1 DP

1.1 KMP

```cpp
void ComputePrefix(string s, int next[])
{
    int n = s.length();
    int q, k;
    next[0] = 0;
    for (k = 0, q = 1; q < n; q++)
    {
        while (k > 0 && s[k] != s[q])
            k = next[k];
        if (s[k] == s[q])
            k++;
        next[q] = k;
    }
}
void KMPMatcher(string text, string pattern)
{
    int n = text.length();
    int m = pattern.length();
    int next[pattern.length()];
    ComputePrefix(pattern, next);

    for (int i = 0, q = 0; i < n; i++)
    {
        while (q > 0 && pattern[q] != text[i
            ])
            q = next[q];
        if (pattern[q] == text[i])
            q++;
        if (q == m)
        {
            cout << "Pattern occurs with
                shift " << i - m + 1 << endl
                ;
            q = 0;
        }
    }
}
string s = "abcdabcdebcd";
string p = "bcd";
KMPMatcher(s, p);
cout << endl;
return 0;
```

1.2 LCS

```cpp
int LCS(vector<int> Ans, vector<int> num) //
    Ans 跟 num 都要 index 從1開始放
{
    vector<vector<int>> LCS(N + 1, vector<
        int>(N + 1, 0));
    for (int i = 1; i <= N; ++i)
    {
        for (int j = 1; j <= N; ++j)
        {
            if (Ans[i] == num[j])
```

```cpp
                LCS[i][j] = LCS[i - 1][j -
                    1] + 1;
            else
                LCS[i][j] = max(LCS[i - 1][j
                    ], LCS[i][j - 1]);
        }
    }
    // printf("%d\n", LCS[N][N]);
    return LCS[N][N];
    //列印 LCS
    vector<int> k;
    while (n && m)
    {
        if (LCS[n][m] != max(LCS[n - 1][m],
            LCS[n][m - 1]))
        {
            k.push_back(arr1[n]);
            n--;
            m--;
        }
        else if (LCS[n][m] == LCS[n - 1][m])
            n--;
        else if (LCS[n][m] == LCS[n][m - 1])
            m--;
    }

    reverse(k.begin(), k.end());
}
```

1.3 LIC

```cpp
void getMaxElementAndPos(vector<int> &LISTbl
    , vector<int> &LISLen, int tNum,
    int tlen, int tStart, int &num, int &pos
    )
{
    int max = numeric_limits<int>::min();
    int maxPos;
    for (int i = tStart; i >= 0; i--)
    {
        if (LISLen[i] == tlen && LISTbl[i] <
            tNum)
        {
            if (LISTbl[i] > max)
            {
                max = LISTbl[i];
                maxPos = i;
            }
        }
    }
    num = max;
    pos = maxPos;
}
int LIS(vector<int> &LISTbl)
{
    if (LISTbl.size() == 0)
        return 0;
    vector<int> LISLen(LISTbl.size(), 1);
    for (int i = 1; i < LISTbl.size(); i++)
    {
        for (int j = 0; j < i; j++)
        {
            if (LISTbl[j] < LISTbl[i])
```

```cpp
                LISLen[i] = max(LISLen[i],
                    LISLen[j] + 1);
        }
    }

    int maxlen = *max_element(LISLen.begin()
        , LISLen.end());
    int num, pos;
    vector<int> buf;
    getMaxElementAndPos(LISTbl, LISLen,
        numeric_limits<int
            >::max(),
        maxlen, LISTbl.size
            () - 1, num, pos
        );
    buf.push_back(num);
    for (int len = maxlen - 1; len >= 1; len
        --)
    {
        int tnum = num;
        int tpos = pos;
        getMaxElementAndPos(LISTbl, LISLen,
            tnum, len, tpos
                - 1, num,
                pos);
        buf.push_back(num);
    }
    reverse(buf.begin(), buf.end());
    for (int k = 0; k < buf.size(); k++) //
        列印
    {
        if (k == buf.size() - 1)
            cout << buf[k] << endl;
        else
            cout << buf[k] << ",";
    }
    return maxlen;
}
```

1.4 Max_subarray

```cpp
/*Kadane's algorithm*/
int maxSubArray(vector<int>& nums) {
    int local_max = nums[0], global_max =
        nums[0];
    for(int i = 1; i < nums.size(); i++){
        local_max = max(nums[i],nums[i]+
            local_max);
        global_max = max(local_max,
            global_max);
    }
    return global_max;
}
```

