

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
6   9223372036854775807)
7 unsigned long long (0 to
8   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 // string s = "abcdabcdebc";
35 // string p = "bcd";
36 // KMPMatcher(s, p);
37 // cout << endl;
```

2.2 Knapsack Bounded

```
1 const int N = 100, W = 100000;
2 int cost[N], weight[N], number[N];
3 int c[W + 1];
4 void knapsack(int n, int w)
5 {
6     for (int i = 0; i < n; ++i)
7     {
8         int num = min(number[i], w / weight[i]);
9         for (int k = 1; num > 0; k *= 2)
10        {
11            if (k > num)
12                k = num;
13            num -= k;
14            for (int j = w; j >= weight[i] * k; --j)
15                c[j] = max(c[j], c[j - weight[i] * k] + cost[i] * k);
16        }
17    }
18    cout << "Max Prince" << c[w];
19 }
```

2.3 Knapsack sample

```
1 int Knapsack(vector<int> weight, vector<int> value, int bag_Weight)
2 {
3     // vector<int> weight = {1, 3, 4};
4     // vector<int> value = {15, 20, 30};
5     // int bagWeight = 4;
6     vector<vector<int>> dp(weight.size(), vector<int>(bagWeight + 1, 0));
7     for (int j = weight[0]; j <= bagWeight; j++)
8         dp[0][j] = value[0];
9     // weight數組的大小就是物品個數
10    for (int i = 1; i < weight.size(); i++)
11    { // 遍歷物品
12        for (int j = 0; j <= bagWeight; j++)
13        { // 遍歷背包容量
14            if (j < weight[i]) dp[i][j] = dp[i - 1][j];
15            else dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - weight[i]] + value[i]);
16        }
17    }
```

```
17 }
18 cout << dp[weight.size() - 1][bagWeight] << endl;
19 }
```

2.4 Knapsack Unbounded

```
1 const int N = 100, W = 100000;
2 int cost[N], weight[N];
3 int c[W + 1];
4 void knapsack(int n, int w)
5 {
6     memset(c, 0, sizeof(c));
7     for (int i = 0; i < n; ++i)
8         for (int j = weight[i]; j <= w; ++j)
9             c[j] = max(c[j], c[j - weight[i]] + cost[i]);
10    cout << "最高的價值為" << c[w];
11 }
```

2.5 LCIS

```
1 int LCIS_len(vector<int> arr1, vector<int> arr2)
2 {
3     int n = arr1.size(), m = arr2.size();
4     vector<int> table(m, 0);
5     for (int j = 0; j < m; j++)
6         table[j] = 0;
7     for (int i = 0; i < n; i++)
8     {
9         int current = 0;
10        for (int j = 0; j < m; j++)
11        {
12            if (arr1[i] == arr2[j])
13                if (current + 1 > table[j])
14                    table[j] = current + 1;
15
16            if (arr1[i] > arr2[j])
17                if (table[j] > current)
18                    current = table[j];
19        }
20    }
21    int result = 0;
22    for (int i = 0; i < m; i++)
23        if (table[i] > result)
24            result = table[i];
25    return result;
26 }
27 }
```

2.6 LCS

```
1 int LCS(vector<string> Ans, vector<string> num)
2 {
```

```
3     int N = Ans.size(), M = num.size();
4     vector<vector<int>> LCS(N + 1, vector<int>(M + 1, 0));
5     for (int i = 1; i <= N; ++i)
6     {
7         for (int j = 1; j <= M; ++j)
8         {
9             if (Ans[i - 1] == num[j - 1])
10                LCS[i][j] = LCS[i - 1][j - 1] + 1;
11            else
12                LCS[i][j] = max(LCS[i - 1][j], LCS[i][j - 1]);
13        }
14    }
15    cout << LCS[N][M] << '\n';
16    //列印 LCS
17    int n = N, m = M;
18    vector<string> k;
19    while (n && m)
20    {
21        if (LCS[n][m] != max(LCS[n - 1][m], LCS[n][m - 1]))
22        {
23            k.push_back(Ans[n - 1]);
24            n--;
25            m--;
26        }
27        else if (LCS[n][m] == LCS[n - 1][m])
28            n--;
29        else if (LCS[n][m] == LCS[n][m - 1])
30            m--;
31    }
32    reverse(k.begin(), k.end());
33    for (auto i : k)
34        cout << i << " ";
35    cout << endl;
36    return LCS[N][M];
37 }
```

2.7 LIS

```
1 void getMaxElementAndPos(vector<int> &LISTbl, vector<int> &LISLen, int tNum, int tlen, int tStart, int &num, int &pos)
2 {
3     int max = numeric_limits<int>::min();
4     int maxPos;
5     for (int i = tStart; i >= 0; i--)
6     {
7         if (LISLen[i] == tlen && LISTbl[i] < tNum)
8         {
9             if (LISTbl[i] > max)
10            {
11                max = LISTbl[i];
12                maxPos = i;
13            }
14        }
15    }
16    num = max;
17    pos = maxPos;
18 }
```

```

19 int LIS(vector<int> &LISTbl)
20 {
21     if (LISTbl.size() == 0)
22         return 0;
23     vector<int> LISlen(LISTbl.size(), 1);
24     for (int i = 1; i < LISTbl.size(); i++)
25         for (int j = 0; j < i; j++)
26             if (LISTbl[j] < LISTbl[i])
27                 LISlen[i] = max(LISlen[i],
28                                 LISlen[j] + 1);
29     int maxlen = *max_element(LISlen.begin(),
30                               LISlen.end());
31     int num, pos;
32     vector<int> buf;
33     getMaxElementAndPos(LISTbl, LISlen,
34                          numeric_limits<int>::max(), maxlen,
35                          LISTbl.size() - 1, num, pos);
36     buf.push_back(num);
37     for (int len = maxlen - 1; len >= 1; len --)
38     {
39         int tnum = num;
40         int tpos = pos;
41         getMaxElementAndPos(LISTbl, LISlen,
42                              tnum, len, tpos - 1, num, pos);
43         buf.push_back(num);
44     }
45     reverse(buf.begin(), buf.end());
46     for (int k = 0; k < buf.size(); k++) // 列印
47     {
48         if (k == buf.size() - 1)
49             cout << buf[k] << endl;
50         else
51             cout << buf[k] << ",";
52     }
53     return maxlen;
54 }

```

2.8 LPS

```

1 void LPS(string s)
2 {
3     int maxlen = 0, l, r;
4     int n = s.size();
5     for (int i = 0; i < n; i++)
6     {
7         int x = 0;
8         while ((s[i - x] == s[i + x]) && (i - x >= 0) && (i + x < n)) //odd length
9             x++;
10        x--;
11        if (2 * x + 1 > maxlen)
12        {
13            maxlen = 2 * x + 1;
14            l = i - x;
15            r = i + x;
16        }
17        x = 0;
18        while ((s[i - x] == s[i + 1 + x]) && (i - x >= 0) && (i + 1 + x < n)) //even length
19            x++;
20    }
21    return maxlen;
22 }

```

```

19     x++;
20     if (2 * x > maxlen)
21     {
22         maxlen = 2 * x;
23         l = i - x + 1;
24         r = i + x;
25     }
26 }
27 cout << maxlen << '\n'; // 最後長度
28 cout << l + 1 << ' ' << r + 1 << '\n'; // 頭到尾
29 }

```

2.9 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], local_max +
7                           nums[i]);
8         global_max = max(local_max,
9                           global_max);
10    }
11    return global_max;
12 }

