ACM-ICPC Template



GuessEver

Last Modify at January 30, 2015

Contents

1	Dyna	mic Programming																		3
	1.1	LCS - Longest Common Subsequence																		3
	1.2	LIS - Longest Increasing Subsequence .																		3
		Maximum Continuous Subsequence Sum																		4
	1.4	数位 dp	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	4
2	Math	1																		5
	2.1	GCD && LCM																		5
		2.1.1 GCD - Greatest Common Divisor .																		5
		2.1.2 LCM - Least Common Multiple																		5
		2.1.3 E_GCD - Extended Greatest Common	D	iv	is	οг														5
	2.2	Prime																		5
		2.2.1 Make Prime List																		5
		2.2.2 Prime Factor																		6
		2.2.3 Fast Power	•	•	•	•	•	•		•		•	•	•	•	•	•	•	•	6
3	Data	structure																		7
	3.1	Leftist Tree																		7
		Partition Tree																		7
	3.3	Size Balanced Tree				•									•		•			8
4	Grap	sh.																		9
•	•	Shortest path																		9
		4.1.1 Dijkstra																		9
		4.1.2 Spfa																		10
		4.1.3 Floyd																		10
	4.2	Minimum Spanning Tree																		11
		4.2.1 Prim																		11
		4.2.2 Kruskal																		12
	4.3	Tarjan - Strong Union																		12
		LCA																		13
		4.4.1 Tarjan																		13
		4.4.2 Doubling Algorithm																		14
	4.5	Bipartite Graph																		14
		4.5.1 Maximal Matching - The Hungarian	a	lg	οг	itl	חר													14
		4.5.2 Optimal Matching - KM																		15
	4.6	Network flow																		15
		4.6.1 Maximum Flow - isap		•	•				•								•			15
		4.6.2 Minimum Cost Maximum Flow - spfa		•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	16
5	Geom	netry																		17
		Convex Hull																		17
		All																		18
6	Stri	ng Tools																		24
-		KMP											_	_	_		_	_		
		Suffix Array																		

ACM-ICPC Template GuessE									
7	pols	2.	5						
	.1 BigInteger - C++	2	5						
	.2 C char*	2	9						
	.3 C++ std::string	2	9						
	.4 Batch test	2	9						
	7.4.1 @Linux	2	9						
	7.4.2 @Windows	3	0						

1 Dynamic Programming

1.1 LCS - Longest Common Subsequence

```
int LCS() // O(N*N)
   {//字符串纠正到以 1 为下标
2
3
       int f[N][N];
       int res = 0;
4
       for(int i = 1; i < lena; i++)</pre>
5
           for(int j = 1; j < lenb; j++)</pre>
6
7
               if(a[i] == a[j]) f[i][j] = f[i-1][j-1] + 1;
8
9
               else f[i][j] = max(f[i-1][j], f[i][j-1]);
               res = max(res, f[i][j]);
10
11
12
       return res;
13
14
   int LCS() // O(NlogN)
15
   {//把 LCM 转化为 LIS 来做
16
17
   // 1 2 5 9 3 --> 1 2 3 4 5
   // 1 5 3 9 2 ---> 1 3 5 4 2 ---> 对这个序列跑LIS()
18
19
       //----change-
           //这里就要针对数据自己想尽办法转化了
20
           for(int i = 1; i <= n; i++) h[a[i]] = i;</pre>
21
22
           for(int i = 1; i <= n; i++) b[i] = h[b[i]];</pre>
23
       //----end-
24
       return LIS();
25 | }
```

1.2 LIS - Longest Increasing Subsequence

```
int f[N];
   int LIS()//0(N*N)
2
3
   {
4
        for(int i = 1; i <= n; i++)</pre>
5
            for(int j = i-1; j >= 1; j--)
6
                 if(a[i] > a[j]) f[i] = max(f[i], f[j] + 1);
7
        int res = 0;
8
        for(int i = 1; i <= n; i++) res = max(res, f[i]);</pre>
9
        return res;
10
   }
11
   int c[N], len = 0;
12
   int LIS()//(NlogN)
13
14
        for(int i = 1; i <= n; i++)</pre>
15
16
        {
17
            //----find----
18
                 int l = 1, r = len, mid;
19
                 while(l <= r)</pre>
20
                 {
```

```
21
                     mid = (l + r) / 2;
22
                     if(a[i] > c[mid]) l = mid + 1;
                     else r = mid - 1;
23
                 }
24
            //----end---
25
            c[l] = a[i];
26
            len = max(len, l);
27
28
29
        return len;
30 | }
```

1.3 Maximum Continuous Subsequence Sum

```
int MaxSubSum()
1
2
3
        int f[N], res;
        for(int i = 1; i <= n; i++)</pre>
4
5
            f[i] = max(a[i], f[i-1] + a[i]);
6
7
            res = max(res, f[i]);
8
9
        return res;
10
11
   int MaxSubSum()
12
13
14
        int res = 0, now = 0;
        for(int i = 1; i <= n; i++)</pre>
15
16
17
            now += a[i];
            res = max(res, now);
18
19
            if(now < 0) now = 0;
20
21
        return res;
22 | }
```

1.4 数位 dp

```
int predoing(LL a, int *num)
1
2
3
       int le = 0;
       while(a)
4
5
6
            num[++le] = a % 10;
7
            a /= 10;
8
       }
9
       return le;
10
11
   int calc(int pos, int d, int u, int last)
   {
12
13
       if(pos == 0) return 1;
```

```
14
       int &res = f[pos][d][u][last];
       if (res !=-1) return res;
15
16
       res = 0;
       int st = d ? L[pos] : 0;
17
       int ed = u ? R[pos] : 9;
18
19
       for(int i = st; i <= ed; i++)</pre>
            if(合法) res += calc(pos - 1, d && i == L[pos], u && i == R[pos
20
21
       return res;
22 | }
```