1.5 MFlow

```cpp
typedef long long ll;
struct MF
{
```

```cpp
static const int N = 5000 + 5;
static const int M = 60000 + 5;
static const ll oo = 10000000000000LL;

int n, m, s, t, tot, tim;
int first[N], next[M];
int u[M], v[M], cur[N], vi[N];
ll cap[M], flow[M], dis[N];
int que[N + N];

void Clear()
{
    tot = 0;
    tim = 0;
    for (int i = 1; i <= n; ++i)
        first[i] = -1;
}
void Add(int from, int to, ll cp, ll flw
    )
{
    u[tot] = from;
    v[tot] = to;
    cap[tot] = cp;
    flow[tot] = flw;
    next[tot] = first[u[tot]];
    first[u[tot]] = tot;
    ++tot;
}
bool bfs()
{
    ++tim;
    dis[s] = 0;
    vi[s] = tim;

    int head, tail;
    head = tail = 1;
    que[head] = s;
    while (head <= tail)
    {
        for (int i = first[que[head]]; i
            != -1; i = next[i])
        {
            if (vi[v[i]] != tim && cap[i
                ] > flow[i])
            {
                vi[v[i]] = tim;
                dis[v[i]] = dis[que[head
                    ]] + 1;
                que[++tail] = v[i];
            }
        }
        ++head;
    }
    return vi[t] == tim;
}
ll dfs(int x, ll a)
{
    if (x == t || a == 0)
        return a;
    ll flw = 0, f;
    int &i = cur[x];
    for (i = first[x]; i != -1; i = next
        [i])
    {
        if (dis[x] + 1 == dis[v[i]] && (
            f = dfs(v[i], min(a, cap[i]
```

```cpp
64                    - flow[i]))) > 0)
65            {
66                flow[i] += f;
67                flow[i ^ 1] -= f;
68                a -= f;
69                flw += f;
70                if (a == 0)
71                    break;
72            }
73        }
74        return flw;
75    }
76    ll MaxFlow(int s, int t)
77    {
78        this->s = s;
79        this->t = t;
80        ll flw = 0;
81        while (bfs())
82        {
83            for (int i = 1; i <= n; ++i)
84                cur[i] = 0;
85            flw += dfs(s, oo);
86        }
87        return flw;
88    }
89 };
90 // MF Net;
91 // Net.n = n;
92 // Net.Clear();
93 // a 到 b (注意從1開始!!!!)
94 // Net.Add(a, b, w, 0);
95 // Net.MaxFlow(s, d)
   // s 到 d 的 MF
```

2 Geometry

2.1 Convexhull 2D

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

3 Graph

3.1 Bellman-Ford

```cpp
1 /*SPA - Bellman-Ford*/
2 #include<bits/stdc++.h>
3 #define inf 99999 //define by you maximum
      edges weight
4 using namespace std;
5 vector<vector<int> > edges;
6 vector<int> dist;
7 vector<int> ancestor;
8 void BellmanFord(int start,int node){
9     dist[start] = 0;
10    for(int it = 0; it < node-1; it++){
11        for(int i = 0; i < node; i++){
12            for(int j = 0; j < node; j++){
13                if(edges[i][j] != -1){
14                    if(dist[i] + edges[i][j]
                          < dist[j]){
15                        dist[j] = dist[i] +
                              edges[i][j];
16                        ancestor[j] = i;
17                    }
18                }
19            }
20        }
21    }
22
23    for(int i = 0; i < node; i++)  //
          negative cycle detection
24        for(int j = 0; j < node; j++)
25            if(dist[i] + edges[i][j] < dist[
                 j])
26            {
27                cout<<"Negative cycle!"<<
                     endl;
28                return;
29            }
30 }
31 int main(){
32    int node;
33    cin>>node;
34    edges.resize(node,vector<int>(node,inf))
          ;
35    dist.resize(node,inf);
36    ancestor.resize(node,-1);
37    int a,b,d;
38    while(cin>>a>>b>>d){
39        /*input: source destination weight*/
40        if(a == -1 && b == -1 && d == -1)
41            break;
42        edges[a][b] = d;
43    }
44    int start;
45    cin>>start;
46    BellmanFord(start,node);
47    return 0;
48 }
```