```

2.10 Money problem

```

1 //能否湊得某個價位
2 void change(vector<int> price, int limit)
3 {
4     vector<bool> c(limit + 1, 0);
5     c[0] = true;
6     for (int i = 0; i < price.size(); ++i)
7         for (int j = price[i]; j <= limit; ++j) // 依序加入各種面額
8             c[j] = c[j] | c[j - price[i]];
9     if (c[limit]) cout << "YES\n";
10    else cout << "NO\n";
11 }
12 // 湊得某個價位的湊法總共幾種
13 void change(vector<int> price, int limit)
14 {
15     vector<int> c(limit + 1, 0);
16     c[0] = true;
17     for (int i = 0; i < price.size(); ++i)
18         for (int j = price[i]; j <= limit; ++j)
19             c[j] += c[j - price[i]];
20     cout << c[limit] << '\n';
21 }
22 // 湊得某個價位的最少錢幣用量
23 void change(vector<int> price, int limit)
24 {
25     vector<int> c(limit + 1, 0);
26     c[0] = true;
27     for (int i = 0; i < price.size(); ++i)
28         for (int j = price[i]; j <= limit; ++j)
29             c[j] = min(c[j], c[j - price[i]] + 1);
30     cout << c[limit] << '\n';
31 }

```

```

25 vector<int> c(limit + 1, 0);
26 c[0] = true;
27 for (int i = 0; i < price.size(); ++i)
28     for (int j = price[i]; j <= limit; ++j)
29         c[j] = min(c[j], c[j - price[i]] + 1);
30 cout << c[limit] << '\n';
31 }
32 //湊得某個價位的錢幣用量，有幾種可能性
33 void change(vector<int> price, int limit)
34 {
35     vector<int> c(limit + 1, 0);
36     c[0] = true;
37     for (int i = 0; i < price.size(); ++i)
38         for (int j = price[i]; j <= limit; ++j)
39             c[j] |= c[j - price[i]] << 1; // 錢幣數量加一，每一種可能性都加一。
40     for (int i = 1; i <= limit; ++i)
41         if (c[i] && (1 <= i))
42             cout << "用 " << i << " 個錢幣可湊得價位 " << i;
43     }
44 }

```

3 Geometry

3.1 Line

```

1 template <typename T>
2 struct line
3 {
4     line() {}
5     point<T> p1, p2;
6     T a, b, c; //ax+by+c=0
7     line(const point<T> &p1, const point<T> &p2) : p1(p1), p2(p2) {}
8     void pton()
9     { //轉成一般式
10         a = p1.y - p2.y;
11         b = p2.x - p1.x;
12         c = -a * p1.x - b * p1.y;
13     }
14     T ori(const point<T> &p) const
15     { //點和有向直線的關係，>0左邊，=0在線上，<0右邊
16         return (p2 - p1).cross(p - p1);
17     }
18     T btw(const point<T> &p) const
19     { //點投影落在線段上 <=0
20         return (p1 - p).dot(p2 - p1);
21     }
22     bool point_on_segment(const point<T> &p) const
23     { //點是否在線段上
24         return ori(p) == 0 && btw(p) <= 0;
25     }
26 }

```

```

27 T dis2(const point<T> &p, bool is_segment = 0) const
28 { //點跟直線/線段的距離平方
29     point<T> v = p2 - p1, v1 = p - p1;
30     if (is_segment)
31     {
32         point<T> v2 = p - p2;
33         if (v.dot(v1) <= 0)
34             return v1.abs2();
35         if (v.dot(v2) >= 0)
36             return v2.abs2();
37     }
38     T tmp = v.cross(v1);
39     return tmp * tmp / v.abs2();
40 }
41 T seg_dis2(const line<T> &l) const
42 { //兩線段距離平方
43     return min({dis2(l.p1, 1), dis2(l.p2, 1), l.dis2(p1, 1), l.dis2(p2, 1)});
44 }
45 point<T> projection(const point<T> &p) const
46 { //點對直線的投影
47     point<T> n = (p2 - p1).normal();
48     return p - n * (p - p1).dot(n) / n.abs2();
49 }
50 point<T> mirror(const point<T> &p) const
51 { //點對直線的鏡射，要先呼叫pton轉成一般式
52     point<T> R;
53     T d = a * a + b * b;
54     R.x = (b * b * p.x - a * a * p.x - 2 * a * b * p.y - 2 * a * c) / d;
55     R.y = (a * a * p.y - b * b * p.y - 2 * a * b * p.x - 2 * b * c) / d;
56     return R;
57 }
58 bool equal(const line &l) const
59 { //直線相等
60     return ori(l.p1) == 0 && ori(l.p2) == 0;
61 }
62 bool parallel(const line &l) const
63 {
64     return (p1 - p2).cross(l.p1 - l.p2) == 0;
65 }
66 bool cross_seg(const line &l) const
67 {
68     return (p2 - p1).cross(l.p1 - p1) * (p2 - p1).cross(l.p2 - p1) <= 0;
69     //直線是否交線段
70 }
71 int line_intersect(const line &l) const
72 { //直線相交情況，-1無限多點，1交於一點，0不相交
73     return parallel(l) ? (ori(l.p1) == 0 ? -1 : 0) : 1;
74 }
75 int seg_intersect(const line &l) const
76 {
77     return parallel(l) ? (ori(l.p1) == 0 ? -1 : 0) : 1;
78 }

```

```

76 T c1 = ori(l.p1), c2 = ori(l.p2);
77 T c3 = l.ori(p1), c4 = l.ori(p2);
78 if (c1 == 0 && c2 == 0)
79 { //共線
80     bool b1 = btw(l.p1) >= 0, b2 =
81         btw(l.p2) >= 0;
82     T a3 = l.btw(p1), a4 = l.btw(p2);
83     ;
84     if (b1 && b2 && a3 == 0 && a4 >=
85         0)
86         return 2;
87     if (b1 && b2 && a3 >= 0 && a4 ==
88         0)
89         return 3;
90     if (b1 && b2 && a3 >= 0 && a4 >=
91         0)
92         return 0;
93     return -1; //無限交點
94 }
95 else if (c1 * c2 <= 0 && c3 * c4 <=
96     0)
97     return 1;
98     return 0; //不相交
99 }
100 point<T> line_intersection(const line &l
101     ) const
102 { /*直線交點*/
103     point<T> a = p2 - p1, b = l.p2 - l.
104     p1, s = l.p1 - p1;
105     //if(a.cross(b)==0)return INF;
106     return p1 + a * (s.cross(b) / a.
107     cross(b));
108 }
109 point<T> seg_intersection(const line &l)
110     const
111 { //線段交點
112     int res = seg_intersect(l);
113     if (res <= 0)
114         assert(0);
115     if (res == 2)
116         return p1;
117     if (res == 3)
118         return p2;
119     return line_intersection(l);
120 }
121 };

```

3.2 Point

```

1 template <typename T>
2 struct point
3 {
4     T x, y;
5     point() {}
6     point(const T &x, const T &y) : x(x), y(y) {}
7     point operator+(const point &b) const
8     {
9         return point(x + b.x, y + b.y);
10    }
11    point operator-(const point &b) const
12    {

```

```

13        return point(x - b.x, y - b.y);
14    }
15    point operator*(const T &b) const
16    {
17        return point(x * b, y * b);
18    }
19    point operator/(const T &b) const
20    {
21        return point(x / b, y / b);
22    }
23    bool operator==(const point &b) const
24    {
25        return x == b.x && y == b.y;
26    }
27    T dot(const point &b) const
28    {
29        return x * b.x + y * b.y;
30    }
31    T cross(const point &b) const
32    {
33        return x * b.y - y * b.x;
34    }
35    point normal() const
36    { //求法向量
37        return point(-y, x);
38    }
39    T abs2() const
40    { //向量長度的平方
41        return dot(*this);
42    }
43    T rad(const point &b) const
44    { //兩向量的弧度
45        return fabs(atan2(fabs(cross(b)),
46        dot(b)));
47    }
48    T getA() const
49    { //對x軸的弧度
50        T A = atan2(y, x); //超過180度會變負
51        if (A <= -PI / 2)
52            A += PI * 2;
53        return A;
54    }
55 };

```

3.3 Polygon

```

1 template <typename T>
2 struct polygon
3 {
4     polygon() {}
5     vector<point<T>> p; //逆時針順序
6     T area() const
7     { //面積
8         T ans = 0;
9         for (int i = p.size() - 1, j = 0; j
10             < (int)p.size(); i = j++)
11             ans += p[i].cross(p[j]);
12         return ans / 2;
13    }
14    point<T> center_of_mass() const

