ACM-ICPC Template



GuessEver

Last Modify at November 9, 2015

Contents

1	Dyna	mic Programming
	-	LCS - Longest Common Subsequence
		LIS - Longest Increasing Subsequence
		Maximum Continuous Subsequence Sum
		RMQ - st
		数位 dp
		状压 dp
		1.6.1 枚举子集
2	Math	
	2.1	GCD && LCM
		2.1.1 GCD - Greatest Common Divisor
		2.1.2 LCM - Least Common Multiple
		2.1.3 E_GCD - Extended Greatest Common Divisor
	2.2	Prime
		2.2.1 Make Prime List
		2.2.2 Prime Factor
	2.3	Fast Power
	2.4	约瑟夫环、丢手绢问题
	2.5	康拓展开 Cantor
3		structure .
		带权并查集
	3.2	手写 Heap
	3.3	Leftist Tree
		Partition Tree
	3.5	Treap
		3.5.1 @ Array
		3.5.2 @ Pointer
	3.6	Size Balanced Tree
	3.7	树链剖分 Heavy-Light Decomposition
	3.8	三维偏序 - CDQ 分治
4	Grap	oh 2:
-	•	Shortest path
	4.1	4.1.1 Dijkstra
		·
		4.1.2 Spfa
	4.2	Minimum Spanning Tree
	4.2	· · · ·
		4.2.1 Prim
	1 2	
		Tarjan - Strong Union
	4.4	LCA
		4.4.1 @ Tarjan
	4 -	4.4.2 @ Doubling Algorithm
	4.5	Bipartite Graph
		4.5.1 Maximal Matching - The Hungarian algorithm
	4 -	4.5.2 Optimal Matching - KM
	4.6	Network Flow
		4.6.1 Maximum Flow - isap
		4.6.2 Minimum Cost Maximum Flow - spfa

	M-ICPC Template GuessEver				
5	Geom	-	33		
			33		
			38		
	5.3	Halfplane Intersection	39		
6	String				
	6.1	HASH	11		
	6.2	Minimum Representation - 最小表示法	11		
	6.3	Manacher	11		
	6.4	KMP	13		
	6.5	Suffix Array	13		
	6.6	Aho-Corasick Automaton	15		
7	Tool	.s 4	17		
	7.1	BigInteger - C++	17		
			51		
	7.3	C char*	51		
	7.4		52		
			52		
		7.5.1 The overall framework	52		
			52		
		·	53		
		7.5.4 String	53		
		-	53		
			54		
	7.6	Batch test	55		
		7.6.1 @Linux	55		
			55		
	7.7	•	55		
8	WHAT	THE FUCK!!!	56		

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
27
            len = max(len, l);
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];
18
            res = max(res, now);
19
            if(now < 0) now = 0;
20
21
        return res;
22 | }
```

1.4 RMQ - st

```
|int _rmq[N][30], Log[你想开多大就开多大];
   for(int i = 2; i < 随便; i++) Log2[i] = Log2[i>>1] + 1;
2
3
   void init_RMQ(int *_orig) // [1, n]
4
5
       for(int i = 1; i <= n; i++) _rmq[i][0] = _orig[i];</pre>
6
       for(int j = 1; j <= Log2[n]; j++)</pre>
7
           for(int i = 1; i \le n + 1 - (1 \le j); i++)
                _{rmq[i][j]} = std::max(_{rmq[i][j-1]}, _{rmq[i+(1<<(j-1))][j]}
8
                   -1]);
9
10
   int query_RMQ(int l, int r) // max{x E [l, r]}
   {
11
12
       int k = Log2[r - l + 1];
```