3.2 BFS-queue

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

3.3 DFS-rec

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

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<
           pair<int,int> > ,greater<pair<int,
           int> > > pq;
10     pq.push(make_pair(0,start));
11     while(!pq.empty()){
12         int cur = pq.top().second;
13         pq.pop();
14         for(int i = 0; i < weight[cur].size
               (); i++){
15             if(dist[i] > dist[cur] + weight[
                   cur][i] && weight[cur][i] !=
                   -1){
16                 dist[i] = dist[cur] + weight
                       [cur][i];
17                 ancestor[i] = cur;
18                 pq.push(make_pair(dist[i],i)
                       );
19             }
20         }
```

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

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],
               union_set); //compress path
4      return union_set[x];
5  }
6  void merge(int x,int y,vector<int> &
       union_set,vector<int> &rank){
7      int rx, ry;
8      rx = find(x,union_set);
9      ry = find(y,union_set);
10     if(rx == ry)
11         return;
12     /*merge by rank -> always merge small
           tree to big tree*/
13     if(rank[rx] > rank[ry])
14         union_set[ry] = rx;
15     else
16     {
17         union_set[rx] = ry;
18         if(rank[rx] == rank[ry])
19             ++rank[ry];
20     }
21 }
22 int main(){
23     int node;
24     cin >> node; //Input Node number
25     vector<int> union_set(node, 0);
26     vector<int> rank(node, 0);
27     for (int i = 0; i < node; i++)
28         union_set[i] = i;
29     int edge;
30     cin >> edge; //Input Edge number
31     for(int i = 0; i < edge; i++)
32     {
33         int a, b;
34         cin >> a >> b;
35         merge(a, b, union_set,rank);
```

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

4 Mathematics

4.1 Extended Euclidean

```
1  // ax + by = gcd(a,b)
2  pair<long long, long long> extgcd(long long
       a, long long b)
3  {
4      if (b == 0)
5          return {1, 0};
6      long long k = a / b;
7      pair<long long, long long> p = extgcd(b,
           a - k * b);
8      //cout << p.first << " " << p.second <<
           endl;
9      //cout << "商數(k)=  " << k << endl <<
           endl;
10     return {p.second, p.first - k * p.second
           };
11 }
12
13 int main()
14 {
15     int a, b;
16     cin >> a >> b;
17     pair<long long, long long> xy = extgcd(a
           , b); //(x0,y0)
18     cout << xy.first << " " << xy.second <<
           endl;
19     cout << xy.first << " * " << a << " + "
           << xy.second << " * " << b << endl;
20     return 0;
21 }
22 // ax + by = gcd(a,b) * r
23 /*find |x|+|y| -> min*/
24 int main()
25 {
26     long long r, p, q; /*px+qy = r*/
27     int cases;
28     cin >> cases;
29     while (cases--)
30     {
31         cin >> r >> p >> q;
32         pair<long long, long long> xy =
               extgcd(q, p); //(x0,y0)
33         long long ans = 0, tmp = 0;
34         double k, k1;
35         long long s, s1;
36         k = 1 - (double)(r * xy.first) / p;
37         s = round(k);
38         ans = llabs(r * xy.first + s * p) +
               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         tmp = llabs(r * xy.first + s1 * p) +
               llabs(r * xy.second - s1 * q);
44         ans = min(ans, tmp);
45
46         cout << ans << endl;
47     }
48     return 0;
49 }
```

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          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 n;
26     return l_intResult;
27 }
```

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
                   += i)
11                 prime[j] = 1;
12         }
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,
       string final index)
3  {
4      int c = 0;
5      for (int i = ri; i >= le; i--)
6      {
7          if (s[i] == ')')
8              c++;
9          if (s[i] == '(')
10             c--;
11         if (s[i] == '+' && c == 0)
12             return DFS(le, i - 1) + DFS(i +
                   1, ri);
13         if (s[i] == '-' && c == 0)
14             return DFS(le, i - 1) - DFS(i +
                   1, ri);
15     }
16     for (int i = ri; i >= le; i--)
17     {
18         if (s[i] == ')')
19             c++;
20         if (s[i] == '(')
21             c--;
22         if (s[i] == '*' && c == 0)
23             return DFS(le, i - 1) * DFS(i +
                   1, ri);
24         if (s[i] == '/' && c == 0)
25             return DFS(le, i - 1) / DFS(i +
                   1, ri);
26         if (s[i] == '%' && c == 0)
27             return DFS(le, i - 1) % DFS(i +
                   1, ri);
28     }
29     if ((s[le] == '(') && (s[ri] == ')'))
30         return DFS(le + 1, ri - 1); //去除刮
                號
31     if (s[le] == ' ' && s[ri] == ' ')
32         return DFS(le + 1, ri - 1); //去除左
                右兩邊空格
33     if (s[le] == ' ')
34         return DFS(le + 1, ri); //去除左邊空
                格
35     if (s[ri] == ' ')
36         return DFS(le, ri - 1); //去除右邊空
                格
37     long long int num = 0;
38     for (int i = le; i <= ri; i++)
39         num = num * 10 + s[i] - '0';
40     return num;
41 }
```