```

```

15    { //重心
16        T cx = 0, cy = 0, w = 0;
17        for (int i = p.size() - 1, j = 0; j
18            < (int)p.size(); i = j++)
19        {
20            T a = p[i].cross(p[j]);
21            cx += (p[i].x + p[j].x) * a;
22            cy += (p[i].y + p[j].y) * a;
23            w += a;
24        }
25        return point<T>(cx / 3 / w, cy / 3 /
26            w);
27    }
28    char ahas(const point<T> &t) const
29    { //點是否在簡單多邊形內，是的話回傳1、
30        在邊上回傳-1、否則回傳0
31        bool c = 0;
32        for (int i = 0, j = p.size() - 1; i
33            < p.size(); j = i++)
34            if (line<T>(p[i], p[j]).
35                point_on_segment(t))
36                return -1;
37            else if ((p[i].y > t.y) != (p[j]
38                .y > t.y) &&
39                t.x < (p[j].x - p[i].x) /
40                    * (t.y - p[i].y) /
41                    (p[j].y - p[i].y) /
42                    + p[i].x)
43                c = !c;
44        return c;
45    }
46    char point_in_convex(const point<T> &x)
47        const
48    {
49        int l = 1, r = (int)p.size() - 2;
50        while (l <= r)
51        { //點是否在凸多邊形內，是的話回傳1
52            、在邊上回傳-1、否則回傳0
53            int mid = (l + r) / 2;
54            T a1 = (p[mid] - p[0]).cross(x -
55                p[0]);
56            T a2 = (p[mid + 1] - p[0]).cross
57                (x - p[0]);
58            if (a1 >= 0 && a2 <= 0)
59            {
60                T res = (p[mid + 1] - p[mid]
61                    ).cross(x - p[mid]);
62                return res > 0 ? 1 : (res >=
63                    0 ? -1 : 0);
64            }
65            else if (a1 < 0)
66                r = mid - 1;
67            else
68                l = mid + 1;
69        }
70        return 0;
71    }
72    vector<T> getA() const
73    { //凸包邊對x軸的夾角
74        vector<T> res; //一定是遞增的
75        for (size_t i = 0; i < p.size(); ++i
76            )
77            res.push_back((p[(i + 1) % p.
78                size()] - p[i]).getA());
79        return res;
80    }

```

```

81    }
82    bool line_intersect(const vector<T> &A,
83        const line<T> &l) const
84    { //O(logN)
85        int f1 = upper_bound(A.begin(), A.
86            end(), (l.p1 - l.p2).getA()) - A.
87            begin();
88        int f2 = upper_bound(A.begin(), A.
89            end(), (l.p2 - l.p1).getA()) - A.
90            begin();
91        return l.cross_seg(line<T>(p[f1], p[
92            f2]));
93    }
94    polygon cut(const line<T> &l) const
95    { //凸包對直線切割，得到直線l左側的凸包
96        polygon ans;
97        for (int n = p.size(), i = n - 1, j
98            = 0; j < n; i = j++)
99        {
100            if (l.ori(p[i]) >= 0)
101            {
102                ans.p.push_back(p[i]);
103                if (l.ori(p[j]) < 0)
104                    ans.p.push_back(l.
105                        line_intersection(
106                            line<T>(p[i], p[j]))
107                        );
108            }
109            else if (l.ori(p[j]) > 0)
110                ans.p.push_back(l.
111                    line_intersection(line<T>
112                        >(p[i], p[j])));
113        }
114        return ans;
115    }
116    static bool graham_cmp(const point<T> &a
117        , const point<T> &b)
118    { //凸包排序函數 // 起始點不同
119        // 回傳 (a.x < b.x) || (a.x == b.x
120            && a.y < b.y); //最左下角開始
121        return (a.y < b.y) || (a.y == b.y &&
122            a.x < b.x); //Y最小開始
123    }
124    void graham(vector<point<T>> &s)
125    { //凸包 Convexhull 2D
126        sort(s.begin(), s.end(), graham_cmp)
127            ;
128        p.resize(s.size() + 1);
129        int m = 0;
130        // cross >= 0 順時針，cross <= 0 逆
131        時針旋轉
132        for (size_t i = 0; i < s.size(); ++i
133            )
134        {
135            while (m >= 2 && (p[m - 1] - p[m
136                - 2]).cross(s[i] - p[m -
137                2]) <= 0)
138                --m;
139            p[m++] = s[i];
140        }
141        for (int i = s.size() - 2, t = m +
142            1; i >= 0; --i)
143        {

```

4 Graph

4.1 Bellman-Ford

```

1  /*SPA - Bellman-Ford*/
2  #include<bits/stdc++.h>
3  #define inf 99999 //define by you maximum
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]
15                        < dist[j]){
16                         dist[j] = dist[i] +
17                             edges[i][j];
18                         ancestor[j] = i;
19                     }
20                 }
21             }
22         }
23     }
24     for(int i = 0; i < node; i++) //
25         for(int j = 0; j < node; j++) //
26             if(dist[i] + edges[i][j] < dist[
27                 j]){
28                 cout<<"Negative cycle!"<<
29                     endl;
30                 return;
31             }
32 }
33 int main(){
34     int node;
35     cin>>node;
36     edges.resize(node,vector<int>(node,inf))
37     ;
38     dist.resize(node,inf);
39     ancestor.resize(node,-1);
40     int a,b,d;
41     while(cin>>a>>b>>d){
42         /*input: source destination weight*/
43         if(a == -1 && b == -1 && d == -1)
44             break;
45         edges[a][b] = d;
46     }
47     int start;
48     cin>>start;
49     BellmanFord(start,node);
50     return 0;
51 }

```

3.4 Triangle

```

1  template <typename T>
2  struct triangle
3  {
4      point<T> a, b, c;
5      triangle() {}
6      triangle(const point<T> &a, const point<
7          T> &b, const point<T> &c) : a(a), b(b),
8          c(c) {}
9      T area() const
10     {
11         T t = (b - a).cross(c - a) / 2;
12         return t > 0 ? t : -t;
13     }
14     point<T> barycenter() const
15     { //重心
16         return (a + b + c) / 3;
17     }
18     point<T> circumcenter() const
19     { //外心
20         static line<T> u, v;
21         u.p1 = (a + b) / 2;
22         u.p2 = point<T>(u.p1.x - a.y + b.y,
23             u.p1.y + a.x - b.x);
24         v.p1 = (a + c) / 2;
25         v.p2 = point<T>(v.p1.x - a.y + c.y,
26             v.p1.y + a.x - c.x);
27         return u.line_intersection(v);
28     }
29     point<T> incenter() const
30     { //內心
31         T A = sqrt((b - c).abs2()), B = sqrt
32             ((a - c).abs2()), C = sqrt((a -
33             b).abs2());
34         return point<T>(A * a.x + B * b.x +
35             C * c.x, A * a.y + B * b.y + C *
36             c.y) / (A + B + C);
37     }
38     point<T> perpcenter() const
39     { //垂心
40         return barycenter() * 3 -
41             circumcenter() * 2;
42     }
43 }
44 };

```

```

207     if (Q[i].y < Q[r].y)
208         r = i;
209     P.push_back(P[0]), Q.push_back(Q[0]);
210     ;
211     T ans = 1e99;
212     for (int i = 0; i < n; ++i)
213     {
214         while ((P[l] - P[l + 1]).cross(Q
215             [r + 1] - Q[r]) < 0)
216             r = (r + 1) % m;
217         ans = min(ans, line<T>(P[l], P[l
218             + 1]).seg_dis2(line<T>(Q[r
219             ], Q[r + 1])));
220         l = (l + 1) % n;
221     }
222     return P.pop_back(), Q.pop_back(),
223         ans;
224 }
225 static char sign(const point<T> &t)
226 {
227     return (t.y == 0 ? t.x : t.y) < 0;
228 }
229 static bool angle_cmp(const line<T> &A,
230     const line<T> &B)
231 {
232     point<T> a = A.p2 - A.p1, b = B.p2 -
233         B.p1;
234     return sign(a) < sign(b) || (sign(a)
235         == sign(b) && a.cross(b) > 0);
236 }
237 int halfplane_intersection(vector<line<T
238     >> &s)
239 {
240     //半平面交
241     sort(s.begin(), s.end(), angle_cmp);
242     //線段左側為該線段半平面
243     int L, R, n = s.size();
244     vector<point<T>> px(n);
245     vector<line<T>> q(n);
246     q[L = R = 0] = s[0];
247     for (int i = 1; i < n; ++i)
248     {
249         while (L < R && s[i].ori(px[R -
250             1]) <= 0)
251             --R;
252         while (L < R && s[i].ori(px[L])
253             <= 0)
254             ++L;
255         q[++R] = s[i];
256         if (q[R].parallel(q[R - 1]))
257         {
258             --R;
259             if (q[R].ori(s[i].p1) > 0)
260                 q[R] = s[i];
261         }
262         if (L < R)
263             px[R - 1] = q[R - 1].
264                 line_intersection(q[R]);
265     }
266     while (L < R && q[L].ori(px[R - 1])
267         <= 0)
268         --L;
269     p.clear();
270     if (R - L <= 1)