```
13 | return std::max(_rmq[l][k], _rmq[r-(1<<k)+1][k]);
14 |}
```

1.5 数位 dp

```
1 #include <cstdio>
   #include <cstring>
   #include <algorithm>
   // calculate the number of numbers in [l, r] which not contain '4' or
      '62'
   long long l, r;
5
   int k;
6
7
   int L[100], R[100];
   long long f[100][2][2][10];
   int predo(long long a, int *num)
10
11
   {
       int len = 0;
12
13
       do num[++len] = a % 10; while(a /= 10);
       return len;
14
15
   }
16
17
   long long calc(int pos, bool d, bool u, int pre)
18
19
       if(pos == 0) return 1;
       long long &res = f[pos][d][u][pre];
20
       if (res !=-1) return res;
21
22
       res = 0;
       int st = d ? L[pos] : 0;
23
24
       int ed = u ? R[pos] : 9;
25
       for(int i = st; i <= ed; i++)</pre>
26
27
           if(i == 4 || (pre == 6 && i == 2)) continue;
            res += calc(pos-1, d && i == L[pos], u && i == R[pos], i);
28
29
       }
30
       return res;
31
   }
32
   int main()
33
34
   {
       while(scanf("%lld%lld", &l, &r) == 2 && (l || r))
35
36
       {
           memset(f, -1, sizeof(f));
37
38
           memset(L, 0, sizeof(L));
39
           memset(R, 0, sizeof(R));
40
           int len = std::max(predo(l, L), predo(r, R));
            printf("%lld\n", calc(len, 1, 1, 0));
41
42
       }
43
       return 0;
44 | }
```

```
1.6 状压 dp
```

1.6.1 枚举子集

```
1 | for(int st = S; st; st = (st -1) & S);
```

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

```
1 | ax + by = 1
   bx1 + (a\%b)y1 = 1
                        ==> bx + (a-a/b*b)y = 1
2
     ==> ay1 + b(x1-a/b*y1) = 1
   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;
14
       int e = egcd(b, a%b, x1, y1);
15
       x = y1;
       y = x1 - a / b * y1;
16
17
       return e;
18 | }
```

2.2 Prime

2.2.1 Make Prime List

```
7
            {
8
                 if(!h[i]) pri[Cnt++] = i;
9
                 for(int j = 0; j < Cnt && pri[j] <= maxp / i; j++)</pre>
                 {
10
                     h[i * pri[j]] = true;
11
                     if(i % pri[j] == 0) break;
12
                 }
13
14
            }
15
        }
16 | }
```

2.2.2 Prime Factor

```
void factor()
        1
        2
                                                                              make_prime_list();
        3
        4
                                                                              for(int j = 0; j < Cnt && pri[j]*pri[j] <= n; j++)</pre>
        5
                                                                                                                        if(n % pri[j] == 0)
        6
        7
                                                                                                                        {
                                                                                                                                                                     printf("%d<sub>\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\underline{\under</sub>
      8
        9
                                                                                                                                                                     while(n % pri[j] == 0) n /= pri[j];
10
11
                                                                            if(n!=1) printf("%d",n);
12
13 | }
```

2.3 Fast Power

```
1 //x^y \% mod
2
   int mul(int x, LL y, int mod) // 递归
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
   int mul(int x, int y, int mod) // 非递归
8
9
       int s = 1;
10
11
       int ss = x;
12
       while(y)
13
14
           if(y & 1) s = s * ss;
15
           y /= 2;
16
           ss *= ss;
17
18
       return s;
19 | }
```

2.4 约瑟夫环、丢手绢问题

```
1 #include <cstdio>
   //UVALive 4727
3
   int n, m;
4
   int Joseph(int totalPeople, int nextNumber, int startIndex, int lastIdx
5
   { // All based on O_Index , the Answer is the last `lastIdx` to leave
6
7
        int now = (nextNumber - 1) % lastIdx + (startIndex - nextNumber);
        for(int i = lastIdx + 1; i <= totalPeople; i++)</pre>
8
9
            now = (now + nextNumber) % i;
10
        return now;
11
   }
12
13
   int main()
14
   {
       int T; scanf("%d", &T);
15
16
        while (T--)
17
        {
18
            scanf("%d%d", &n, &m);
            printf("%d_{\square}%d_{\square}%d_{\square}", Joseph(n, m, m, 3)+1, Joseph(n, m, m, 2)
19
               +1, Joseph(n, m, m, 1)+1);
20
21
        return 0;
22 | }
```

2.5 康拓展开 Cantor