2 Math

2.1 GCD && LCM

2.1.1 GCD - Greatest Common Divisor

```
1 | int gcd(int a, int b) { return b ? gcd(b, a % b) : a; }
```

2.1.2 LCM - Least Common Multiple

```
1 | inline int lcm(int a, int b) { return a / gcd(a, b) * b; }
```

2.1.3 E_GCD - Extended Greatest Common Divisor

```
ax + by = 1
   bx1 + (a\%b)y1 = 1 ==> bx + (a-a/b*b)y = 1
2
3
     ==> ay1 + b(x1-a/b*y1) = 1
4
   对应
        ax + by
5
   int egcd(int a, int b, int &x, int &y)
6
7
       if(b == 0)
8
9
       {
10
           x = 1; y = 0;
11
           return a;
12
13
       int x1, y1;
       int e = egcd(b, a%b, x1, y1);
14
15
       x = y1;
       y = x1 - a / b * y1;
16
17
       return e;
18 | }
```

2.2 Prime

2.2.1 Make Prime List

```
1
   void make_prime_list(int maxp) // O(2*N)
2
        for(int i = 2; i <= maxp; i++)</pre>
3
4
5
             if(!h[i]) pri[l++] = i;
             for(int j = 0; j < l && pri[j] <= maxp / i; j++)</pre>
6
7
             {
8
                 h[i * pri[j]] = true;
9
                 if(i % pri[j] == 0) break;
10
             }
        }
11
12 | }
   2.2.2 Prime Factor
1
   void factor()
2
3
        make_prime_list();
        for(int j = 0; j < Cnt && pri[j]*pri[j] <= n; j++)</pre>
4
5
6
             if(n % pri[j] == 0)
7
                 printf("%d<sub>□</sub>", pri[j]);
8
9
                 while(n % pri[j] == 0) n /= pri[j];
             }
10
11
12
        if(n!=1) printf("%d",n);
13 | }
   2.2.3 Fast Power
1 //x^y \% mod
   int mul(int x, LL y, int mod) // \mu \square \diamond
2
3
4
        if(y == 1) return x;
        if(y & 1) return (mul((x * (LL)x) % mod, y / 2, mod) * (LL)x) % mod
5
        else return mul((x * (LL)x) % mod, y / 2, mod) % mod;
6
7
8
   int mul(int x, int y, int mod) // \cdot g \square \Leftrightarrow
9
10
        int s = 1;
        int ss = x;
11
12
        while(y)
13
        {
14
             if(y \& 1) s = s * ss;
15
             y /= 2;
16
             ss *= ss;
17
18
        return s;
19 | }
```

3 Datastructure

3.1 Leftist Tree

```
1 // 很多时候需要配合并查集一起使用
  int getroot(int x){return f[x]==x ? x : f[x]=getroot(f[x]);}
2
3
  //把x和y合并在一起, 其实就是把y插入x
4
  int merge(int x,int y)//返回合并后子树的根
5
6
7
      if(!x \mid | !y) return x \mid y;
      if(A[x] < A[y]) swap(x,y);//大根堆,如果y比x大,与其让y插入x,
8
         不如让x插入y
      R[x]=merge(R[x],y);//始终往右子树合并
9
      f[R[x]] = x;//更新并查集
10
      if(D[R[x]] > D[L[x]]) swap(L[x],R[x]);//保持左偏树性质
11
      D[x] = D[R[x]] + 1;
12
      若还有其他维护信息也需要更新;
13
      return x;//返回根
14
15
  }
16
17
  int del(int x)
18
  {
19
      int t = merge(L[x],R[x]);
      f[L[x]] = L[x]; f[R[x]] = R[x];//更新并查集
20
21
      L[x] = R[x] = D[x] = 0;
22
      return t;
23 | }
```

3.2 Partition Tree

```
struct Parti{int val, left;} val[30][N];
   void build tree(int d, int l, int r)
2
   {
3
4
       if(l == r) return;
5
       int m = (l + r) >> 1, same = m - l + 1;
       int lcnt = l, rcnt = m + 1;
6
7
       for(int i = l; i <= r; i++)</pre>
            if(val[d][i].val < sorted[m]) same--;</pre>
8
9
       for(int i = l; i <= r; i++)</pre>
10
            int flag = 0;
11
            if((val[d][i].val < soted[m]) || (val[d][i].val == sorted[m] &&
12
                same))
            {
13
                flag = 1;
14
                val[d + 1][lcnt++] = val[d][i];
15
                if(val[d][i].val == sorted[m]) same--;
16
17
18
            else val[d][rcnt++] = val[d][i];
19
            val[d][i].left = val[d][i - 1].left + flag;
20
       }
```

```
21
       build_tree(d + 1, l, m);
22
       build_tree(d + 1, m + 1, r);
23
   int query(int d, int l, int r, int x, int y, int k)
24
25
26
       if(l == r) return val[d][l].val;
27
       int m = (l + r) >> 1;
28
       int lx = val[d][x - 1].left - val[d][l - 1].left; //[l,x-1] to left
       int ly = val[d][y].left - val[d][x - 1].left; //[x,y] to left
29
30
       int rx = (x - 1 - l + 1) - lx; //[l, x-1] to right
       int ry = (y - x + 1) - ly; //[x,y] to right
31
       if(ly >= k) return query(d+1, l, m, l-1+lx+1, l-1+lx+ly, k);
32
33
       else return query(d+1, m+1, r, m+1-1+rx+1, m+1-1+rx+ry, k-ly);
34 | }
```