```

```

104     while (m >= t && (p[m - 1] - p[m
105         - 2]).cross(s[i] - p[m -
106         2]) <= 0)
107         --m;
108     p[m++] = s[i];
109 }
110 if (s.size() > 1) // 重複頭一次需扣
111     掉
112     --m;
113 p.resize(m);
114 }
115 T diam()
116 { //直徑
117     int n = p.size(), t = 1;
118     T ans = 0;
119     p.push_back(p[0]);
120     for (int i = 0; i < n; ++i)
121     {
122         point<T> now = p[i + 1] - p[i];
123         while (now.cross(p[t + 1] - p[i]
124             ]) > now.cross(p[t] - p[i]))
125             t = (t + 1) % n;
126         ans = max(ans, (p[i] - p[t]).
127             abs2());
128     }
129     return p.pop_back(), ans;
130 }
131 T min_cover_rectangle()
132 { //最小覆蓋矩形
133     int n = p.size(), t = 1, r = 1, l;
134     if (n < 3)
135         return 0; //也可以做最小周長矩形
136     T ans = 1e99;
137     p.push_back(p[0]);
138     for (int i = 0; i < n; ++i)
139     {
140         point<T> now = p[i + 1] - p[i];
141         while (now.cross(p[t + 1] - p[i]
142             ]) > now.cross(p[t] - p[i]))
143             t = (t + 1) % n;
144         while (now.dot(p[r + 1] - p[i])
145             > now.dot(p[r] - p[i]))
146             r = (r + 1) % n;
147         if (!i)
148             l = r;
149         while (now.dot(p[l + 1] - p[i])
150             <= now.dot(p[l] - p[i]))
151             l = (l + 1) % n;
152         T d = now.abs2();
153         T tmp = now.cross(p[t] - p[i]) *
154             (now.dot(p[r] - p[i]) - now
155             .dot(p[l] - p[i])) / d;
156         ans = min(ans, tmp);
157     }
158     return p.pop_back(), ans;
159 }
160 T dis2(polygon &p1)
161 { //凸包最近距離平方
162     vector<point<T>> &P = p, &Q = p1.p;
163     int n = P.size(), m = Q.size(), l =
164         0, r = 0;
165     for (int i = 0; i < n; ++i)
166     {
167         if (P[i].y < P[l].y)
168             l = i;
169     }
170     for (int i = 0; i < m; ++i)

```

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            */
52            if(a == -1 && b == -1)
53                break;
54            edges.push_back(pair<int, int>(a, b)
55            );
56        }
57        vector<int> result(node, -1);
58        BFS(result, edges, node, 0);
59        return 0;
60    }

```

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    int main(){
28        int node;
29        cin>>node;
30        pass.resize(node,0);
31        int a,b;
32        while(cin>>a>>b){
33            if(a == -1 && b == -1)
34                break;
35            edges.insert(pair<pair<int,int>,int>
36                (>(pair<int,int>(a,b),0)));
37        }
38        int start;
39        cin>>start;
40        route.push_back(start);
41        DFS(start);
42        return 0;
43    }

```

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()){

```

```

12     int cur = pq.top().second;
13     pq.pop();
14     for(int i = 0; i < weight[cur].size
15         ()); i++){
16         if(dist[i] > dist[cur] + weight[
17             cur][i] && weight[cur][i] !=
18             -1){
19             dist[i] = dist[cur] + weight
20                 [cur][i];
21             ancestor[i] = cur;
22             pq.push(make_pair(dist[i], i)
23                 );
24         }
25     }
26 }
27 int main(){
28     int node;
29     cin>>node;
30     int a,b,d;
31     weight.resize(node, vector<int>(node, -1))
32     ;
33     while(cin>>a>>b>>d){
34         /*input: source destination weight*/
35         if(a == -1 && b == -1 && d == -1)
36             break;
37         weight[a][b] = d;
38     }
39     ancestor.resize(node, -1);
40     dist.resize(node, inf);
41     int start;
42     cin>>start;
43     dist[start] = 0;
44     dijkstra(start);
45     return 0;
46 }

```

4.5 Edmonds_karp

```

1 /*Flow - Edmonds-karp*/
2 /*Based on UVa820*/
3 #include<bits/stdc++.h>
4 #define inf 1000000;
5 using namespace std;
6
7 int getMaxFlow(vector<vector<int>> &capacity
8     , int s, int t, int n){
9     int ans = 0;
10    vector<vector<int>> residual(n+1, vector<
11        int>(n+1, 0)); //residual network
12    while(true){
13        vector<int> bottleneck(n+1, 0);
14        bottleneck[s] = inf;
15        queue<int> q;
16        q.push(s);
17        vector<int> pre(n+1, 0);
18        while(!q.empty() && bottleneck[t] == 0){
19            int cur = q.front();
20            q.pop();
21            for(int i = 1; i <= n ; i++){
22                if(bottleneck[i] == 0 && capacity[
23                    cur][i] > residual[cur][i]){
24                    q.push(i);

```

```

25                pre[i] = cur;
26                bottleneck[i] = min(bottleneck[cur
27                    ], capacity[cur][i] - residual
28                    [cur][i]);
29            }
30        }
31        if(bottleneck[t] == 0) break;
32        for(int cur = t; cur != s; cur = pre[cur
33            ]){
34            residual[pre[cur]][cur] +=
35            bottleneck[t];
36            residual[cur][pre[cur]] -=
37            bottleneck[t];
38        }
39        ans += bottleneck[t];
40    }
41    return ans;
42 }
43 int main(){
44     int testcase = 1;
45     int n;
46     while(cin>>n){
47         if(n == 0)
48             break;
49         vector<vector<int>> capacity(n+1, vector<
50             int>(n+1, 0));
51         int s, t, c;
52         cin >> s >> t >> c;
53         int a, b, bandwidth;
54         for(int i = 0 ; i < c ; ++i){
55             cin >> a >> b >> bandwidth;
56             capacity[a][b] += bandwidth;
57             capacity[b][a] += bandwidth;
58         }
59         cout << "Network " << testcase++ << endl
60         ;
61         cout << "The bandwidth is " <<
62             getMaxFlow(capacity, s, t, n) << "."
63             << endl;
64         cout << endl;
65     }
66     return 0;
67 }

```

4.6 Floyd-warshall

```

1 /*SPA - Floyd-Warshall*/
2 #include<bits/stdc++.h>
3 #define inf 99999
4 using namespace std;
5 void floyd_warshall(vector<vector<int>>&
6     distance, vector<vector<int>>& ancestor,
7     int n){
8     for (int k = 0; k < n; k++){
9         for (int i = 0; i < n; i++){
10            for (int j = 0; j < n; j++){
11                if(distance[i][k] + distance
12                    [k][j] < distance[i][j])
13                    {
14                        distance[i][j] =
15                            distance[i][k] +
16                            distance[k][j];

```