```
1 #include <cstdio>
   #include <cstring>
2
3
   int fac[10], a[10];
4
5
   bool Read(int *p)
6
7
        for(int i = 0; i < 9; i++)</pre>
8
9
10
             char chtmp;
            if(scanf("<sub>□</sub>%c", &chtmp) != 1) return 0;
11
             p[i] = chtmp == 'x' ? 0 : chtmp - '0';
12
13
        }
14
        return 1;
15
16
   int Cantor(int *p) // Eight puzzle status -> Integer
17
18
   {
19
        int res = 0;
20
        for(int i = 0; i < 9; i++)</pre>
21
22
             int cnt = 0;
23
             for(int j = i + 1; j < 9; j++)</pre>
```

```
24
                if(p[j] < p[i]) cnt++;
25
            res += cnt * fac[9 - i - 1];
26
27
       return res;
28
29
30
   bool used[10] = {0};
31
   int getRank(int r)
32
       for(int i = 0, j = 0; i < 9; i++)
33
34
35
            if(!used[i] && j == r) return i;
            if(!used[i]) j++;
36
37
       }
38
39
   void getStatus(int cantor, int *p) // Integer -> Eight puzzle status
40
       memset(used, 0, sizeof(used));
41
       for(int i = 0; i < 9; i++)</pre>
42
43
       {
44
            p[i] = getRank(cantor / fac[9 - i - 1]);
            used[p[i]] = 1;
45
            cantor \%= fac[9 - i - 1];
46
47
       }
48
49
50
   void PRINT(int *p)
51
   {
52
       int hash = Cantor(p);
53
       printf("Cantor uvalue = 1%d\n", hash);
54
       getStatus(hash, p);
55
       printf("Cantor Status = ");
56
       for(int i = 0; i < 9; i++) printf("%d", p[i]); puts("");</pre>
57
58
59
   int main()
60
       fac[0] = 1; for(int i = 1; i < 10; i++) fac[i] = fac[i-1] * i;
61
       while(Read(a)) PRINT(a);
62
63
       return 0;
64 | }
```

3 Datastructure

3.1 带权并查集

```
#include <cstdio>
#include <cstdlib>
const int N = 100000 + 10;
```

```
6
   int n, f[N], g[N];
7
   int getroot(int x)
8
9
10
       if(f[x] == x) return x;
       int tmp = getroot(f[x]);
11
       g[x] += g[f[x]]; // update the value
12
13
       return f[x] = tmp;
14
15
   void merge(int x, int y) // merge x's set and y's set
16
   \{ // Guarantee that the x must be the root of its set, which means x ==
17
       getroot(x) is true, but it may not be same for y
18
       int fy = getroot(y);
19
       g[x] += g[y] + abs(x - y) % 1000; // update the value
20
       f[x] = fy;
21
22
   int main()
23
24
   {
25
       scanf("%d", &n);
26
       for(int i = 1; i <= n; i++) f[i] = i;</pre>
27
       char op; int x, y;
       while(scanf("<sub>□</sub>%c", &op) == 1 && op != '0')
28
29
            if(op == 'I')
30
31
32
                scanf("%d%d", &x, &y);
33
                if(getroot(x) == getroot(y)) continue;
34
                merge(x, y);
35
            }
            else{
36
37
                scanf("%d", &x);
                getroot(x); // !!! update the value of x before output
38
39
                printf("%d\n", g[x]);
40
            //for(int i = 1; i <= n; i++) printf("%d ", f[i]); puts("");
41
42
43
       return 0;
44 | }
```

3.2 手写 Heap

```
1 #include <cstdio>
2 #include <algorithm>
3
4 const int N = 250000;
5
6 int n, a[N], x, size = 0;
7
8 void update(int i)
9 {
```