3.3 Size Balanced Tree

```
int A[N], S[N], L[N], R[N], root, total;
2
   void rotate_left(int &x)
3
4
       int y = R[x];
       R[x] = L[y];
5
       L[y] = x;
6
7
       S[y] = S[x];
8
       S[x] = S[L[x]] + S[R[x]] + 1;
9
       x = y;
10
11
   void rotate_right(int &x)
12
       int y = L[x];
13
14
       L[x] = R[y];
15
       R[y] = x;
       S[y] = S[x];
16
       S[x] = S[L[x]] + S[R[x]] + 1;
17
18
       x = y;
19
20
21
   void maintain(int &p, bool flag)
   {
22
       if(flag)//调整右边
23
24
25
            if(S[R[R[p]]] > S[L[p]] rotate_left(p);
26
                     else if(S[R[L[p]]] > S[L[p]])
27
            {
28
                rotate_right(R[p]);
29
                     rotate_left(p);
30
                }
31
                else return;
32
                }
33
       else
34
       {
35
            if(S[L[L[p]]] > S[R[p]]) rotate_right(p);
```

```
36
            else if(S[L[R[p]]] > S[R[p]])
37
                rotate_left(L[p]);
38
39
                rotate_right(p);
40
            }
41
            else return;
42
        }
43
        maintain(L[p], 0);
44
        maintain(R[p], 1);
45
        maintain(p, 0);
46
        maintain(p, 1);
47
48
49
   void insert(int &p, int e)
50
51
       if(!p)
        {
52
53
            p = ++total;
54
            L[p] = R[p] = 0;
            A[p] = e; S[p] = 1;
55
56
            return;
57
        }
        S[p]++;
58
59
       if(e < A[p]) insert(L[p], e);
       else insert(R[p], e);
60
        maintain(p, k >= A[p]);
61
62
63
64
   int getmin()
65
        for(int x = root; L[x]; x = L[x]);
66
        return A[x];
67
68
69
   int getmax()
70
71
        for(int x = root; R[x]; x = R[x]);
72
        return A[x];
73
   int kth(int &p, int k)
74
75
76
        int tmp = S[L[p]] + 1;
77
        if(k == tmp) return A[p];
78
        else if(k < tmp) return kth(L[p], k);</pre>
79
        else return kth(R[p], k - tmp);
80
  | }
```

4 Graph

4.1 Shortest path

4.1.1 Dijkstra

```
1
   void dijkstra()
2
       memset(dist, 0, sizeof(dist));
3
4
       while(!Q.empty())
5
       {
            int x = Q.top().second; Q.pop();
6
7
           if(done[x]) continue;
8
            done[x] = 1;
9
            for(Link p = head[x]; p; p = p->next)
10
                if(dist[p->y] > dist[x] + p->z)
                {
11
                    dist[p->y] = dist[x] + p->z;
12
                    Q.push(make_pair(dist[p->y], p->y));
13
                }
14
15
       }
16 | }
   4.1.2 Spfa
1
   void spfa()
2
       memset(dist, 0x3f, sizeof(dist));
3
       Q.push(S);//S为源点
4
 5
       while(!Q.empty())
6
       {
7
            int x = Q.front();
8
           Q.pop(); inQ[x] = 0;
9
            for(Link p = head[x]; p; p = p->next)
                if(dist[p->y] > dist[x] + p->z)
10
11
                {
12
                    dist[p->y] = dist[x] + p->z;
13
                    if(!inQ[p->y])
14
                    {
15
                         Q.push(p->y);
                         inQ[p->y] = 1;
16
                    }
17
18
                }
19
       }
20
   }
   4.1.3 Floyd
   void floyd()
1
2
   {
3
       for(int k = 1; k <= n; k++) // 这里可以看作是一个加边的过程
4
            for(int i = 1; i <= n; i++)</pre>
 5
                for(int j = 1; j <= n; j++)</pre>
6
                    map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
7
   }
8
   // 最小环
```

```
10
   void MinCircle()
11
   {
12
        cap[] = map[];
        int circle = 0x3f3f3f3f;
13
        for(int k = 1; k <= n; k++)</pre>
14
15
            for(int i = 1; i < k; i++)</pre>
16
17
                for(int j = i+1; j < k; j++)</pre>
                     circle = min(circle, map[i][j] + cap[j][k]+cap[k][i]);
18
19
            for(int i = 1; i <= n; i++)</pre>
                for(int j = 1; j <= n; j++)</pre>
20
21
                     map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
22
        return circle == 0x3f3f3f3f ? −1 : circle;
23
24
   }
25
   // floyd判圈法 (大白书 p44)
26
   void Circle()
27
28
29
        int ans = k;
30
        int k1 = k, k2 = k;
31
        do{
32
            k1 = next(k1);
33
            k2 = next(k2); ans = max(ans, k2);
            k2 = next(k2); ans = max(ans, k2);
34
35
        }while(k1 != k2);
        return ans;
36
37
  |}
```

4.2 Minimum Spanning Tree

4.2.1 Prim

```
void prime()
1
2
   {
       memset(dist, 0, sizeof(dist));
3
       int res = 0;
4
5
       while(!Q.empty())
       {
6
7
            int x = Q.top().second;
            if(done[x]) {Q.pop(); continue;}
8
9
            res += Q.top().first;
            Q.pop();
10
            for(Link p = head[x]; p; p = p->next)
11
                if(dist[p->y] > p->z)
12
                {
13
                     dist[p->y] = p->z;
14
                     Q.push(make_pair(dist[p->y], p->y));
15
                }
16
       }
17
18 | }
```

4.2.2 Kruskal

```
void prime()
2
3
       sort(edge, edge+Cnt, cmp);
4
       int res = 0;
5
       for(int i = 0; i < Cnt; i++)</pre>
6
7
            if(getroot(edge[i].x) == getroot(edge[i].y)) continue;
8
            merge(edge[i].x, edge[i].y);
9
            res += edge[i].z;
       }
10
11 | }
```

