
71         snprintf(str, 5, "%.4d", a.v[i])
72         ;
73         out << str;
74     }
75     return out;
76 }
77 int cp3(const Bigint &b) const {
78     if (s != b.s) return s - b.s;
79     if (s == -1) return -(*this).cp3(-b
80     );
81     if (len() != b.len()) return len()-b
82     .len(); //int
83     for (int i=len()-1; i>=0; i--)
84         if (v[i] != b.v[i]) return v[i]-b
85         .v[i];
86     return 0;
87 }
88 bool operator<(const Bigint &b) const
89 { return cp3(b)<0; }
90 bool operator<=(const Bigint &b) const
91 { return cp3(b)<=0; }
92 bool operator==(const Bigint &b) const
93 { return cp3(b)==0; }
94 bool operator!=(const Bigint &b) const
95 { return cp3(b)!=0; }
96 bool operator>(const Bigint &b) const
97 { return cp3(b)>0; }
98 bool operator>=(const Bigint &b) const
99 { return cp3(b)>=0; }
100 Bigint operator - () const {
101     Bigint r = (*this);
102     r.s = -r.s;
103     return r;
104 }
105 Bigint operator + (const Bigint &b)
106     const {
107     if (s == -1) return -(*this)+(-b))
108     ;
109     if (b.s == -1) return (*this)-(-b);
110 }
111
112 Bigint r;
113 int nl = max(len(), b.len());
114 r.resize(nl + 1);
115 for (int i=0; i<nl; i++) {
116     if (i < len()) r.v[i] += v[i];
117     if (i < b.len()) r.v[i] += b.v[i]
118     ];
119     if (r.v[i] >= BIGMOD) {
120         r.v[i+1] += r.v[i] / BIGMOD;
121         r.v[i] %= BIGMOD;
122     }
123 }
124 r.n();
125 return r;
126 }
127 Bigint operator - (const Bigint &b)
128     const {
129     if (s == -1) return -(*this)-(-b))
130     ;
131     if (b.s == -1) return (*this)+(-b);
132     if ((*this) < b) return -(b-(*this))
133     ;
134     Bigint r;
135     r.resize(len());
136     for (int i=0; i<len(); i++) {
137         r.v[i] += v[i];
138         if (i < b.len()) r.v[i] -= b.v[i]
139         ];
140         if (r.v[i] < 0) {
141             r.v[i] += BIGMOD;
142             r.v[i+1]--;
143         }
144     }
145     r.n();
146     return r;
147 }
148 Bigint operator * (const Bigint &b) {
149     Bigint r;
150     r.resize(len() + b.len() + 1);
151     r.s = s * b.s;
152     for (int i=0; i<len(); i++) {
153         for (int j=0; j<b.len(); j++) {
154             r.v[i+j] += v[i] * b.v[j];
155             if (r.v[i+j] >= BIGMOD) {
156                 r.v[i+j+1] += r.v[i+j] /
157                 BIGMOD;
158                 r.v[i+j] %= BIGMOD;
159             }
160         }
161     }
162     r.n();
163     return r;
164 }
165 Bigint operator / (const Bigint &b) {
166     Bigint r;
167     r.resize(max(1, len()-b.len()+1));
168     int oriS = s;
169     Bigint b2 = b; // b2 = abs(b)
170     s = b2.s = r.s = 1;
171     for (int i=r.len()-1; i>=0; i--) {
172         int d=0, u=BIGMOD-1;
173         while(d<u) {
174             int m = (d+u+1)>>1;
175             r.v[i] = m;
176             if((r*b2) > (*this)) u = m
177             -1;
178         }
179     }
180     r.n();
181     return r;
182 }

```

```

183         else d = m;
184     }
185     r.v[i] = d;
186 }
187 s = oriS;
188 r.s = s * b.s;
189 r.n();
190 return r;
191 }
192 Bigint operator % (const Bigint &b) {
193     return (*this)-(*this)/b*b;
194 }
195 }
196 };
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

## 8.2 分數

```

1 class Rational
2 {
3     friend istream &operator>>(istream &,
4     Rational &);
5     friend ostream &operator<<(ostream &,
6     const Rational &);
7 public:
8     Rational() //constructor one
9     {
10         m_numerator = 0;
11         m_denominator = 1;
12     }
13     Rational(int a, int b) //constructor two
14     {
15         if (b < 0 || b == 0) //avoids negative
16             denominators. && prevents a 0
17             denominator
18         {
19             cout << "This Rational number can't be
20             used.\n\n";
21             m_numerator = 0;
22             m_denominator = 0;
23         }
24         else
25         {
26             cout << "This Rational number can be
27             used.\n\n";
28             m_numerator = a;
29             m_denominator = b;
30         }
31     }
32     Rational operator+(const Rational& a); //
33     加
34     Rational operator-(const Rational& a); //
35     減
36     Rational operator*(const Rational& a); //
37     乘
38     Rational operator/(const Rational& a); //
39     除
40     bool operator==(const Rational& a); //相
41     等
42     void reduce(); //化簡
43 private:
44     int m_numerator;
45     int m_denominator;
46 };