```

11         ancestor[i][j] =
12             ancestor[k][j];
13     }
14 }
15 }
16 }
17 int main(){
18     int n;
19     cin >> n;
20     int a, b, d;
21     vector<vector<int>> distance(n, vector<
22         int>(n,99999));
23     vector<vector<int>> ancestor(n, vector<
24         int>(n,-1));
25     while(cin>>a>>b>>d){
26         if(a == -1 && b == -1 && d == -1)
27             break;
28         distance[a][b] = d;
29         ancestor[a][b] = a;
30     }
31     for (int i = 0; i < n; i++)
32         distance[i][i] = 0;
33     floyd_warshall(distance, ancestor, n);
34     /*Negative cycle detection*/
35     for (int i = 0; i < n; i++){
36         if(distance[i][i] < 0){
37             cout << "Negative cycle!" <<
38                 endl;
39             break;
40         }
41     }
42     return 0;
43 }

```

4.7 Kruskal

```

1 /*mst - Kruskal*/
2 #include<bits/stdc++.h>
3 using namespace std;
4 struct edges{
5     int from;
6     int to;
7     int weight;
8     friend bool operator < (edges a, edges b)
9     ){
10         return a.weight > b.weight;
11     }
12 };
13 int find(int x,vector<int>& union_set){
14     if(x != union_set[x])
15         union_set[x] = find(union_set[x],
16             union_set);
17     return union_set[x];
18 }
19 void merge(int a,int b,vector<int>&
20     union_set){
21     int pa = find(a, union_set);
22     int pb = find(b, union_set);
23     if(pa != pb)
24         union_set[pa] = pb;
25 }

```

```

23 void kruskal(priority_queue<edges> pq,int n)
24 {
25     vector<int> union_set(n, 0);
26     for (int i = 0; i < n; i++){
27         union_set[i] = i;
28     }
29     int edge = 0;
30     int cost = 0; //evaluate cost of mst
31     while(!pq.empty() && edge < n - 1){
32         edges cur = pq.top();
33         int from = find(cur.from, union_set);
34         int to = find(cur.to, union_set);
35         if(from != to){
36             merge(from, to, union_set);
37             edge += 1;
38             cost += cur.weight;
39         }
40         pq.pop();
41     }
42     if(edge < n-1)
43         cout << "No mst" << endl;
44     else
45         cout << cost << endl;
46 }
47 int main(){
48     int n;
49     cin >> n;
50     int a, b, d;
51     priority_queue<edges> pq;
52     while(cin>>a>>b>>d){
53         if(a == -1 && b == -1 && d == -1)
54             break;
55         edges tmp;
56         tmp.from = a;
57         tmp.to = b;
58         tmp.weight = d;
59         pq.push(tmp);
60     }
61     kruskal(pq, n);
62     return 0;
63 }

```

4.8 Prim

```

1 /*mst - Prim*/
2 #include<bits/stdc++.h>
3 #define inf 99999
4 using namespace std;
5 struct edges{
6     int from;
7     int to;
8     int weight;
9     friend bool operator < (edges a, edges b)
10    ){
11        return a.weight > b.weight;
12    }
13 };
14 void Prim(vector<vector<int>> gp,int n,int
15     start){
16     vector<bool> pass(n,false);
17     int edge = 0;
18     int cost = 0; //evaluate cost of mst
19     priority_queue<edges> pq;

```

```

19     for (int i = 0; i < n; i++){
20         if(gp[start][i] != inf){
21             edges tmp;
22             tmp.from = start;
23             tmp.to = i;
24             tmp.weight = gp[start][i];
25             pq.push(tmp);
26         }
27     }
28     pass[start] = true;
29     while(!pq.empty() && edge < n-1){
30         edges cur = pq.top();
31         pq.pop();
32         if(!pass[cur.to]){
33             for (int i = 0; i < n; i++){
34                 if(gp[cur.to][i] != inf){
35                     edges tmp;
36                     tmp.from = cur.to;
37                     tmp.to = i;
38                     tmp.weight = gp[cur.to][
39                         i];
40                     pq.push(tmp);
41                 }
42             }
43             pass[cur.to] = true;
44             edge += 1;
45             cost += cur.weight;
46         }
47     }
48     if(edge < n-1)
49         cout << "No mst" << endl;
50     else
51         cout << cost << endl;
52 }
53 int main(){
54     int n;
55     cin >> n;
56     int a, b, d;
57     vector<vector<int>> gp(n,vector<int>(n,
58         inf));
59     while(cin>>a>>b>>d){
60         if(a == -1 && b == -1 && d == -1)
61             break;
62         if(gp[a][b] > d)
63             gp[a][b] = d;
64     }
65     Prim(gp,n,0);
66     return 0;
67 }

```

4.9 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--)
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--)
6   {
7     if (num[i] <= '9' && num[i] >= '0')
8     {
9       temp += (num[i] - '0') * base;
10      base = base * 16;
11    }
12    else if (num[i] <= 'F' && num[i] >= 'A')
13    {
14      temp += (num[i] - 'A' + 10) * 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 }

```

5.4 Mod

```

1 int pow_mod(int a, int n, int m) // a ^ n
2   mod m;
3 { // a, n, m < 10 ^ 9
4   if (n == 0)
5     return 1;
6   int x = pow_mid(a, n / 2, m);
7   long long ans = (long long)x * x % m;
8   if (n % 2 == 1)
9     ans = ans * a % m;
10  return (int)ans;
11 }
12 // 加法: (a + b) % p = (a % p + b % p) % p;
13 // 減法: (a - b) % p = (a % p - b % p + p) % p;
14 // 乘法: (a * b) % p = (a % p * b % p) % p;
15 // 次方: (a ^ b) % p = ((a % p) ^ b) % p;
16 // 加法結合律: ((a + b) % p + c) % p = (a +
17   (b + c)) % p;
18 // 乘法結合律: ((a * b) % p * c) % p = (a *
19   (b * c)) % p;
20 // 加法交換律: (a + b) % p = (b + a) % p;
21 // 乘法交換律: (a * b) % p = (b * a) % p;
22 // 結合律: ((a + b) % p * c) % p = ((a * c) % p
23   + (b * c) % p) % p;
24 // 如果 a ≡ b(mod m) · 我們會說 a,b 在模 m
25   下同餘。
26 // 整除性: a ≡ b(mod m) ⇔ c ≡ m = a - b, c
27   ⇔ Z ⇔ a ≡ b(mod m) ⇔ m|a-b
28 // 遞移性: 若 a ≡ b(mod c), b ≡ d(mod c) 則
29   a ≡ d(mod c)
30 /****基本運算***

```

```

26 // a ≡ b(mod m) ⇔ { a ± c ≡ b ± d(mod m) }
27 // c ≡ d(mod m) ⇔ { a * c ≡ b * d(mod m) }
28 // 放大縮小模數: kZ+, a ≡ b(mod m) ⇔ kZ+ a
29   ≡ kZ+ b(mod km)

```

5.5 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());

```

5.6 PI

```

1 #define PI acos(-1)
2 #define PI M_PI
3 const double PI = atan2(0.0, -1.0);

```

5.7 Prime table

```

1 const int maxn = sqrt(INT_MAX);
2 vector<int> p;
3 bitset<maxn> is_notp;
4 void PrimeTable()
5 {
6   is_notp.reset();
7   is_notp[0] = is_notp[1] = 1;
8   for (int i = 2; i <= maxn; ++i)
9   {
10    if (!is_notp[i])
11      p.push_back(i);
12    for (int j = 0; j < (int)p.size(); ++j)
13    {
14      if (i * p[j] > maxn)
15        break;
16      is_notp[i * p[j]] = 1;
17      if (i % p[j] == 0)
18        break;
19    }
20  }
21 }

```

5.8 primeBOOL

```

1 // n < 4759123141   chk = [2, 7, 61]
2 // n < 1122004669633   chk = [2, 13, 23,
3   1662803]
4 // n < 2^64   chk = [2, 325, 9375,
5   28178, 450775, 9780504, 1795265022]
6 long long fmul(long long a, long long n,
7   long long mod)
8 {

```