4.3 Tarjan - Strong Union

```
1
   void dfs(int x)
2
       now[x] = low[x] = ++dfstime;
3
       hash[x] = 1;
4
5
       st.push(x); inst[x] = 1;
       for(int i = 1; i <= n; i++)</pre>
6
7
            if(map[x][i])
8
            {
9
                if(!hash[i])
10
                {
                    dfs(i);
11
                    low[x] = min(low[x], low[i]);
12
13
                else if(inst[i]) low[x] = min(low[x], now[i]);
14
15
       if(low[x] == now[x])
16
17
18
            while(!st.empty())
19
20
                int u = st.top();
                st.pop(); inst[u] = 0;
21
22
                belong[u] = number;
                if(u == x) break;
23
24
25
            numer++;
26
       }
27
28
   void tarjan()
29
   {
       for(int i = 1; i <= n; i++)</pre>
30
            if(!hash[i]) dfs(i);
31
       if(!st.empty()) // 这是一个未知 bug 栈中还会剩下一个强连通分量
32
33
34
            while!st.empty())
35
            {
36
                int u = st.top();
```

4.4 LCA

4.4.1 Tarjan

```
1 // poj 1330 (changed something)
   // LCA tarjan
3
   #include <cstdio>
4
   #include <cstring>
5
6
   const int N = 10000 + 10;
7
8
   int n;
   struct Link{int y, idx; Link *next;}*head[N], *ask[N];
9
10
   int tx, ty;
11
   bool in[N], vis[N];
   int f[N];
12
   int ans[N]; // Query Answer
13
14
15
   void inLink(int x, int y)
   {
16
17
        Link *p = new Link;
18
        p \rightarrow y = y;
19
        p -> next = head[x];
20
        head[x] = p;
21
22
   void inAsk(int x, int y, int idx)
23
24
        Link *p = new Link;
25
        p \rightarrow y = y;
        p \rightarrow idx = idx;
26
27
        p \rightarrow next = ask[x];
28
        ask[x] = p;
29
30
31
   int getroot(int x)
32
33
        return f[x] == x ? x : f[x] = getroot(f[x]);
34
35
36
   void LCA(int x)
37
38
        vis[x] = 1;
39
        f[x] = x;
40
        for(Link *p = ask[x]; p; p = p \rightarrow next)
            if(vis[p->y]) ans[p->idx] = getroot(p->y);
41
```

```
42
        for(Link *p = head[x]; p; p = p -> next)
43
            if(!vis[p->y])
44
            {
45
                 LCA(p->y);
46
                 f[p->y] = x;
            }
47
48
49
   int main()
50
51
   {
        int T; scanf("%d", &T);
52
        while (T--)
53
54
        {
            memset(head, 0, sizeof(head));
55
56
            memset(ask, 0, sizeof(ask));
57
            memset(in, 0, sizeof(in));
58
            memset(vis, 0, sizeof(vis));
59
            scanf("%d", &n);
            for(int i = 1; i <= n; i++) f[i] = i;</pre>
60
            for(int i = 1; i < n; i++)</pre>
61
62
            {
63
                 int x, y;
                 scanf("%d%d", &x, &y);
64
65
                 inLink(x, y);
                 in[y] = 1;
66
            }
67
            int q = 1;// the number of query
68
69
            for(int i = 1; i <= q; i++)</pre>
70
            {
71
                 int x, y; scanf("%d%d", &x, &y);
72
                 inAsk(x, y, i); inAsk(y, x, i);
73
            int root = -1;
74
75
            for(int i = 1; i <= n; i++)</pre>
76
                 if(!in[i]) {root = i; break;}
77
            LCA(root);
78
            for(int i = 1; i <= q; i++)</pre>
79
                 printf("%d\n", ans[i]);
80
81
        return 0;
82 | }
```

4.4.2 Doubling Algorithm

还不会...

4.5 Bipartite Graph

4.5.1 Maximal Matching - The Hungarian algorithm

```
1 | int ttt = 0; // 全局时间戳变量
```

```
3
   bool search(int x)
4
        for(int i = 1; i <= m; i++)</pre>
5
            if(map[x][i] && vis[i] != ttt)
6
7
            {
                 vis[i] = ttt;
8
                 if(pre[i] == -1 \mid | search(pre[i]))
9
10
                      pre[i] = x;
11
                      return 1;
12
                 }
13
            }
14
15
        return 0;
16
17
18
   int match()
19
   {
        int res = 0;
20
        for(int i = 1; i <= n; i++)</pre>
21
22
23
            ++ttt; // 这里不用 memset 节省时间
24
            res += search(i);
25
26
        return res;
27
   }
```