```
10
        while(i > 1 && a[i] > a[i/2])
11
            std::swap(a[i], a[i/2]);
12
            i /= 2;
13
14
        }
15
16
17
   void pop()
18
        int i = 1; a[i] = 0;
19
        while(i * 2 <= size && (a[i] < a[i*2] || a[i] < a[i*2+1]))</pre>
20
21
        {
            if(i * 2 == size || (i * 2 < size && a[i*2] >= a[i*2+1]))
22
23
             {
24
                 a[i] = a[i*2];
25
                 a[i*2] = 0;
                 i = i * 2;
26
27
             }
             else {
28
                 a[i] = a[i*2+1];
29
30
                 a[i*2+1] = 0;
31
                 i = i * 2 + 1;
             }
32
33
        }
        a[i] = a[size]; size--;
34
35
        update(i);
36
37
38
   int main()
39
        scanf("%d", &n);
40
        for(int i = 1; i <= n; i++)</pre>
41
42
             scanf("%d", &x);
43
44
            a[++size] = x;
45
            update(size);
46
47
        for(int i = 1; i <= n; i++)</pre>
48
             printf("%d<sub>□</sub>", a[1]);
49
50
             pop();
51
        }
52
        return 0;
53 | }
```

3.3 Leftist Tree

```
6
  | {
      if(!x \mid | !y) return x|y;
7
      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
      11
12
      D[x] = D[R[x]] + 1;
      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 | }
```

3.4 Partition Tree

```
| struct Parti{int val, left;} val[30][N];
   void build_tree(int d, int l, int r)
2
3
   {
       if(l == r) return;
4
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>
8
            if(val[d][i].val < sorted[m]) same——;
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] &&</pre>
12
                same))
            {
13
                flag = 1;
14
                val[d + 1][lcnt++] = val[d][i];
15
                if(val[d][i].val == sorted[m]) same--;
16
17
            else val[d][rcnt++] = val[d][i];
18
19
            val[d][i].left = val[d][i - 1].left + flag;
20
21
       build_tree(d + 1, l, m);
       build_tree(d + 1, m + 1, r);
22
23
24
   int query(int d, int l, int r, int x, int y, int k)
25
   {
       if(l == r) return val[d][l].val;
26
       int m = (l + r) >> 1;
27
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
```

```
int rx = (x - 1 - l + 1) - lx; //[l,x-1] to right
int ry = (y - x + 1) - ly; //[x,y] to right
if(ly >= k) return query(d+1, l, m, l-1+lx+1, l-1+lx+ly, k);
else return query(d+1, m+1, r, m+1-1+rx+1, m+1-1+rx+ry, k-ly);
}
```

3.5 Treap

3.5.1 @ Array

```
struct treap {
1
       const int N = 100000 + 10;
2
       int L[N*20], R[N*20], S[N*20], fix[N*20], A[N*20];
3
4
       int root, total;
5
       void rotate_left(int &p)
6
       {
7
            int tmp = R[p];
8
            R[p] = L[tmp];
9
            int zsize = S[L[tmp]];
            S[p] = S[p] - S[tmp] + zsize;
10
11
            L[tmp] = p;
12
            S[tmp] = S[tmp] - zsize + S[p];
13
            p = tmp;
14
       }
15
       void rotate_right(int &p)
16
17
            int tmp = L[p];
18
            L[p] = R[tmp];
19
            int zsize = S[R[tmp]];
20
            S[p] = S[p] - S[tmp] + zsize;
21
            R[tmp] = p;
            S[tmp] = S[tmp] - zsize + S[p];
22
23
            p = tmp;
24
25
       void Insert(int &p, int x)
26
       {
            if(!p)
27
28
            {
29
                p = ++total;
30
                L[p] = R[p] = 0;
                S[p] = 1;
31
32
                fix[p] = rand();
33
                A[p] = x;
34
                return;
35
            }
            S[p]++;
36
            if(x < A[p])
37
38
39
                Insert(L[p], x);
40
                if(fix[L[p]] > fix[p]) rotate_right(p);
41
            else {
42
```