```

6 long long ret = 0;
7 for (; n; n >>= 1)
8 {
9   if (n & 1)
10     (ret += a) %= mod;
11   (a += a) %= mod;
12 }
13 return ret;
14 }
15 }
16 long long fpow(long long a, long long n,
17   long long mod)
18 {
19   long long ret = 1LL;
20   for (; n; n >>= 1)
21   {
22     if (n & 1)
23       ret = fmul(ret, a, mod);
24     a = fmul(a, a, mod);
25   }
26   return ret;
27 }
28 bool check(long long a, long long u, long
29   long n, int t)
30 {
31   a = fpow(a, u, n);
32   if (a == 0)
33     return true;
34   if (a == 1 || a == n - 1)
35     return true;
36   for (int i = 0; i < t; ++i)
37   {
38     a = fmul(a, a, n);
39     if (a == 1)
40       return false;
41     if (a == n - 1)
42       return true;
43   }
44   return false;
45 }
46 bool is_prime(long long n)
47 {
48   if (n < 2)
49     return false;
50   if (n % 2 == 0)
51     return n == 2;
52   long long u = n - 1;
53   int t = 0;
54   for (; u & 1; u >>= 1, ++t)
55     ;
56   for (long long i : chk)
57   {
58     if (!check(i, u, n, t))
59       return false;
60   }
61   return true;
62 }
63 // if (is_prime(int num)) // true == prime
64 // 反之亦然

```

5.9 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("%.31f\n", R);
14 }
```

5.10 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); //去除左
32        右兩邊空格
33    if (s[le] == '(')
```

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

5.11 數字乘法組合

```
1 void dfs(int j, int old, int num, vector<int>
2     > com, vector<vector<int>> &ans)
3 {
4     for (int i = j; i <= sqrt(num); i++)
5     {
6         if (old == num)
7             com.clear();
8         if (num % i == 0)
9         {
10            vector<int> a;
11            a = com;
12            a.push_back(i);
13            finds(i, old, num / i, a, ans);
14            a.push_back(num / i);
15            ans.push_back(a);
16        }
17    }
18    vector<vector<int>> ans;
19    vector<int> zero;
20    dfs(2, num, num, zero, ans);
21    /*num 為 input 數字*/
22    for (int i = 0; i < ans.size(); i++)
23    {
24        for (int j = 0; j < ans[i].size() - 1; j++)
25            cout << ans[i][j] << " ";
26        cout << ans[i][ans[i].size() - 1] <<
27        endl;
```

5.12 數字加法組合

```
1 void recur(int i, int n, int m, vector<int>
2     &out, vector<vector<int>> &ans)
3 {
4     if (n == 0)
5     {
6         for (int i : out)
7             if (i > m)
8                 return;
9         ans.push_back(out);
10    }
11    for (int j = i; j <= n; j++)
12    {
13        out.push_back(j);
14        recur(j, n - j, m, out, ans);
15        out.pop_back();
16    }
```

```
13        recur(j, n - j, m, out, ans);
14        out.pop_back();
15    }
16 }
17 vector<vector<int>> ans;
18 vector<int> zero;
19 recur(1, num, num, zero, ans);
20 /* num 為 input 數字
21 for (int i = 0; i < ans.size(); i++)
22 {
23     for (int j = 0; j < ans[i].size() - 1; j++)
24         cout << ans[i][j] << " ";
25     cout << ans[i][ans[i].size() - 1] <<
26     endl;
```

5.13 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 }
```

5.14 質因數分解

```
1 void primeFactorization(int n) // 配合質數表
2 {
3     for (int i = 0; i < (int)p.size(); ++i)
4     {
5         if (p[i] * p[i] > n)
6             break;
7         if (n % p[i])
8             continue;
9         cout << p[i] << ' ';
10        while (n % p[i] == 0)
11            n /= p[i];
12    }
13    if (n != 1)
14        cout << n << ' ';
15    cout << '\n';
16 }
```

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
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 }
```



```

14 if (board[rowNum][colNum] != 0)
15     return backtracking(board, rows,
16         cols, boxes, index + 1, n);
17
18 for (int i = 1; i <= n2; i++)
19 {
20     if (!rows[rowNum][i] && !cols[colNum]
21         ][i] && !boxes[getSquareIndex(
22             rowNum, colNum, n)][i])
23     {
24         rows[rowNum][i] = true;
25         cols[colNum][i] = true;
26         boxes[getSquareIndex(rowNum,
27             colNum, n)][i] = true;
28         board[rowNum][colNum] = i;
29         if (backtracking(board, rows,
30             cols, boxes, index + 1, n))
31             return true;
32         board[rowNum][colNum] = 0;
33         rows[rowNum][i] = false;
34         cols[colNum][i] = false;
35         boxes[getSquareIndex(rowNum,
36             colNum, n)][i] = false;
37     }
38 }
39 return false;
40
41 /*用法 main*/
42 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
43 vector<vector<int>> board(n * n + 1, vector<
44     int>(n * n + 1, 0));
45 vector<vector<bool>> isRow(n * n + 1, vector<
46     bool>(n * n + 1, false));
47 vector<vector<bool>> isColumn(n * n + 1,
48     vector<bool>(n * n + 1, false));
49 vector<vector<bool>> isSquare(n * n + 1,
50     vector<bool>(n * n + 1, false));
51
52 for (int i = 0; i < n * n; ++i)
53 {
54     for (int j = 0; j < n * n; ++j)
55     {
56         int number;
57         cin >> number;
58         board[i][j] = number;
59         if (number == 0)
60             continue;
61         isRow[i][number] = true;
62         isColumn[j][number] = true;
63         isSquare[getSquareIndex(i, j, n)][
64             number] = true;
65     }
66 }
67 if (backtracking(board, isRow, isColumn,
68     isSquare, 0, n))
69     /*有解答*/
70 else
71     /*解答*/

```

7 String

7.1 Sliding window

```

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

```

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
8     char *strs = new char[str.length() + 1];
9     char *d = new char[delim.length() + 1];
10    strcpy(strs, str.c_str());
11    strcpy(d, delim.c_str());
12    char *p = strtok(strs, d);
13    while (p)
14    {
15        string s = p;
16        res.push_back(s);
17        p = strtok(NULL, d);
18    }
19    return res;

```

8 data structure

8.1 Bigint

```

1 //台大
2 struct Bigint
3 {
4     static const int LEN = 60;
5     static const int BIGMOD = 10000;
6     int s;
7     int vl, v[LEN];
8     // vector<int> v;
9     Bigint() : s(1) { vl = 0; }
10    Bigint(long long a)
11    {
12        s = 1;
13        vl = 0;
14        if (a < 0)
15        {
16            s = -1;
17            a = -a;
18        }
19        while (a)
20        {
21            push_back(a % BIGMOD);
22            a /= BIGMOD;
23        }
24    }
25    Bigint(string str)
26    {
27        s = 1;
28        vl = 0;
29        int stPos = 0, num = 0;
30        if (!str.empty() && str[0] == '-')
31        {
32            stPos = 1;
33            s = -1;
34        }
35        for (int i = str.length() - 1, q =
36            1; i >= stPos; i--)
37        {
38            num += (str[i] - '0') * q;
39            if ((q * 10) >= BIGMOD)
40            {
41                push_back(num);
42                num = 0;
43                q = 1;
44            }
45        }
46        if (num)
47            push_back(num);
48        n();
49    }
50    int len() const
51    {
52        return vl; //return SZ(v);
53    }
54    bool empty() const { return len() == 0; }
55    void push_back(int x)
56    {
57        v[vl++] = x; //v.PB(x);