4.5.2 Optimal Matching - KM

不会... 用费用流解决

4.6 Network flow

4.6.1 Maximum Flow - isap

```
//
        h[x]:
                  点 x 在第 h[x] 层
1
2
        v[k]:
                  第 k 层有 v[k] 个点
   int sap(int x, int flow)
3
4
   {
5
       if(x == n) return flow;
6
       int res = 0;
7
       for(int i = S; i <= T; i++)</pre>
           if(g[x][i] && h[x] == h[i] + 1)
8
9
           {
                int t = sap(i, min(g[x][i], flow - res));
10
                res += t; g[x][i] -= t; g[i][x] += t;
11
                if(res == flow) return res;
12
13
                if(h[S] >= T) return res;
14
15
       //if(h[S] >= T) return res;
16
       if((--v[h[x]]) == 0) h[S] = T;
17
       ++v[++h[x]];
       return res;
18
```

```
19
  | }
   int main()
20
21
   {
       v[0] = T;
22
23
       int maxflow = 0;
       while(h[S] < T) maxflow += sap(1, inf);</pre>
24
25
       reutrn 0;
26 | }
   4.6.2 Minimum Cost Maximum Flow - spfa
   struct EG{int from, to, flow, cost, next;}edge[M];
1
2
   void add_edge(int a,int b,int c,int d)
3
4
5
       edge[L]=(EG){a,b,c,+d,head[a]};
6
       head[a]=L++;
       edge[L]=(EG){b,a,0,-d,head[b]};
7
       head[b]=L++;
8
9
10
   bool spfa()
11
12
13
       memset(inQ, 0, sizeof(inQ));
       memset(dist, 0x3f, sizeof(dist));
14
15
       dist[S] = 0;
       q.push(S);
16
       while(!q.empty())
17
18
19
            int x = q.front();
20
            q.pop();
21
            inQ[x] = 0;
            for(int i = head[x]; i != -1; i = edge[i].next)
22
                if(edge[i].flow && dist[edge[i].to] > dist[x] + edge[i].
23
                   cost)
                {
24
25
                     pre[edge[i].to] = i;
26
                     dist[edge[i].to] = dist[x] + edge[i].cost;
27
                     if(!inQ[edge[i].to])
28
29
                         inQ[edge[i].to] = 1;
30
                         q.push(edge[i].to);
31
                     }
                }
32
33
34
       return dist[T] != inf;
35
   void MFMC()
36
37
       memset(head, -1, sizeof(head));
38
        建图调用 add_edge();
39
40
```

```
41
       int mincost = 0, maxflow = 0;
42
       while(spfa())
43
       {
44
            int res = inf;
            for(int i = T; i != S; i = edge[pre[i]].from)
45
            {
46
                res = min(res, edge[pre[i]].flow);
47
48
49
            for(int i = T; i != S; i = edge[pre[i]].from)
50
                edge[pre[i]].flow -= res;
51
52
                edge[pre[i] ^ 1].flow += res;
53
            maxflow += res;
54
55
            mincost += res * dist[T];
56
       }
57 | }
```

5 Geometry

5.1 Convex Hull

```
1 |//♦♦□°♦ μ♦list[0~n−1]
   //□°�%��stack[0~top-1]
2
  |Point list[Maxn];
3
   int Stack[Maxn],top;
   bool _cmp (Point p1,Point p2)
5
6
7
       double tmp=(p1-list[0])^(p2-list[0]);
       if (fuhao(tmp)>0) return true;
8
       else if (fuhao(tmp)==0&&fuhao(dist(p1,list[0])-dist(p2,list[0]))
9
           <=0)
10
            return true;
11
       else
                return false;
12
   void Graham(int n)
13
14
       Point p0;
15
16
       int k=0;
17
       p0=list[0];
18
       for (int i=1;i<n;++i)
19
            if ((p0.y>list[i].y)||(p0.y==list[i].y&&p0.x>list[i].x))
20
21
            {
                p0=list[i];
22
23
                k=i;
24
            }
25
       }
26
       swap(list[k],list[0]);
27
       sort(list+1,list+n,_cmp);
28
       if (n==1)
```

GuessEver

```
ACM-ICPC Template
29
        {
30
             top=1;
31
             stack[0]=0;
32
             return;
33
        }
        if (n==2)
34
35
        {
36
             top=2;
37
             stack[0]=0;
38
             stack[1]=1;
39
             return;
40
        }
        stack[0]=0;
41
42
        stack[1]=1;
43
        top=2;
44
        for (int i=2;i<n;++i)</pre>
45
        {
             while (top>1 && fuhao((list[stack[top-1]]-list[stack[top-2]])^(
46
                list[i]-list[stack[top-2]]) <= 0
47
                 top--;
48
             stack[top++]=i;
49
        }
50
   }
```

