

# 1 DP

## 1.1 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             q = 0;
33         }
34     }
35     string s = "abcdabcdebc";
36     string p = "bcd";
37     KMPMatcher(s, p);
38     cout << endl;
39     return 0;

```

## 1.2 LCS

```

1 int LCS(vector<int> Ans, vector<int> num) //
2     Ans 跟 num 都要 index 從1開始放
3 {
4     vector<vector<int>> LCS(N + 1, vector<
5         int>(N + 1, 0));
6     for (int i = 1; i <= N; i++)
7     {
8         for (int j = 1; j <= N; j++)
9         {
10            if (Ans[i] == num[j])

```

```

9         LCS[i][j] = LCS[i - 1][j -
10             1] + 1;
11         else
12             LCS[i][j] = max(LCS[i - 1][j
13                 ], LCS[i][j - 1]);
14     }
15 }
16 // printf("%d\n", LCS[N][N]);
17 return LCS[N][N];
18 //列印 LCS
19 vector<int> k;
20 while (n && m)
21 {
22     if (LCS[n][m] != max(LCS[n - 1][m],
23         LCS[n][m - 1]))
24     {
25         k.push_back(arr1[n]);
26         n--;
27         m--;
28     }
29     else if (LCS[n][m] == LCS[n - 1][m])
30         n--;
31     else if (LCS[n][m] == LCS[n][m - 1])
32         m--;
33 }
reverse(k.begin(), k.end());

```

## 1.3 LIC

```

1 void getMaxElementAndPos(vector<int> &LISTbl
2     , vector<int> &LISlen, int tNum,
3     int tlen, int tStart, int &num, int &pos
4     )
5 {
6     int max = numeric_limits<int>::min();
7     int maxPos;
8     for (int i = tStart; i >= 0; i--)
9     {
10        if (LISlen[i] == tlen && LISTbl[i] <
11            tNum)
12        {
13            if (LISTbl[i] > max)
14            {
15                max = LISTbl[i];
16                maxPos = i;
17            }
18        }
19    }
20    num = max;
21    pos = maxPos;
22 }
23 int LIS(vector<int> &LISTbl)
24 {
25     if (LISTbl.size() == 0)
26         return 0;
27     vector<int> LISlen(LISTbl.size(), 1);
28     for (int i = 1; i < LISTbl.size(); i++)
29     {
30         for (int j = 0; j < i; j++)
31         {
32             if (LISTbl[j] < LISTbl[i])

```

```

4         LISlen[i] = max(LISlen[i],
5             LISlen[j] + 1);
6     }
7 }
8 int maxlen = *max_element(LISlen.begin()
9     , LISlen.end());
10 int num, pos;
11 vector<int> buf;
12 getMaxElementAndPos(LISTbl, LISlen,
13     numeric_limits<int>
14     >::max(),
15     maxlen, LISTbl.size()
16     - 1, num, pos);
17 buf.push_back(num);
18 for (int len = maxlen - 1; len >= 1; len
19     --)
20 {
21     int tnum = num;
22     int tpos = pos;
23     getMaxElementAndPos(LISTbl, LISlen,
24         tnum, len, tpos
25         - 1, num,
26         pos);
27     buf.push_back(num);
28 }
29 reverse(buf.begin(), buf.end());
30 for (int k = 0; k < buf.size(); k++) //
31     列印
32 {
33     if (k == buf.size() - 1)
34         cout << buf[k] << endl;
35     else
36         cout << buf[k] << ",";
37 }
38 return maxlen;
39 }

```

## 1.4 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    }
11    return global_max;
12 }