```

```

36 istream &operator>>(istream &input, Rational
37     &test)
38 {
39     char temp;
40     input >> test.m_numerator;
41     input >> temp;
42     input >> test.m_denominator;
43     Rational final(test.m_numerator, test.
44         m_denominator); //final用來告訴使用者
45         這數字不符合!
46     if (test.m_denominator < 0 || test.
47         m_denominator == 0) //不符合(再輸入
48         一次)
49     {
50         while (test.m_denominator < 0 || test.
51             m_denominator == 0) //有可能輸入的
52             東西還是不符合,所以用迴圈
53     {
54         cout << "Enter another Rational number
55         (n/d): ";
56         input >> test.m_numerator;
57         input >> temp;
58         input >> test.m_denominator;
59         Rational final(test.m_numerator, test.
60             m_denominator); //final用來告訴使
61             用者這數字不符合!
62     }
63     return input;
64 }
65 else
66     return input;
67 }
68 ostream &operator<<(ostream &output, const
69     Rational &test)
70 {
71     output << test.m_numerator;
72     if(test.m_numerator == 0)
73         return output;
74     if (test.m_denominator == 1)
75         return output;
76     else
77     {
78         output << "/";
79         output << test.m_denominator;
80     }
81     return output;
82 }
83 Rational Rational::operator+(const Rational&
84     a)
85 {
86     Rational c;
87     c.m_denominator = this->m_denominator * a.
88         m_denominator; //通分(同乘)
89     c.m_numerator = (this->m_numerator * a.
90         m_denominator) + (a.m_numerator * this
91         ->m_denominator);
92     c.reduce();
93     return c;
94 }
95 Rational Rational::operator-(const Rational&
96     a)
97 {
98     Rational c;
99 }

```

```
84 | c.m_denominator = this->m_denominator * a.  
    | m_denominator;  
85 | c.m_numeitor = (this->m_numeitor * a.  
    | m_denominator) - (a.m_numeitor * this  
    | ->m_denominator);  
86 | c.reduce();  
87 | return c;  
88 | }  
89 | Rational Rational::operator*(const Rational&  
    | a)  
90 | {  
91 |     Rational c;  
92 |     c.m_denominator = this->m_denominator * a.  
    | m_denominator;  
93 |     c.m_numeitor = this->m_numeitor * a.  
    | m_numeitor;  
94 |     c.reduce();  
95 |     return c;  
96 | }  
97 | Rational Rational::operator/(const Rational&  
    | a)  
98 | {  
99 |     Rational c;  
100 |    c.m_denominator = this->m_denominator * a.  
    | m_numeitor;  
101 |    c.m_numeitor = this->m_numeitor * a.  
    | m_denominator;  
102 |    c.reduce();  
103 |    return c;  
104 | }  
105 | bool Rational::operator==(const Rational& a)  
106 | {  
107 |     if (m_numeitor == a.m_numeitor)  
108 |     {  
109 |         if (m_denominator == a.m_denominator)  
110 |             return true;  
111 |         else  
112 |             return false;  
113 |     }  
114 |     else  
115 |         return false;  
116 | }  
117 | void Rational::reduce()  
118 | {  
119 |     int i;  
120 |     int max;  
121 |     if(m_numeitor > m_denominator)  
122 |         max = m_numeitor;  
123 |     else  
124 |         max = m_denominator;  
125 |     for (i = 2; i <= max; i++)  
126 |     {  
127 |         if (m_denominator % i == 0 && m_numeitor  
    |             % i == 0)  
128 |         {  
129 |             m_denominator /= i;  
130 |             m_numeitor /= i;  
131 |             i = 1;  
132 |             max = m_denominator;  
133 |             continue;  
134 |         }  
135 |     }  
136 | }
```

TO DO WRITING  
NOT THINKING

Contents

1	Basic	1	2	DP	1	4.4	Dijkstra	3	5.11	羅馬數字	4
1.1	data range	1	2.1	KMP	1	4.5	union_find	3	5.12	質因數分解	5
1.2	IO_fast	1	2.2	LCS	1	5	Mathematics	3	6	Other	5
			2.3	LIC	1	5.1	Combination	3	6.1	Weighted Job Scheduling	5
			2.4	Max_subarray	1	5.2	Extended Euclidean	3	6.2	數獨解法	5
			2.5	MFlow	1	5.3	Hex to Dec	3	7	String	5
			3	Geometry	2	5.4	Permutation	4	7.1	sliding window	5
			3.1	Convexhull 2D	2	5.5	PI	4	7.2	split	5
			4	Graph	2	5.6	Prime table	4	8	data structure	5
			4.1	Bellman-Ford	2	5.7	二分逼近法	4	8.1	Bigint	5
			4.2	BFS-queue	2	5.8	四則運算	4	8.2	分數	6
			4.3	DFS-rec	3	5.9	數字乘法組合	4			
						5.10	數字加法組合	4			