5.2 All

```
1 #include <cstdio>
  #include <cstdlib>
  #include <cstring>
3
  #include <cmath>
4
5
  #include <algorithm>
  #include <utility>
6
7
  using std::max;
  using std::min;
8
9
   using std::sort;
   using std::swap;
10
  using std::pair;
11
12
   using std::make_pair;
   const double eps = 1e-8, inf = 1e20;
13
   const double pi = 4.0 * atan(1.0);
14
  #define Degree(_rad) (180.0 / pi * (_rad))
15
16
17
   int fuhao(double x)
18
19
       if (fabs(x)<eps) return 0;</pre>
20
       if (x<0) return -1;
       else return 1;
21
22
   }
23
   24
                                Point && Vector
     25
  struct Point{
```

```
26
       double x, y;
27
       Point (){}
28
       Point (double _x,double _y):x(_x),y(_y){}
       void init(double a, double b) { x = a; y = b; }
29
30
       // basic calc
31
32
           bool operator == (const Point &b) const
33
               return !fuhao(x - b.x) && !fuhao(y - b.y);
34
35
           Point operator + (const Point &b) const
36
37
               return Point(x + b.x, y + b.y);
38
39
           }
40
           Point operator - (const Point &b) const
41
               return Point(x - b.x, y - b.y);
42
43
           Point operator * (const double &b) const
44
45
               return Point(x * b, y * b);
46
47
           }
48
49
           Point Rotate(Point p, double alpha) // alpha E [0, +oo) 逆时针
50
51
               double x0 = p.x, y0 = p.y;
52
               double tx = x - x0, ty = y - y0;
53
               double nx = tx * cos(alpha) - ty * sin(alpha);
54
               double ny = tx * sin(alpha) + ty * cos(alpha);
55
               nx += x0; ny += y0;
               return Point(nx, ny);
56
           }
57
58
       // Vector
59
           double operator *(const Point &b)const
60
61
           {// Dot
               return x * b.x + y * b.y;
62
63
           double operator ^ (const Point &b)const
64
           {// Cross
65
               return x * b.y - y * b.x;
66
67
68
           double Abs() { return sqrt(x * x + y * y); }
69
   double Dist(const Point &a, const Point &b) { return (a - b).Abs(); }
70
   typedef Point Vector;
71
72
73
   double Angle(Vector a, Vector b)
74
       return acos(a * b / a.Abs() / b.Abs());
75
76
77
   Vector Get_H(Vector A)
   { // 求与向量垂直的单位向量
                                  使用前确保不为0向量
78
```

```
79
       // A != Vector(0.0, 0.0);
       double L = A.Abs();
80
       return Vector(-A.y / L, A.x / L);
81
82
83
                                 E - N - D
84
   85
86
87
   Line
      struct Line{
88
89
       Point s,e;
90
       Line() {}
91
       Line(Point ss, Point ee)
92
93
          s = ss; e = ee;
94
       }
95
       // 两直线的关系: 重合0,
                              平行1,
                                     相交2 并返回交点
96
97
       pair < int , Point > operator & (const Line &b) const
98
99
          Point ans = s;
          if(fuhao((s-e)^(b.s-b.e))==0)
100
101
              if (fuhao((s-b.e)^(b.s-b.e))==0)
102
                  return make_pair(0,ans);//重合
103
104
              else return make_pair(1,ans);//平行
105
           }
106
           double t = ((s-b.s)^(b.s-b.e)) / ((s-e)^(b.s-b.e));
           ans.x += (e.x-s.x) * t;
107
           ans.y += (e.y-s.y) * t;
108
           return make_pair(2,ans);//相交
109
110
       }
111
   };
112
   E - N - D
      113
   //判断线段相交
114
   bool inter(Line l1, Line l2)
115
116
117
       return
       max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
118
119
       \max(l1.s.y, l1.e.y) >= \min(l2.s.y, l2.e.y) &&
       max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
120
       \max(l2.s.y, l2.e.y) >= \min(l1.s.y, l1.e.y) &&
121
       fuhao((l2.s-l1.e)^{(l1.s-l1.e)}) * fuhao((l2.e-l1.e)^{(l1.s-l1.e)}) <= 0
122
123
       fuhao((l1.s-l2.e)^{(l2.s-l2.e)}) * fuhao((l1.e-l2.e)^{(l2.s-l2.e)} <=0;
124
   // 判断直线与线段相交
125
126
   bool Seg_inter_line(Line l1,Line l2)//l1为直线 l2为线段
127
```

```
128
        return fuhao((l2.s-l1.e)^(l1.s-l1.e))*fuhao((l2.e-l1.e)^(l1.s-l1.e)
129
    //点到直线距离
130
    //返回点到直线最近的点
131
    Point PointToLine(Point P,Line L)
132
133
134
        Point ans;
        double t=((P-L.s)*(L.e-L.s))/((L.e-L.s)*(L.e-L.s));
135
136
        ans.x=L.s.x+(L.e.x-L.s.x)*t;
        ans.y=L.s.y+(L.e.y-L.s.y)*t;
137
138
        return ans;
139
    //点到线段距离
140
    //返回点到线段最近的点
141
    Point NearestPointToLineSeg(Point P,Line L)
142
143
    {
144
        Point ans;
145
        double t = ((P-L.s)*(L.e-L.s)) / ((L.e-L.s)*(L.e-L.s));
        if (t>=0\&\&t<=1)
146
147
        {
148
            ans.x = L.s.x + (L.e.x-L.s.x)*t;
149
            ans.y = L.s.y + (L.e.y-L.s.y)*t;
150
        }
        else {
151
152
            if (Dist(P,L.s)<Dist(P,L.e))</pre>
153
                ans = L.s;
154
            else
                     ans = L.e;
155
        }
156
        return ans;
157
    //多边形面积
158
159
    double CalcArea(Point p[],int n)
160
        double ans=0;
161
162
        for (int i=0;i<n;++i)</pre>
            ans+=(p[i]^p[(i+1)\%n])/2;
163
164
        return fabs(ans);
165
    // 判断点在线段上
166
167
    bool OnSeg(Point P,Line L)
168
169
        return
                fuhao((L.s-P)^(L.e-P))==0 &&
170
                fuhao((P.x-L.s.x)*(P.x-L.e.x))<=0 &&
171
                fuhao((P.y-L.s.y)*(P.y-L.e.y))<=0;
172
173
    //三点求圆心坐标
174
175
    Point waixin(Point a, Point b, Point c)
176
177
        double a1=b.x-a.x,b1=b.y-a.y,c1=(a1*a1+b1*b1)/2;
        double a2=c.x-a.x,b2=c.y-a.y,c2=(a2*a2+b2*b2)/2;
178
        double d=a1*b2-a2*b1;
179
```

```
180
        return Point(a.x+(c1*b2-c2*b1)/d,a.y+(a1*c2-a2*c1)/d);
181
   }
182
183
    184
                                        Graham
       //求凸包 点list[0~n-1]
185
186
    const int Maxn = 100;/////////////here!!
187
                                /////////?!?!?!?! 补全 Maxn
188
    Point list[Maxn];
       !?!?!?!?!?!?!?!?!?!?!
189
    int Stack[Maxn],top;
190
    bool _cmp (Point p1,Point p2)
191
192
        double tmp=(p1-list[0])^(p2-list[0]);
193
        if (fuhao(tmp)>0) return true;
194
        else if (fuhao(tmp)==0&&fuhao(Dist(p1,list[0])-Dist(p2,list[0]))
           <=0)
195
            return true;
196
        else
                return false;
197
198
    void Graham(int n)
199
200
        Point p0;
201
        int k=0;
202
        p0=list[0];
        for (int i=1;i<n;++i)</pre>
203
204
205
            if ((p0.y>list[i].y)||(p0.y==list[i].y&&p0.x>list[i].x))
206
207
                p0=list[i];
208
                k=i;
209
            }
210
211
        swap(list[k],list[0]);
212
        sort(list+1,list+n,_cmp);
213
        if (n==1)
214
        {
215
            top=1;
216
            Stack[0]=0;
217
            return;
218
        }
        if (n==2)
219
220
221
            top=2;
222
            Stack[0]=0;
223
            Stack[1]=1;
224
            return;
225
        }
226
        Stack[0]=0;
227
        Stack[1]=1;
228
        top=2;
229
        for (int i=2;i<n;++i)</pre>
```