```

```

58 void pop_back()
59 {
60     vl--; //v.pop_back();
61 }
62 int back() const
63 {
64     return v[vl - 1]; //return v.back();
65 }
66 void n()
67 {
68     while (!empty() && !back())
69         pop_back();
70 }
71 void resize(int nl)
72 {
73     vl = nl;
74     fill(v, v + vl, 0); //fill(ALL(v),
75         0);
76 }
77 void print() const
78 {
79     if (empty())
80     {
81         putchar('0');
82         return;
83     }
84     if (s == -1)
85         putchar('-');
86     printf("%d", back());
87     for (int i = len() - 2; i >= 0; i--)
88         printf("%.4d", v[i]);
89 }
90 friend std::ostream &operator<<(std::
91     ostream &out, const Bigint &a)
92 {
93     if (a.empty())
94     {
95         out << "0";
96         return out;
97     }
98     if (a.s == -1)
99         out << "-";
100    out << a.back();
101    for (int i = a.len() - 2; i >= 0; i
102        --)
103    {
104        char str[10];
105        snprintf(str, 5, "%.4d", a.v[i]);
106        out << str;
107    }
108    return out;
109 }
110 int cp3(const Bigint &b) const
111 {
112     if (s != b.s)
113         return s - b.s;
114     if (s == -1)
115         return -(*this).cp3(-b);
116     if (len() != b.len())
117         return len() - b.len(); //int
118     for (int i = len() - 1; i >= 0; i--)
119         if (v[i] != b.v[i])
120             return v[i] - b.v[i];
121     return 0;

```

```

120 bool operator<(const Bigint &b) const
121 {
122     return cp3(b) < 0;
123 }
124 bool operator<=(const Bigint &b) const
125 {
126     return cp3(b) <= 0;
127 }
128 bool operator==(const Bigint &b) const
129 {
130     return cp3(b) == 0;
131 }
132 bool operator!=(const Bigint &b) const
133 {
134     return cp3(b) != 0;
135 }
136 bool operator>(const Bigint &b) const
137 {
138     return cp3(b) > 0;
139 }
140 bool operator>=(const Bigint &b) const
141 {
142     return cp3(b) >= 0;
143 }
144 Bigint operator-() const
145 {
146     Bigint r = (*this);
147     r.s = -r.s;
148     return r;
149 }
150 Bigint operator+(const Bigint &b) const
151 {
152     if (s == -1)
153         return -(*this) + (-b);
154     if (b.s == -1)
155         return (*this) - (-b);
156     Bigint r;
157     int nl = max(len(), b.len());
158     r.resize(nl + 1);
159     for (int i = 0; i < nl; i++)
160     {
161         if (i < len())
162             r.v[i] += v[i];
163         if (i < b.len())
164             r.v[i] += b.v[i];
165         if (r.v[i] >= BIGMOD)
166         {
167             r.v[i + 1] += r.v[i] /
168                 BIGMOD;
169             r.v[i] %= BIGMOD;
170         }
171     }
172     r.n();
173     return r;
174 }
175 Bigint operator-(const Bigint &b) const
176 {
177     if (s == -1)
178         return -(*this) - (-b);
179     if (b.s == -1)
180         return (*this) + (-b);
181     if ((*this) < b)
182         return -(b - (*this));
183     Bigint r;
184     r.resize(len());
185     for (int i = 0; i < len(); i++)

```

```

186 {
187     r.v[i] += v[i];
188     if (i < b.len())
189         r.v[i] -= b.v[i];
190     if (r.v[i] < 0)
191     {
192         r.v[i] += BIGMOD;
193         r.v[i + 1]--;
194     }
195     r.n();
196     return r;
197 }
198 Bigint operator*(const Bigint &b)
199 {
200     Bigint r;
201     r.resize(len() + b.len() + 1);
202     r.s = s * b.s;
203     for (int i = 0; i < len(); i++)
204     {
205         for (int j = 0; j < b.len(); j
206             ++
207         {
208             r.v[i + j] += v[i] * b.v[j];
209             if (r.v[i + j] >= BIGMOD)
210             {
211                 r.v[i + j + 1] += r.v[i
212                     + j] / BIGMOD;
213                 r.v[i + j] %= BIGMOD;
214             }
215         }
216     }
217     r.n();
218     return r;
219 }
220 Bigint operator/(const Bigint &b)
221 {
222     Bigint r;
223     r.resize(max(1, len() - b.len() + 1)
224         );
225     int oriS = s;
226     Bigint b2 = b; // b2 = abs(b)
227     s = b2.s * r.s = 1;
228     for (int i = r.len() - 1; i >= 0; i
229         --
230     {
231         int d = 0, u = BIGMOD - 1;
232         while (d < u)
233         {
234             int m = (d + u + 1) >> 1;
235             r.v[i] = m;
236             if ((r * b2) > (*this))
237                 u = m - 1;
238             else
239                 d = m;
240         }
241         r.v[i] = d;
242     }
243     s = oriS;
244     r.s = s * b.s;
245     r.n();
246     return r;
247 }
248 Bigint operator%(const Bigint &b)
249 {
250     return (*this) - (*this) / b * b;
251 }

```

8.2 matirx

```

1 template <typename T>
2 struct Matrix
3 {
4     using rt = std::vector<T>;
5     using mt = std::vector<rt>;
6     using matrix = Matrix<T>;
7     int r, c; // [r][c]
8     mt m;
9     Matrix(int r, int c) : r(r), c(c), m(r,
10         rt(c)) {}
11     Matrix(mt a) { m = a, r = a.size(), c =
12         a[0].size(); }
13     rt &operator[](int i) { return m[i]; }
14     matrix operator+(const matrix &a)
15     {
16         matrix rev(r, c);
17         for (int i = 0; i < r; ++i)
18             for (int j = 0; j < c; ++j)
19                 rev[i][j] = m[i][j] + a.m[i
20                     ][j];
21         return rev;
22     }
23     matrix operator-(const matrix &a)
24     {
25         matrix rev(r, c);
26         for (int i = 0; i < r; ++i)
27             for (int j = 0; j < c; ++j)
28                 rev[i][j] = m[i][j] - a.m[i
29                     ][j];
30         return rev;
31     }
32     matrix operator*(const matrix &a)
33     {
34         matrix rev(r, a.c);
35         matrix tmp(a.c, a.r);
36         for (int i = 0; i < a.r; ++i)
37             for (int j = 0; j < a.c; ++j)
38                 tmp[j][i] = a.m[i][j];
39         for (int i = 0; i < r; ++i)
40             for (int j = 0; j < a.c; ++j)
41                 for (int k = 0; k < c; ++k)
42                     rev.m[i][j] += m[i][k] *
43                         tmp[j][k];
44         return rev;
45     }
46     bool inverse() //逆矩阵判断
47     {
48         Matrix t(r, r + c);
49         for (int y = 0; y < r; y++)
50         {
51             t.m[y][c + y] = 1;
52             for (int x = 0; x < c; ++x)
53                 t.m[y][x] = m[y][x];
54         }
55         if (!t.gas())
56             return false;
57         for (int y = 0; y < r; y++)
58             for (int x = 0; x < c; ++x)

```

```

59         m[y][x] = t.m[y][c + x] / t.
60             m[y][y];
61     }
62     return true;
63 }
64 T gas() //行列式
65 {
66     vector<T> lazy(r, 1);
67     bool sign = false;
68     for (int i = 0; i < r; ++i)
69     {
70         if (m[i][i] == 0)
71         {
72             int j = i + 1;
73             while (j < r && !m[j][i])
74                 j++;
75             if (j == r)
76                 continue;
77             m[i].swap(m[j]);
78             sign = !sign;
79         }
80         for (int j = 0; j < r; ++j)
81         {
82             if (i == j)
83                 continue;
84             lazy[j] = lazy[j] * m[i][i];
85             T mx = m[j][i];
86             for (int k = 0; k < c; ++k)
87                 m[j][k] = m[j][k] * m[i
88                     ][i] - m[i][k] * mx;
89         }
90     }
91     T det = sign ? -1 : 1;
92     for (int i = 0; i < r; ++i)
93     {
94         det = det * m[i][i];
95         det = det / lazy[i];
96         for (auto &j : m[i])
97             j /= lazy[i];
98     }
99     return det;
100 }
101 };