```

## 1.5 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            {
45                if (vi[v[i]] != tim && cap[i]
46                    > flow[i])
47                {
48                    vi[v[i]] = tim;
49                    dis[v[i]] = dis[que[head]]
50                        + 1;
51                    que[++tail] = v[i];
52                }
53            }
54            ++head;
55        }
56        return vi[t] == tim;
57    }
58    ll dfs(int x, ll a)
59    {
60        if (x == t || a == 0)
61            return a;
62        ll flw = 0, f;
63        int &i = cur[x];
64        for (i = first[x]; i != -1; i = next[i])
65        {
66            if (dis[x] + 1 == dis[v[i]] && (
67                f = dfs(v[i], min(a, cap[i]

```

```

        - flow[i])))) > 0)
    {
        flow[i] += f;
        flow[i ^ 1] -= f;
        a -= f;
        flw += f;
        if (a == 0)
            break;
    }
}
return flw;
}
ll MaxFlow(int s, int t)
{
    this->s = s;
    this->t = t;
    ll flw = 0;
    while (bfs())
    {
        for (int i = 1; i <= n; ++i)
            cur[i] = 0;
        flw += dfs(s, oo);
    }
    return flw;
}
};
// MF Net;
// Net.n = n;
// Net.Clear();
// a 到 b (注意從1開始!!!!)
// Net.Add(a, b, w, 0);
// Net.MaxFlow(s, d)
// s 到 d 的 MF

```

## 2 Geometry

### 2.1 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)

```

```

16 {
17     double c = cos(o), s = sin(o);
18     return P(c * x - s * y, s * x + c *
19       y);
20 }
21 double angle() { return atan2(y, x); }
22 };
23 bool operator<(const P &a, const P &b) {
24     return same(a.x, b.x) ? a.y < b.y : a.x
25     < b.x; }
26 bool operator>(const P &a, const P &b) {
27     return same(a.x, b.x) ? a.y > b.y : a.x
28     > b.x; }
29 #define crx(a, b, c) ((b - a) ^ (c - a)) //
30 (向量OA叉積向量OB) > 0 表示從OA到OB為
31 逆時針旋轉。
32 vector<P> convex(vector<P> ps) // Andrew's
33 Monotone Chain
34 {
35     vector<P> p;
36     sort(ps.begin(), ps.end(), [](P &a, P &b
37       )
38       { return a.y < b.y || (a.y == b.y
39         && a.x < b.x); });
40     for (int i = 0; i < ps.size(); ++i)
41     {
42         while (p.size() >= 2 && crx(p[p.size()
43           - 2], ps[i], p[p.size() - 1])
44           >= 0)
45             p.pop_back();
46         p.push_back(ps[i]);
47     }
48     int t = p.size();
49     for (int i = (int)ps.size() - 2; i >= 0;
50       --i)
51     {
52         while (p.size() > t && crx(p[p.size()
53           - 2], ps[i], p[p.size() - 1])
54           >= 0)
55             p.pop_back();
56         p.push_back(ps[i]);
57     }
58     // p.pop_back(); //起點依照題目
59     return p;
60 }

```

## 3 Graph

### 3.1 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 void BellmanFord(int start, int node){

```

```

dist[start] = 0;
for(int it = 0; it < node-1; it++){
    for(int i = 0; i < node; i++){
        for(int j = 0; j < node; j++){
            if(edges[i][j] != -1){
                if(dist[i] + edges[i][j]
                < dist[j]){
                    dist[j] = dist[i] +
                    edges[i][j];
                    ancestor[j] = i;
                }
            }
        }
    }
}
for(int i = 0; i < node; i++) //
negative cycle detection
for(int j = 0; j < node; j++)
if(dist[i] + edges[i][j] < dist[
j])
{
    cout<<"Negative cycle!"<<
    endl;
    return;
}
}
int main(){
    int node;
    cin>>node;
    edges.resize(node, vector<int>(node, inf))
    ;
    dist.resize(node, inf);
    ancestor.resize(node, -1);
    int a, b, d;
    while(cin>>a>>b>>d){
        /*input: source destination weight*/
        if(a == -1 && b == -1 && d == -1)
            break;
        edges[a][b] = d;
    }
    int start;
    cin>>start;
    BellmanFord(start, node);
    return 0;
}

```

### 3.2 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()){
        pass[q.front()] = 1;