230

231

```
232
233
234
235
236
237
238
239
240
241
242
       for(int i = 1; i < nn-1; i++)</pre>
243
244
          ans_area += (pp[i] - pp[0]) ^ (pp[i+1] - pp[0]);
245
246
247
       return fabs(ans_area / 2);
248
249
   E - N - D
     250
   点在多边形内
251
      252
   int isPointInPolygon(Point p, Point *poly, int nn)
253
   {
254
       int w = 0;
       for(int i = 0; i < n; i++)</pre>
255
256
257
          if(OnSeg(p, Line(poly[i], poly[(i+1)%n]))) return -1; // 边界上
258
          int k = fuhao((poly[(i+1)%n] - poly[i]) ^ (p - poly[i]));
          int d1 = fuhao(poly[i].y - p.y);
259
260
          int d2 = fuhao(poly[(i+1)%n].y - p.y);
          if(k > 0 \&\& d1 <= 0 \&\& d2 > 0) wn++;
261
          if(k < 0 && d1 > 0 && d2 <= 0) wn--;
262
263
       if(wn != 0) return 1; //内部
264
265
       return 0; // 外部
266
267
   E - N - D
      268
269
270
   int main()
271
272
   }
```

6 String Tools

6.1 KMP

```
void getnext(char *t)
2
3
        memset(next, 0, sizeof(next));
4
        next[0] = -1;
5
        int j = 0, k = -1;
        int len = strlen(t);
6
7
        while(j < len)</pre>
        {
8
9
            if(k == -1 \mid | t[j] == t[k]) next[++j] = ++k;
10
            else k = next[k];
11
        }
12
   void kmp()
13
14
15
        getnext();
16
        int lens = strlen(s), lent = strlen(t);
17
        int i = 0, j = 0;
        while(i < lens)</pre>
18
19
        {
            if(j == -1 \mid | s[i] == t[j]) i++, j++;
20
21
            else j = next[j];
            if(j == lent) return j - lent;
22
23
        }
24
        return -1;
25 | }
```

6.2 Suffix Array

```
1 | int sa[N], rank[N];
   int c[N], tmp[N];
2
   int height[N];
   bool cmp(int *r, int a, int b, int l)
4
5
6
       return r[a] == r[b] && r[a + l] == r[b + l];
7
   void DA(int n, int m) //n为长度, m为字串的最大值。 均为开区间
8
9
       int i, j, p, *x = rank, *y = tmp;
10
       memset(c, 0, sizeof(c));
11
       for(i = 0; i < n; i++) c[ x[i] = s[i] ] ++;</pre>
12
13
       for(i = 1; i < m; i++) c[i] += c[i - 1];
       for(i = n - 1; i \ge 0; i - - c[x[i]] = i;
14
       for(j = 1, p = 0; p < n; j *= 2, m = p)
15
16
       {
           for(p = 0, i = n - j; i < n; i++) y[p++] = i;
17
18
           for(i = 0; i < n; i++) if(sa[i] >= j) y[p++] = sa[i] - j;
19
           memset(c, 0, sizeof(c));
20
           for(i = 0; i < n; i++) c[ x[y[i]] ]++;</pre>
```

```
21
           for(i = 1; i < m; i++) c[i] += c[i - 1];
22
           for(i = n - 1; i \ge 0; i--) sa[ --c[x[y[i]]] ] = y[i];
           swap(x, y); x[sa[0]] = 0;
23
24
           for(i = 1, p = 1; i < n; i++)
               if(x[i]) x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p
25
                  ++;
26
27
       for(i = 0; i < n; i++) rank[sa[i]] = i;</pre>
       int k = 0;
28
       for(i = 0; i < n; height[x[i++]] = k)</pre>
29
           for (k ? k - : 0, j = sa[x[i] - 1]; s[i + k] == s[j + k]; k++);
30
31
   int main()
32
33
   {
       读入字串s, n=s的长度
34
       s[n++]='$';//最后加入一个不存在的值, 排除超界情况
35
36
       DA(n,m);
37 | }
```