```

8.3 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 {
24     u[tot] = from;
25     v[tot] = to;
26     cap[tot] = cp;
27     flow[tot] = flw;
28     next[tot] = first[u[tot]];
29     first[u[tot]] = tot;
30     ++tot;
31 }
32 bool bfs()
33 {
34     ++tim;
35     dis[s] = 0;
36     vi[s] = tim;
37
38     int head, tail;
39     head = tail = 1;
40     que[head] = s;
41     while (head <= tail)
42     {
43         for (int i = first[que[head]]; i
44             != -1; i = next[i])
45         {
46             if (vi[v[i]] != tim && cap[i]
47                 > flow[i])
48             {
49                 vi[v[i]] = tim;
50                 dis[v[i]] = dis[que[head]]
51                     + 1;
52                 que[++tail] = v[i];
53             }
54         }
55         ++head;
56         return vi[t] == tim;
57     }
58 }
59 ll dfs(int x, ll a)
60 {
61     if (x == t || a == 0)
62         return a;
63     ll flw = 0, f;
64     int &i = cur[x];
65     for (i = first[x]; i != -1; i = next
66         [i])
67     {
68         if (dis[x] + 1 == dis[v[i]] && (
69             f = dfs(v[i], min(a, cap[i]
70                 - flow[i])) > 0)
71         {
72             flow[i] += f;
73             flow[i ^ 1] -= f;
74             a -= f;
75             flw += f;
76             if (a == 0)
77                 break;
78         }
79     }
80     return flw;
81 }
82 ll MaxFlow(int s, int t)
83 {
84     this->s = s;

```

```

78     this->t = t;
79     ll flw = 0;
80     while (bfs())
81     {
82         for (int i = 1; i <= n; ++i)
83             cur[i] = 0;
84         flw += dfs(s, oo);
85     }
86     return flw;
87 }
88 };
89 // MF Net;
90 // Net.n = n;
91 // Net.Clear();
92 // a 到 b (注意從1開始!!!!)
93 // Net.Add(a, b, w, 0);
94 // Net.MaxFlow(s, d)
95 // s 到 d 的 MF

```

8.4 Trie

```

1 // biginter字典數
2 struct BigInteger{
3     static const int BASE = 100000000;
4     static const int WIDTH = 8;
5     vector<int> s;
6     BigInteger(long long num = 0){
7         *this = num;
8     }
9     BigInteger operator = (long long num){
10         s.clear();
11         do{
12             s.push_back(num % BASE);
13             num /= BASE;
14         }while(num > 0);
15         return *this;
16     }
17     BigInteger operator = (const string& str)
18     {
19         s.clear();
20         int x, len = (str.length() - 1) /
21             WIDTH + 1;
22         for(int i = 0; i < len; i++){
23             int end = str.length() - i*WIDTH
24                 ;
25             int start = max(0, end-WIDTH);
26             sscanf(str.substr(start, end-
27                 start).c_str(), "%d", &x);
28             s.push_back(x);
29         }
30         return *this;
31     }
32     BigInteger operator + (const BigInteger&
33         b) const{
34         BigInteger c;
35         c.s.clear();
36         for(int i = 0, g = 0; i++){
37             if(g == 0 && i >= s.size() && i
38                 >= b.s.size()) break;
39             int x = g;
40             if(i < s.size()) x+=s[i];

```

```

36         if(i < b.s.size()) x+=b.s[i];
37         c.s.push_back(x % BASE);
38         g = x / BASE;
39     }
40     return c;
41 }
42 };
43
44 ostream& operator << (ostream &out, const
45     BigInteger& x){
46     out << x.s.back();
47     for(int i = x.s.size()-2; i >= 0; i--){
48         char buf[20];
49         sprintf(buf, "%08d", x.s[i]);
50         for(int j = 0; j < strlen(buf); j++){
51             out << buf[j];
52         }
53     }
54     return out;
55 }
56
57 istream& operator >> (istream &in,
58     BigInteger& x){
59     string s;
60     if(!(in >> s))
61         return in;
62
63     x = s;
64     return in;
65 }
66
67 struct Trie{
68     int c[5000005][10];
69     int val[5000005];
70     int sz;
71     int getIndex(char c){
72         return c - '0';
73     }
74     void init(){
75         memset(c[0], 0, sizeof(c[0]));
76         memset(val, -1, sizeof(val));
77         sz = 1;
78     }
79     void insert(BigInteger x, int v){
80         int u = 0;
81         int max_len_count = 0;
82         int firstNum = x.s.back();
83         char firstBuf[20];
84         sprintf(firstBuf, "%d", firstNum);
85         for(int j = 0; j < strlen(firstBuf);
86             j++){
87             int index = getIndex(firstBuf[j]
88                 );
89             if(!c[u][index]){
90                 memset(c[sz], 0, sizeof(c[
91                     sz]));
92                 val[sz] = v;
93                 c[u][index] = sz++;
94             }
95             u = c[u][index];
96             max_len_count++;
97         }
98     }

```

```

96     for(int i = x.s.size()-2; i >= 0; i
97         --){
98         char buf[20];
99         sprintf(buf, "%08d", x.s[i]);
100         for(int j = 0; j < strlen(buf)
101             && max_len_count < 50; j++){
102             int index = getIndex(buf[j])
103                 ;
104             if(!c[u][index]){
105                 memset(c[sz], 0, sizeof
106                     (c[sz]));
107                 val[sz] = v;
108                 c[u][index] = sz++;
109             }
110             u = c[u][index];
111             max_len_count++;
112         }
113         if(max_len_count >= 50){
114             break;
115         }
116     }
117 }
118
119 int find(const char* s){
120     int u = 0;
121     int n = strlen(s);
122     for(int i = 0; i < n; ++i)
123     {
124         int index = getIndex(s[i]);
125         if(!c[u][index]){
126             return -1;
127         }
128         u = c[u][index];
129     }
130     return val[u];
131 }

```

8.5 分數

```

1 typedef long long ll;
2 struct fraction
3 {
4     ll n, d;
5     fraction(const ll &n = 0, const ll &d =
6         1) : n(n), d(d)
7     {
8         ll t = __gcd(n, d);
9         n /= t, d /= t;
10         if (d < 0)
11             n = -n, d = -d;
12     }
13     fraction operator-() const
14     {
15         return fraction(-n, d);
16     }
17     fraction operator+(const fraction &b)
18         const
19     {
20         return fraction(n * b.d + b.n * d, d * b
21             .d);
22     }

```

```
20 fraction operator-(const fraction &b)
    const
21 {
22     return fraction(n * b.d - b.n * d, d * b
        .d);
23 }
24 fraction operator*(const fraction &b)
    const
25 {
26     return fraction(n * b.n, d * b.d);
27 }
28 fraction operator/(const fraction &b)
    const
29 {
30     return fraction(n * b.d, d * b.n);
31 }
32 void print()
33 {
34     cout << n;
35     if (d != 1)
36         cout << "/" << d;
37 }
38 };
```

TO DO WRITING NOT THINKING

Contents

1 Basic	1	2.5 LCIS	1	4.5 Edmonds_karp	5	5.12 數字加法組合	8
1.1 data range	1	2.6 LCS	1	4.6 Floyd-warshall	5	5.13 羅馬數字	8
1.2 IO_fast	1	2.7 LIS	1	4.7 Kruskal	6	5.14 質因數分解	8
2 DP	1	2.8 LPS	2	4.8 Prim	6	6 Other	8
2.1 KMP	1	2.9 Max_subarray	2	4.9 Union_find	6	6.1 Weighted Job Scheduling . .	8
2.2 Knapsack Bounded	1	2.10 Money problem	2	5 Mathematics	6	6.2 數獨解法	8
2.3 Knapsack sample	1	3 Geometry	2	5.1 Combination	6	7 String	9
2.4 Knapsack Unbounded	1	3.1 Line	2	5.2 Extended Euclidean	6	7.1 Sliding window	9
		3.2 Point	3	5.3 Hex to Dec	7	7.2 Split	9
		3.3 Polygon	3	5.4 Mod	7	8 data structure	9
		3.4 Triangle	4	5.5 Permutation	7	8.1 Bigint	9
		4 Graph	4	5.6 PI	7	8.2 matirx	10
		4.1 Bellman-Ford	4	5.7 Prime table	7	8.3 MFlow	10
		4.2 BFS-queue	5	5.8 primeBOOL	7	8.4 Trie	11
		4.3 DFS-rec	5	5.9 二分逼近法	8	8.5 分數	11
		4.4 Dijkstra	5	5.10 四則運算	8		
				5.11 數字乘法組合	8		