```
43
                Insert(R[p], x);
44
                if(fix[R[p]] > fix[p]) rotate_left(p);
45
            }
       }
46
47
       int Delete_min(int &p)
48
49
            S[p]--;
50
            if(!L[p])
51
            {
52
                int value = A[p];
53
                p = R[p];
54
                return value;
55
            else return Delete_min(L[p]);
56
57
58
       void Delete(int &p, int x)
59
       {
            if(!p) return;
60
61
            S[p]--;
            if(x < A[p]) Delete(L[p], x);
62
63
            else if(x > A[p]) Delete(R[p], x);
64
            else {
65
                if(!L[p] && !R[p]) p = 0;
                else if(!L[p] || !R[p])
66
67
                     if(!L[p]) p = R[p];
68
                     else p = L[p];
69
70
                }
71
                else A[p] = Delete_min(R[p]);
72
            }
73
74
       int Count_leq(int &p, int x)
75
76
            if(!p) return 0;
77
            if(A[p] <= x) return S[L[p]] + 1 + Count_leq(R[p], x);</pre>
78
            else return Count_leq(L[p], x);
79
80
       int Count_geq(int &p, int x)
81
82
            if(!p) return 0;
83
            if(A[p] >= x) return S[R[p]] + 1 + Count_geq(L[p], x);
84
            else return Count_geq(R[p], x);
85
       }
86
       int Find_kth(int &p, int k)
87
       {
            if(k == S[L[p]] + 1) return A[p];
88
89
            if(k <= S[L[p]]) return Find_kth(L[p], k);</pre>
            else return Find_kth(R[p], k - S[L[p]] - 1);
90
91
       }
92 | };
```

3.5.2 @ Pointer

```
1
   struct treap {
2
       struct Treap{
3
            int fix, key, size;
4
            Treap *left, *right;
5
            Treap() { fix = key = size = left = right = 0; }
       }*root, *null;
6
7
       void init()
8
9
10
            null = new Treap;
            root = null;
11
12
       }
       void rotate left(Treap *&p)
13
14
15
            Treap *tmp = p -> right;
            p -> right = tmp -> left;
16
17
            int zsize = tmp -> left -> size;
            p -> size = p -> size - tmp -> size + zsize;
18
            tmp -> left = p;
19
20
            tmp -> size = tmp -> size - zsize + p -> size;
21
            p = tmp;
22
23
       void rotate_right(Treap *&p)
24
       {
25
            Treap *tmp = p -> left;
26
            p -> left = tmp -> right;
27
            int zsize = tmp -> right -> size;
28
            p -> size = p -> size - tmp -> size + zsize;
29
            tmp -> right = p;
            tmp -> size = tmp -> size - zsize + p -> size;
30
31
            p = tmp;
       }
32
33
34
       void Insert(Treap *&p, int x)
35
       {
            if(p == null)
36
37
                p = new Treap;
38
39
                p \rightarrow fix = rand();
40
                p \rightarrow key = x;
41
                p \rightarrow size = 1;
42
                p -> left = null;
43
                p -> right = null;
44
                return;
45
            if(x 
46
47
            {
48
                Insert(p -> left, x);
49
                p -> size++;
                if(p -> left -> fix > p -> fix) rotate right(p);
50
51
            }
```

```
52
                                                 else {
  53
                                                                Insert(p -> right, x);
  54
                                                                p -> size++;
                                                                if(p -> right -> fix > p -> fix) rotate_left(p);
  55
                                                }
   56
   57
   58
                                int Delete_min(Treap *&p)
   59
  60
                                                p -> size--;
                                                if(p -> left == null)
  61
  62
  63
                                                                int value = p -> key;
  64
                                                                p = p -> right;
  65
                                                                return value;
  66
                                                else return Delete_min(p -> left);
  67
  68
                                void Delete(Treap *&p, int x) // Make sure that `x` is existed
  69
  70
  71
                                                if(p == null) return;
  72
                                                 p -> size--;
  73
                                                if(x  key) Delete(p -> left, x);
  74
                                                 else if(x > p -> key) Delete(p -> right, x);
  75
                                                 else { // delete *p
                                                                if(p -> left == null && p -> right == null)
  76
  77
                                                                {
  78
                                                                                p = null;
  79
  80
                                                                else if(p -> left == null || p -> right == null)
  81
                                                                                 if(p -> left == null)
  82
  83
  84
                                                                                                 p = p -> right;
  85
                                                                                 else { // p -> right == null
  86
                                                                                                 p = p -> left;
  87
                                                                                 }
  88
  89
                                                                else \{ // p \rightarrow left != null \&\& p \rightarrow right != null \&
  90
                                                                                 p -> key = Delete_min(p -> right);
  91
  92
                                                                }
                                                }
  93
  94
  95
                                int Count_leq(Treap *&p, int x)
                                {
  96
  97
                                                if(p == null) return 0;
                                                98
                                                                right, x);
  99
                                                else return Count_leq(p -> left, x);
100
                                int Count_geq(Treap *&p, int x)
101
                                {
102
                                                if(p == null) return 0;
103
```

ACM-ICPC Template GuessEver