```

```

for (int i = 0; i < edges.size(); i
++){
    if(edges[i].first == q.front()
    && pass[edges[i].second] ==
    0){
        p.push(edges[i].second);
        result[edges[i].second] =
        count;
    }
    else if(edges[i].second == q.
    front() && pass[edges[i].
    first] == 0){
        p.push(edges[i].first);
        result[edges[i].first] =
        count;
    }
    else
        newedges.push_back(edges[i])
        ;
}
edges = newedges;
newedges.clear();
q.pop();
if(q.empty() == true){
    q = p;
    queue<int> tmp;
    p = tmp;
    count++;
}
}
int main(){
    int node;
    cin >> node;
    vector<pair<int, int>> edges;
    int a, b;
    while(cin>>a>>b){
        /*a = b = -1 means input edges ended
        */
        if(a == -1 && b == -1)
            break;
        edges.push_back(pair<int, int>(a, b)
        );
    }
    vector<int> result(node, -1);
    BFS(result, edges, node, 0);
    return 0;
}

```

### 3.3 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.
    end(); iter++){

```

```

11     if((*iter).first.first == start &&
12        (*iter).second == 0 && pass[(*)
13           iter).first.second] == 0){
14         route.push_back((*iter).first.
15            second);
16         DFS((*iter).first.second);
17     }
18     else if((*iter).first.second ==
19        start && (*iter).second == 0 &&
20        pass[(*)iter).first.first] == 0){
21         route.push_back((*iter).first.
22            first);
23         DFS((*iter).first.first);
24     }
25 }
26 int main(){
27     int node;
28     cin >> node;
29     pass.resize(node,0);
30     int a,b;
31     while(cin >> a >> b){
32         if(a == -1 && b == -1)
33             break;
34         edges.insert(pair<pair<int,int>,int>
35            >(pair<int,int>(a,b),0));
36     }
37     int start;
38     cin >> start;
39     route.push_back(start);
40     DFS(start);
41     return 0;
42 }

```

### 3.4 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  }

```

```

21     }
22 }
23 int main(){
24     int node;
25     cin >> node;
26     int a,b,d;
27     weight.resize(node,vector<int>(node,-1))
28     ;
29     while(cin >> a >> b >> d){
30         /*input: source destination weight*/
31         if(a == -1 && b == -1 && d == -1)
32             break;
33         weight[a][b] = d;
34     }
35     ancestor.resize(node,-1);
36     dist.resize(node,inf);
37     int start;
38     cin >> start;
39     dist[start] = 0;
40     dijkstra(start);
41     return 0;

```

### 3.5 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);
12     if(rx == ry)
13         return;
14     /*merge by rank -> always merge small
15        tree to big tree*/
16     if(rank[rx] > rank[ry])
17         union_set[ry] = rx;
18     else
19     {
20         union_set[rx] = ry;
21         if(rank[rx] == rank[ry])
22             ++rank[ry];
23     }
24 }
25 int main(){
26     int node;
27     cin >> node; //Input Node number
28     vector<int> union_set(node, 0);
29     vector<int> rank(node, 0);
30     for (int i = 0; i < node; i++)
31         union_set[i] = i;
32     int edge;
33     cin >> edge; //Input Edge number
34     for(int i = 0; i < edge; i++)
35     {
36         int a, b;
37         cin >> a >> b;
38         merge(a, b, union_set,rank);

```