4.7 數字乘法組合

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

4.8 數字加法組合

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

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

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.
          substr(minStart, minLength);
24 }
```

6.2　split

```
1  vector<string> mysplit(const string& str,
      const string& delim)
2  {
3      vector<string> res;
4      if ("" == str)
5          return res;
6
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 }
```

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         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     void resize(int nl) {
51         vl = nl; //v.resize(nl);
52         fill(v, v+vl, 0); //fill(ALL(v), 0);
53     }
54     void print() const {
55         if (empty()) { putchar('0'); return;
                }
56         if (s == -1) putchar('-');
57         printf("%d", back());
58         for (int i=len()-2; i>=0; i--)
59             printf("%.4d",v[i]);
60     }
61     friend std::ostream& operator << (std::
          ostream& out, const Bigint &a) {
62         if (a.empty()) { out << "0"; return
              out; }
63         if (a.s == -1) out << "-";
64         out << a.back();
65         for (int i=a.len()-2; i>=0; i--) {
66             char str[10];
67             snprintf(str, 5, "%.4d", a.v[i])
                  ;
68             out << str;
69         }
70         return out;
71     }
72     int cp3(const Bigint &b)const {
73         if (s != b.s) return s - b.s;
74         if (s == -1) return -(-*this).cp3(-b
              );
75         if (len() != b.len()) return len()-b
              .len();//int
76         for (int i=len()-1; i>=0; i--)
77             if (v[i]!=b.v[i]) return v[i]-b.
                  v[i];
78         return 0;
79     }
80     bool operator<(const Bigint &b)const
          { return cp3(b)<0; }
81     bool operator<=(const Bigint &b)const
          { return cp3(b)<=0; }
82     bool operator==(const Bigint &b)const
          { return cp3(b)==0; }
83     bool operator!=(const Bigint &b)const
          { return cp3(b)!=0; }
```

```cpp
    bool operator>(const Bigint &b)const
    { return cp3(b)>0; }
    bool operator>=(const Bigint &b)const
    { return cp3(b)>=0; }
    Bigint operator - () const {
        Bigint r = (*this);
        r.s = -r.s;
        return r;
    }
    Bigint operator + (const Bigint &b)
        const {
        if (s == -1) return -(-(*this)+(-b))
            ;
        if (b.s == -1) return (*this)-(-b);
        Bigint r;
        int nl = max(len(), b.len());
        r.resize(nl + 1);
        for (int i=0; i<nl; i++) {
            if (i < len()) r.v[i] += v[i];
            if (i < b.len()) r.v[i] += b.v[i
                ];
            if(r.v[i] >= BIGMOD) {
                r.v[i+1] += r.v[i] / BIGMOD;
                r.v[i] %= BIGMOD;
            }
        }
        r.n();
        return r;
    }
    Bigint operator - (const Bigint &b)
        const {
        if (s == -1) return -(-(*this)-(-b))
            ;
        if (b.s == -1) return (*this)+(-b);
        if ((*this) < b) return -(b-(*this))
            ;
        Bigint r;
        r.resize(len());
        for (int i=0; i<len(); i++) {
            r.v[i] += v[i];
            if (i < b.len()) r.v[i] -= b.v[i
                ];
            if (r.v[i] < 0) {
                r.v[i] += BIGMOD;
                r.v[i+1]--;
            }
        }
        r.n();
        return r;
    }
    Bigint operator * (const Bigint &b) {
        Bigint r;
        r.resize(len() + b.len() + 1);
        r.s = s * b.s;
        for (int i=0; i<len(); i++) {
            for (int j=0; j<b.len(); j++) {
                r.v[i+j] += v[i] * b.v[j];
                if(r.v[i+j] >= BIGMOD) {
                    r.v[i+j+1] += r.v[i+j] /
                        BIGMOD;
                    r.v[i+j] %= BIGMOD;
                }
            }
        }
        r.n();
        return r;
    }
    Bigint operator / (const Bigint &b) {
        Bigint r;
        r.resize(max(1, len()-b.len()+1));
        int oriS = s;
        Bigint b2 = b; // b2 = abs(b)
        s = b2.s = r.s = 1;
        for (int i=r.len()-1; i>=0; i--) {
            int d=0, u=BIGMOD-1;
            while(d<u) {
                int m = (d+u+1)>>1;
                r.v[i] = m;
                if((r*b2) > (*this)) u = m
                    -1;
                else d = m;
            }
            r.v[i] = d;
        }
        s = oriS;
        r.s = s * b.s;
        r.n();
        return r;
    }
    Bigint operator % (const Bigint &b) {
        return (*this)-(*this)/b*b;
    }
};
```