7 Tools

7.1 BigInteger - C++

```
1 //程序中全部为正整数之间的操作
   #include <cstdio>
   #include <cstring>
3
   #include <algorithm>
4
   using std::max;
5
6
7
   const int base = 10000; // 压4位
8
9
   struct BigInt{
       int c[1000], len, sign;
10
       BigInt() { memset(c, 0, sizeof(c)); len = 1; sign = 0; }
11
12
       void Zero()
13
       {
           while(len > 1 && c[len] == 0) len--;
14
           if(len == 1 && c[len] == 0) sign = 0;
15
16
17
       void writein(char *s)
18
       {
19
           int k = 1, L = strlen(s);
           for(int i = L-1; i >= 0; i--)
20
21
           {
22
               c[len] += (s[i]-'0') * k;
23
               k *= 10;
24
               if(k == base)
25
               {
26
                   k = 1;
27
                    len++;
               }
28
```

```
29
30
        void Read()
31
32
33
            char s[5000] = \{0\};
            scanf("%s", s);
34
35
            writein(s);
36
        }
        void Print()
37
38
            if(sign) printf("-");
39
            printf("%d", c[len]);
40
            for(int i = len-1; i >= 1; i--) printf("%04d", c[i]);
41
            printf("\n");
42
43
        BigInt operator = (int a)
44
45
            char s[100] = \{0\};
46
            sprintf(s, "%d", a);
47
48
            writein(s);
49
            return *this;
50
        }
51
        bool operator > (const BigInt &b)
52
            if(len != b.len) return len > b.len;
53
            for(int i = len; i >= 1; i--)
54
55
56
                 if(c[i] != b.c[i]) return c[i] > b.c[i];
57
58
            return 0;
59
        bool operator < (const BigInt &b)</pre>
60
61
            if(len != b.len) return len < b.len;</pre>
62
            for(int i = len; i >= 1; i--)
63
64
                 if(c[i] != b.c[i]) return c[i] < b.c[i];</pre>
65
66
67
            return 0;
68
        bool operator == (const BigInt &b)
69
70
            if(len != b.len) return 0;
71
72
            for(int i = 1; i <= len; i++)</pre>
                 if(c[i] != b.c[i]) return 0;
73
74
            return 1;
75
        }
76
        bool operator == (const int &a)
77
        {
78
            BigInt b; b = a;
79
            return *this == b;
80
        BigInt operator + (const BigInt &b)
81
```

```
82
         {
             BigInt r; r.len = max(len, b.len) + 1;
83
             for(int i = 1; i <= r.len; i++)</pre>
84
             {
85
86
                  r.c[i] += c[i] + b.c[i];
                  r.c[i+1] += r.c[i] / base;
87
88
                  r.c[i] %= base;
89
90
             r.Zero();
91
             return r;
         }
92
93
         BigInt operator + (const int &a)
94
95
             BigInt b; b = a;
96
             return *this + b;
97
98
         BigInt operator - (const BigInt &b)
99
100
             BigInt a, c;// a - c
             a = *this; c = b;
101
102
             if(a < c)
103
             {
104
                  std::swap(a, c);
105
                  a.sign = 1;
106
107
             for(int i = 1; i <= len; i++)</pre>
108
109
                  a.c[i] -= c.c[i];
110
                  if(a.c[i] < 0)
111
112
                      a.c[i] += base;
                      a.c[i+1]--;
113
114
                  }
115
116
             a.Zero();
117
             return a;
118
119
         BigInt operator - (const int &a)
120
121
             BigInt b; b = a;
122
             return *this - b;
123
         }
         BigInt operator * (const BigInt &b)
124
125
             BigInt r; r.len = len + b.len + 2;
126
             for(int i = 1; i <= len; i++)</pre>
127
128
                  for(int j = 1; j <= b.len; j++)</pre>
129
130
                      r.c[j+i-1] += c[i] * b.c[j];
131
132
                  }
133
             for(int i = 1; i <= r.len; i++)</pre>
134
```

```
135
            {
                 r.c[i+1] += r.c[i] / base;
136
                 r.c[i] %= base;
137
138
139
            r.Zero();
140
             return r;
141
        }
142
        BigInt operator * (const int &a)
143
        {
144
             BigInt b; b = a;
145
             return *this * b;
146
        BigInt operator / (BigInt b)//整除
147
148
149
             BigInt t, r;
             if(b == 0) return r;
150
151
             r.len = len;
             for(int i = len; i >= 1; i--)
152
153
154
                 t = t * base + c[i];
155
                 int div;
156
                 //--
                        -try-
                     int up = 10000, down = 0;
157
158
                     while(up >= down)
159
                     {
160
                          int mid = (up + down) / 2;
                          BigInt ccc ; ccc = b * mid;
161
162
                          if(ccc > t) up = mid - 1;
                          else {
163
164
                              down = mid + 1;
165
                              div = mid;
                          }
166
167
                     }
                 //---end----
168
169
                 r.c[i] = div;
170
                 t = t - b * div;
171
             }
             // 最后的 t 为 余数 , 要 用 的 自 己 想 办 法 传 出 去
172
             r.Zero();
173
174
             return r;
175
176
        BigInt operator / (const int &a)
177
        {
178
             BigInt b; b = a;
             return *this / b;
179
        }
180
181
        BigInt operator % (const BigInt &b)
        {//其实可以复制上面除法的,这里换一种写法
182
183
             return *this - *this / b * b;
184
185
        BigInt operator % (const int &a)
        {
186
             BigInt b; b = a;
187
```

7.2 C char*

7.3 C++ std::string

```
1 //====初始化 ====
  头文件string并加上std::
  string s(str);//相当于 string s=str;
  string s(cstr);//把char数组类型的字符串cstr作为s的初值
  s.clear();//清空, 相当于 s="";
5
6
  //==== 长度====
7
  s.length();//获取s的长度,0(1)
  s.size();//一样
9
10
  //====插入删除 ====
11
  s.insert(2, "a"); //在s的位置 2插入 string 类字符串 "a"
12
  s.erase(2, 3); //从s的位置2开始删除3个字符
13
14
  //====查找 ====
15
  |s.find("abc");//查找字符串"abc"在s中第一次出现的位置(据说是KMP实现的)
  //s="aabcc"; printf("%d %d\n",(int)s.find("abc"),(int)s.find("aabb"));
17
18 // 上一行程序应输出 1 -1 (若没找到必须强行转换为 int 才为 -1 )
```

7.4 Batch test

7.4.1 @Linux

```
1 | mkdata=mk
2 | filea=a
3 | fileb=b
4 |
5 | g++ $mkdata.cpp -o $mkdata
```

```
6 g++ $filea.cpp —o $filea
7
   g++ $fileb.cpp -o $fileb
   cas=0
8
   while true; do
9
       ./$mkdata > $filea.in
10
       ./$filea < $filea.in > $filea.out
11
       ./$fileb < $filea.in > $fileb.out
12
       if ! diff $filea.out $fileb.out
13
14
       then
15
            echo "⊔Wrong⊔Answer"
16
            break
17
       fi
       echo ((cas=cas+1)) "_{\sqcup}Accepted"
18
   done
19
```

7.4.2 @Windows

```
:loop
1
2
      mk > A.in
      A < A.in > A.out
3
4
      p < A.in > p.out
5
      fc A.out p.out
      if errorlevel 1 goto end
6
7
      goto loop
8
  :end
9
      pause
```