```

36     }
37     /*build party*/
38     vector<vector<int>> > party(node, vector<
39        int>(0));
40     for (int i = 0; i < node; i++)
41         party[find(i, union_set)].push_back(
42            i);

```

## 4 Mathematics

### 4.1 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};
7      long long k = a / b;
8      pair<long long, long long> p = extgcd(b,
9         a - k * b);
10     //cout << p.first << " " << p.second <<
11        endl;
12     //cout << "商數(k)= " << k << endl <<
13        endl;
14     return {p.second, p.first - k * p.second
15        };
16 }
17 int main()
18 {
19     int a, b;
20     cin >> a >> b;
21     pair<long long, long long> xy = extgcd(a,
22        b); //(x0,y0)
23     cout << xy.first << " " << xy.second <<
24        endl;
25     cout << xy.first << " * " << a << " + "
26        << xy.second << " * " << b << endl;
27     return 0;
28 }
29 // ax + by = gcd(a,b) * r
30 /*find |x|+|y| -> min*/
31 int main()
32 {
33     long long r, p, q; /*px+qy = r*/
34     int cases;
35     cin >> cases;
36     while (cases-->0)
37     {
38         cin >> r >> p >> q;
39         pair<long long, long long> xy =
40            extgcd(q, p); //(x0,y0)
41         long long ans = 0, tmp = 0;
42         double k, k1;
43         long long s, s1;
44         k = 1 - (double)(r * xy.first) / p;
45         s = round(k);
46         ans = llabs(r * xy.first + s * p) +
47            llabs(r * xy.second - s * q);

```

```

39         k1 = -(double)(r * xy.first) / p;
40         s1 = round(k1);
41         /*cout << k << endl << k1 << endl;
42            cout << s << endl << s1 << endl;
43            */
44         tmp = llabs(r * xy.first + s1 * p) +
45            llabs(r * xy.second - s1 * q);
46         ans = min(ans, tmp);
47         cout << ans << endl;
48     }
49     return 0;

```

### 4.2 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
6         --)
7      {
8          if (num[i] == '0' && num[i] == '9')
9          {
10             temp += (num[i] - 48) * base;
11             base = base * 16;
12         }
13         else if (num[i] == 'A' && num[i] == 'F'
14            ')
15         {
16             temp += (num[i] - 55) * base;
17             base = base * 16;
18         }
19     }
20     return temp;
21 }
22 void DecToHex(int p_intValue) //10 to 16
23 {
24     char l_pCharRes = new (char);
25     sprintf(l_pCharRes, "%X", p_intValue);
26     int l_intResult = stoi(l_pCharRes);
27     cout << l_pCharRes << endl;
28     return l_intResult;

```

### 4.3 PI

```

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

```

### 4.4 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
11                 += i)
12                 prime[j] = 1;
13         }
14     }
15     /* 0跟1要寫if過濾掉 */
16     // if(!prime[數字])
17     //     我是質數

```

## 4.5 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 }

```

## 4.6 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 +
14                1, ri);
15        if (s[i] == '-' && c == 0)
16            return DFS(le, i - 1) - DFS(i +
17                1, ri);
18    }
19    for (int i = ri; i >= le; i--)
20    {
21        if (s[i] == ')')
22            c++;
23        if (s[i] == '(')
24            c--;

```

```

22     if (s[i] == '*' && c == 0)
23         return DFS(le, i - 1) * DFS(i +
24             1, ri);
25     if (s[i] == '/' && c == 0)
26         return DFS(le, i - 1) / DFS(i +
27             1, ri);
28     if (s[i] == '%' && c == 0)
29         return DFS(le, i - 1) % DFS(i +
30             1, ri);
31 }
32 if ((s[le] == '(' && (s[ri] == ')'))
33     return DFS(le + 1, ri - 1); //去除刮
34     號
35 if (s[le] == '-' && s[ri] == '-')
36     return DFS(le + 1, ri - 1); //去除左
37     右兩邊空格
38 if (s[le] == '(')
39     return DFS(le + 1, ri); //去除左邊空
40     格
41 if (s[ri] == ')')
42     return DFS(le, ri - 1); //去除右邊空
43     格
44 long long int num = 0;
45 for (int i = le; i <= ri; i++)
46     num = num * 10 + s[i] - '0';
47 return num;
48 }