7.2　分數

```cpp
class Rational
{
    friend istream &operator>>(istream &,
        Rational & );
    friend ostream &operator<<(ostream &,
        const Rational & );
public:
    Rational()   //constructor one
    {
        m_numeitor = 0;
        m_denominator = 1;
    }
    Rational(int a, int b)   //constructor two
    {
        if (b < 0 || b == 0)   //avoids negative
            denominators. && prevents a 0
            denominator
        {
            cout << "This Rational number can't be
                used.\n\n";
            m_numeitor = 0;
            m_denominator = 0;
        }
        else
        {
            cout << "This Rational number can be
                used.\n\n";
            m_numeitor = a;
            m_denominator = b;
        }
    }
    Rational operator+(const Rational& a);   //加
    Rational operator-(const Rational& a);   //減
    Rational operator*(const Rational& a);   //乘
    Rational operator/(const Rational& a);   //除
    bool operator==(const Rational& a);   //相等
    void reduce();   //化簡
private:
    int m_numeitor;
    int m_denominator;
};
istream &operator>>(istream &input, Rational
    &test )
{
    char temp;

    input >> test.m_numeitor;
    input >> temp;
    input >> test.m_denominator;
    Rational final(test.m_numeitor, test.
        m_denominator);   //final用來告訴使用者
        這數字符不符合!
    if (test.m_denominator < 0 || test.
        m_denominator == 0)   //不符合(再輸入一次)
    {
        while (test.m_denominator < 0 || test.
            m_denominator == 0)   //有可能輸入的
            東西還是不符合,所以用迴圈
        {
            cout << "Enter another Rational number
                (n/d): ";
            input >> test.m_numeitor;
            input >> temp;
            input >> test.m_denominator;
            Rational final(test.m_numeitor, test.
                m_denominator);   //final用來告訴使
                用者這數字符不符合!
        }
        return input;
    }
    else
        return input;
}
ostream &operator<<(ostream &output, const
    Rational &test )
{
    output << test.m_numeitor;
    if(test.m_numeitor == 0)
        return output;
    if (test.m_denominator == 1)
        return output;
    else
    {
        output << "/";
        output << test.m_denominator;
    }
    return output;
}
Rational Rational::operator+(const Rational&
    a)
{
    Rational c;
    c.m_denominator = this->m_denominator * a.
        m_denominator;   //通分(同乘)
    c.m_numeitor = (this->m_numeitor * a.
        m_denominator) + (a.m_numeitor * this
        ->m_denominator);
    c.reduce();
    return c;
}
Rational Rational::operator-(const Rational&
    a)
{
    Rational c;
    c.m_denominator = this->m_denominator * a.
        m_denominator;
    c.m_numeitor = (this->m_numeitor * a.
        m_denominator) - (a.m_numeitor * this
        ->m_denominator);
    c.reduce();
    return c;
}
Rational Rational::operator*(const Rational&
    a)
{
    Rational c;
    c.m_denominator = this->m_denominator * a.
        m_denominator;
    c.m_numeitor = this->m_numeitor * a.
        m_numeitor;
    c.reduce();
    return c;
}
Rational Rational::operator/(const Rational&
    a)
{
    Rational c;
    c.m_denominator = this->m_denominator * a.
        m_numeitor;
    c.m_numeitor = this->m_numeitor * a.
        m_denominator;
    c.reduce();
    return c;
}
bool Rational::operator==(const Rational& a)
{
    if (m_numeitor == a.m_numeitor)
    {
        if (m_denominator == a.m_denominator)
            return true;
        else
            return false;
    }
    else
        return false;
}
void Rational::reduce()
{
    int i;
    int max;
    if(m_numeitor> m_denominator)
        max = m_numeitor;
    else
        max = m_denominator;
    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