```
104
          -> left, x);
105
          else return Count_geq(p -> right, x);
106
107
       int Find_kth(Treap *&p, int x)
108
109
          if(k == p -> left -> size + 1) return p -> key;
110
          if(k <= p -> left -> size) return Find_kth(p -> left, k);
          else return Find_kth(p -> right, k - p -> left -> size - 1);
111
112
      }
113
   };
```

3.6 Size Balanced Tree

```
struct SBT {
1
2
       const int N = 100000 + 10;
3
       int A[N*20], S[N*20], L[N*20], R[N*20];
4
       int root, total;
5
       void rotate_left(int &x)
6
7
            int y = R[x];
            R[x] = L[y];
8
9
            L[y] = x;
10
            S[y] = S[x];
11
            S[x] = S[L[x]] + S[R[x]] + 1;
12
            x = y;
       }
13
       void rotate_right(int &x)
14
15
            int y = L[x];
16
17
            L[x] = R[y];
            R[y] = x;
18
19
            S[y] = S[x];
20
            S[x] = S[L[x]] + S[R[x]] + 1;
21
            x = y;
22
       void maintain(int &p, bool flag)
23
24
25
            if(flag)
            {
26
                if(S[R[R[p]]] > S[L[p]]) rotate_left(p);
27
                else if(S[R[L[p]]] > S[L[p]])
28
29
                {
30
                     rotate_right(R[p]);
31
                     rotate_left(p);
32
33
                else return;
            }
34
            else
35
36
            {
37
                if(S[L[L[p]]] > S[R[p]]) rotate_right(p);
38
                else if(S[L[R[p]]] > S[R[p]])
```

```
39
                {
                     rotate_left(L[p]);
40
41
                     rotate_right(p);
                }
42
                else return;
43
44
45
            maintain(L[p], 0);
46
            maintain(R[p], 1);
47
            maintain(p, 0);
48
            maintain(p, 1);
        }
49
        void Insert(int &p, int x)
50
51
            if(!p)
52
53
            {
54
                p = ++total;
55
                L[p] = R[p] = 0;
                A[p] = x; S[p] = 1;
56
57
                return;
            }
58
59
            S[p]++;
60
            if(x < A[p]) Insert(L[p], x);
            else Insert(R[p], x);
61
            maintain(p, x >= A[p]);
62
63
        }
64
       int Delete_min(int &p)
65
66
            S[p]--;
67
            if(!L[p])
68
69
                int value = A[p];
70
                p = R[p];
71
                return value;
72
73
            else return Delete_min(L[p]);
74
        }
75
        void Delete(int &p, int x)
        {
76
            if(!p) return;
77
            S[p]--;
78
            if(x < A[p]) Delete(L[p], x);
79
80
            else if(x > A[p]) Delete(R[p], x);
            else {
81
82
                if(!L[p] && !R[p]) p = 0;
                else if(!L[p] || !R[p])
83
84
85
                     if(!L[p]) p = R[p];
86
                     else p = L[p];
87
                }
88
                else A[p] = Delete_min(R[p]);
            }
89
90
        int Count_leq(int &p, int x)
91
```