```

## 4.7 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;

```

```

30     finds(2, num, num, zero, ans);
31     // num 為 input 數字
32     for (int i = 0; i < ans.size(); i++)
33     {
34         for (int j = 0; j < ans[i].size() -
35             1; j++)
36             cout << ans[i][j] << " ";
37         cout << ans[i][ans[i].size() - 1] <<
38             endl;
39     }
40 }

```

## 4.8 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() -
30             1; j++)
31             cout << ans[i][j] << " ";
32         cout << ans[i][ans[i].size() - 1] <<
33             endl;
34     }
35 }

```

## 4.9 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 }

```

## 4.10 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 }

```

## 5 Other

### 5.1 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
34 return result;
35 }

```

## 5.2 數獨解法

```

1 int getSquareIndex(int row, int column, int
2     n)
3 {
4     return row / n * n + column / n;
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
14         % n2;
15     if (index >= n2 * n2)
16         return true;
17
18     if (board[rowNum][colNum] != 0)
19         return backtracking(board, rows,
20             cols, boxes, index + 1, n);
21
22     for (int i = 1; i <= n2; i++)
23     {
24         if (!rows[rowNum][i] && !cols[colNum
25             ][i] && !boxes[getSquareIndex(
26                 rowNum, colNum, n)][i])
27         {
28             rows[rowNum][i] = true;
29             cols[colNum][i] = true;
30             boxes[getSquareIndex(rowNum,
31                 colNum, n)][i] = true;
32             board[rowNum][colNum] = i;
33             if (backtracking(board, rows,
34                 cols, boxes, index + 1, n))
35                 return true;
36             board[rowNum][colNum] = 0;
37             rows[rowNum][i] = false;
38             cols[colNum][i] = false;
39             boxes[getSquareIndex(rowNum,
40                 colNum, n)][i] = false;
41         }
42     }
43     return false;
44 }
45 /*用法 main*/

```

```

36 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
37 vector<vector<int>> board(n * n + 1, vector<
38     int>(n * n + 1, 0));
39 vector<vector<bool>> isRow(n * n + 1, vector<
40     bool>(n * n + 1, false));
41 vector<vector<bool>> isColumn(n * n + 1,
42     vector<bool>(n * n + 1, false));
43 vector<vector<bool>> isSquare(n * n + 1,
44     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         if (number == 0)
54             continue;
55         isRow[i][number] = true;
56         isColumn[j][number] = true;
57         isSquare[getSquareIndex(i, j, n)][
58             number] = true;
59     }
60 }
61 if (backtracking(board, isRow, isColumn,
62     isSquare, 0, n))
63     /*有解答*/
64 else
65     /*解答*/

```

## 6 String

### 6.1 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.
24         substr(minStart, minLength);
25 }

```

## 6.2 split

```

1 vector<string> mysplit(const string& str,
2     const string& delim)
3 {
4     vector<string> res;
5     if (" " == str)
6         return res;
7
8     char *strs = new char[str.length() + 1];
9     strcpy(strs, str.c_str());
10
11     char *d = new char[delim.length() + 1];
12     strcpy(d, delim.c_str());
13
14     char *p = strtok(strs, d);
15     while (p)
16     {
17         string s = p;
18         res.push_back(s);
19         p = strtok(NULL, d);
20     }
21     return res;
22 }

```

## 7 data structure

### 7.1 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     }
32 }

```

```

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 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 std::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 }

```



```

87 bool operator>(const Bigint &b) const
88 { return cp3(b)>0; }
89 bool operator>=(const Bigint &b) const
90 { return cp3(b)>=0; }
91 Bigint operator - () const {
92     Bigint r = (*this);
93     r.s = -r.s;
94     return r;
95 }
96 Bigint operator + (const Bigint &b)
97 const {
98     if (s == -1) return -(*this)+(-b);
99     ;
100 if (b.s == -1) return (*this)-(-b);
101 Bigint r;
102 int nl = max(len(), b.len());
103 r.resize(nl + 1);
104 for (int i=0; i<nl; i++) {
105     if (i < len()) r.v[i] += v[i];
106     if (i < b.len()) r.v[i] += b.v[i];
107     ;
108 if (r.v[i] >= BIGMOD) {
109     r.v[i+1] += r.v[i] / BIGMOD;
110     r.v[i] %= BIGMOD;
111 }
112 }
113 r.n();
114 return r;
115 }
116 Bigint operator - (const Bigint &b)
117 const {
118     if (s == -1) return -(*this)-(-b);
119     ;
120 if (b.s == -1) return (*this)+(-b);
121 if ((*this) < b) return -(b-(*this));
122 ;
123 Bigint r;
124 r.resize(len());
125 for (int i=0; i<len(); i++) {
126     r.v[i] += v[i];
127     if (i < b.len()) r.v[i] -= b.v[i];
128     ;
129 if (r.v[i] < 0) {
130     r.v[i] += BIGMOD;
131     r.v[i+1]--;
132 }
133 }
134 r.n();
135 return r;
136 }
137 Bigint operator * (const Bigint &b) {
138     Bigint r;
139     r.resize(len() + b.len() + 1);
140     r.s = s * b.s;
141     for (int i=0; i<len(); i++) {
142         for (int j=0; j<b.len(); j++) {
143             r.v[i+j] += v[i] * b.v[j];
144             if (r.v[i+j] >= BIGMOD) {
145                 r.v[i+j+1] += r.v[i+j] /

```

```

145         BIGMOD;
146         r.v[i+j] %= BIGMOD;
147     }
148     }
149     r.n();
150     return r;
151 }
152 Bigint operator / (const Bigint &b) {
153     Bigint r;
154     r.resize(max(1, len()-b.len()+1));
155     int oriS = s;
156     Bigint b2 = b; // b2 = abs(b)
157     s = b2.s = r.s = 1;
158     for (int i=r.len()-1; i>=0; i--) {
159         int d=0, u=BIGMOD-1;
160         while(d<u) {
161             int m = (d+u+1)>>1;
162             r.v[i] = m;
163             if((r*b2) > (*this)) u = m
164                 -1;
165             else d = m;
166         }
167         r.v[i] = d;
168     }
169     s = oriS;
170     r.s = s * b.s;
171     r.n();
172     return r;
173 }
174 Bigint operator % (const Bigint &b) {
175     return (*this)-(*this)/b*b;
176 }
177 }
178 };

```

## 7.2 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_num = 0;
11         m_den = 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_num = 0;
22             m_den = 0;
23         }
24         else
25         {
26             cout << "This Rational number can be

```

```

27 Rational operator-(const Rational& a); //
28 Rational operator*(const Rational& a); //
29 Rational operator/(const Rational& a); //
30 bool operator==(const Rational& a); //相
31 void reduce(); //化簡
32 private:
33 int m_num;
34 int m_den;
35 };
36 istream &operator>>(istream &input, Rational
37 &test)
38 {
39     char temp;
40     input >> test.m_num;
41     input >> temp;
42     input >> test.m_den;
43     Rational final(test.m_num, test.
44         m_den); //final用來告訴使用者
45         這數不符合!
46     if (test.m_den < 0 || test.
47         m_den == 0) //不符合(再輸入
48         一次)
49     {
50         while (test.m_den < 0 || test.
51             m_den == 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_num;
57         input >> temp;
58         input >> test.m_den;
59         Rational final(test.m_num, test.
60             m_den); //final用來告訴使
61             用者這數不符合!
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_num;
72     if (test.m_num == 0)
73         return output;
74     if (test.m_den == 1)
75         return output;
76     else
77     {
78         output << "/";
79         output << test.m_den;
80     }
81     return output;
82 }
83 Rational Rational::operator+(const Rational&
84     a)

```

```

85 {
86     Rational c;
87     c.m_den = this->m_den * a.
88         m_den; //通分(同乘)
89     c.m_num = (this->m_num * a.
90         m_den) + (a.m_num * this
91         ->m_den);
92     c.reduce();
93     return c;
94 }
95 Rational Rational::operator-(const Rational&
96     a)
97 {
98     Rational c;
99     c.m_den = this->m_den * a.
100        m_den;
101     c.m_num = (this->m_num * a.
102        m_den) - (a.m_num * this
103        ->m_den);
104     c.reduce();
105     return c;
106 }
107 Rational Rational::operator*(const Rational&
108     a)
109 {
110     Rational c;
111     c.m_den = this->m_den * a.
112        m_den;
113     c.m_num = (this->m_num * a.
114        m_den) - (a.m_num * this
115        ->m_den);
116     c.reduce();
117     return c;
118 }
119 Rational Rational::operator/(const Rational&
120     a)
121 {
122     Rational c;
123     c.m_den = this->m_den * a.
124        m_den;
125     c.m_num = (this->m_num * a.
126        m_den) - (a.m_num * this
127        ->m_den);
128     c.reduce();
129     return c;
130 }
131 bool Rational::operator==(const Rational& a)
132 {
133     if (m_num == a.m_num)
134     {
135         if (m_den == a.m_den)
136             return true;
137         else
138             return false;
139     }
140     else
141         return false;
142 }
143 void Rational::reduce()
144 {
145     int i;
146     int max;
147     if (m_num > m_den)
148         max = m_num;
149     else
150         max = m_den;
151     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 | }
```

ACM ICPC  
TEAM  
REFERENCE -  
ANGRY CROW  
TAKES FLIGHT!

Contents

|                             |          |                                       |          |
|-----------------------------|----------|---------------------------------------|----------|
| <b>1 DP</b>                 | <b>1</b> |                                       |          |
| 1.1 KMP . . . . .           | 1        | 3.1 Bellman-Ford . . . . .            | 2        |
| 1.2 LCS . . . . .           | 1        | 3.2 BFS-queue . . . . .               | 2        |
| 1.3 LIC . . . . .           | 1        | 3.3 DFS-rec . . . . .                 | 2        |
| 1.4 Max_subarray . . . . .  | 1        | 3.4 Dijkstra . . . . .                | 3        |
| 1.5 MFlow . . . . .         | 1        | 3.5 union_find . . . . .              | 3        |
| <b>2 Geometry</b>           | <b>2</b> | <b>4 Mathematics</b>                  | <b>3</b> |
| 2.1 Convexhull 2D . . . . . | 2        | 4.1 Extended Euclidean . . . . .      | 3        |
| <b>3 Graph</b>              | <b>2</b> | 4.2 Hex to Dec . . . . .              | 3        |
|                             |          | 4.3 PI . . . . .                      | 3        |
|                             |          | 4.4 Prime table . . . . .             | 3        |
|                             |          | 4.5 二分逼近法 . . . . .                   | 4        |
|                             |          | 4.6 四則運算 . . . . .                    | 4        |
|                             |          | 4.7 數字乘法組合 . . . . .                  | 4        |
|                             |          | 4.8 數字加法組合 . . . . .                  | 4        |
|                             |          | 4.9 羅馬數字 . . . . .                    | 4        |
|                             |          | 4.10 質因數分解 . . . . .                  | 4        |
|                             |          | <b>5 Other</b>                        | <b>4</b> |
|                             |          | 5.1 Weighted Job Scheduling . . . . . | 4        |
|                             |          | 5.2 數獨解法 . . . . .                    | 5        |
|                             |          | <b>6 String</b>                       | <b>5</b> |
|                             |          | 6.1 sliding window . . . . .          | 5        |
|                             |          | 6.2 split . . . . .                   | 5        |
|                             |          | <b>7 data structure</b>               | <b>5</b> |
|                             |          | 7.1 Bigint . . . . .                  | 5        |
|                             |          | 7.2 分數 . . . . .                      | 6        |