```
ACM-ICPC Template
92
        {
93
             if(!p) return 0;
             if(A[p] <= x) return S[L[p]] + 1 + Count_leq(R[p], x);</pre>
94
             else return Count_leq(L[p], x);
95
96
        }
97
        int Count geg(int &p, int x)
98
        {
99
             if(!p) return 0;
             if(A[p] >= x) return S[R[p]] + 1 + Count_geq(L[p], x);
100
             else return Count_geq(R[p], x);
101
        }
102
        int Find kth(int &p, int k)
103
104
105
             if(k == S[L[p]] + 1) return A[p];
106
             if(k <= S[L[p]]) return Find_kth(L[p], k);</pre>
             else return Find_kth(R[p], k - S[L[p]] - 1);
107
108
        }
109
   |};
```

树链剖分 Heavy-Light Decomposition

```
1 // Solution: www.guessbug.com/problem/HDU/3966
  #pragma comment(linker, "/STACK:1024000000,1024000000")
2
   #include <cstdio>
3
  |#include <cstring>
   #include <vector>
5
   using std::vector;
6
7
   // HDU 3966 : Increase or decrease a value on path [x-y] on a tree.
8
                  Query a value of a certain point
9
   const int N = 50000 + 10;
10
11
   int n, m, q, a[N];
12
   vector < int > path[N];
13
14
   // Heavy—Light Decomposition
   int size[N], father[N], deep[N], heavy_son[N];
15
   int top[N], segid[N], time_stamp;
16
17
   void dfs1(int x, int fa, int deepth)
   {
18
19
       size[x] = 1; father[x] = fa; deep[x] = deepth;
       for(vector<int>::iterator it = path[x].begin(); it != path[x].end()
20
          ; it++)
21
       {
22
           if(*it == father[x]) continue;
23
           dfs1(*it, x, deepth + 1);
24
           size[x] += size[*it];
           if(size[*it] > size[heavy_son[x]]) heavy_son[x] = *it;
25
       }
26
27
28
   void dfs2(int x, int topx)
29
   {
30
       top[x] = topx;
```

```
31
       segid[x] = ++time_stamp;
       if(heavy_son[x]) dfs2(heavy_son[x], topx); // not leaf
32
33
       for(vector < int >::iterator it = path[x].begin(); it != path[x].end()
          ; it++)
           if(*it != father[x] && *it != heavy_son[x])
34
                dfs2(*it, *it);
35
36
   // Heavy—Light Decomposition ——— END
37
38
39
   int add[N*4];
   void pushDown(int p)
40
41
       add[p*2] += add[p];
42
43
       add[p*2+1] += add[p];
44
       add[p] = 0;
45
   void modify(int p, int l, int r, int a, int b, int c)
46
47
48
       if(a <= l && b >= r)
49
       {
50
            add[p] += c;
51
            return;
52
       int mid = l + (r - l) / 2;
53
54
       pushDown(p);
55
       if(a <= mid) modify(p*2, l, mid, a, b, c);
       if(b > mid) modify(p*2+1, mid+1, r, a, b, c);
56
57
58
   int query(int p, int l, int r, int a)
59
       if(l == r && l == a) return add[p];
60
       int mid = l + (r - l) / 2;
61
       pushDown(p);
62
       if(a <= mid) return query(p*2, l, mid, a);</pre>
63
       else return query(p*2+1, mid+1, r, a);
64
65
66
   void change(int a, int b, int c)
67
68
69
       while(top[a] != top[b])
70
71
           if(deep[top[a]] < deep[top[b]]) std::swap(a, b);</pre>
            modify(1, 1, n, segid[top[a]], segid[a], c);
72
73
            a = father[top[a]];
74
75
       if(deep[a] > deep[b]) std::swap(a, b);
76
       modify(1, 1, n, segid[a], segid[b], c);
77
78
79
   int main()
80
       while(scanf("%d%d%d", &n, &m, &q) == 3)
81
82
```

```
83
             time_stamp = 0;
             for(int i = 1; i <= n; i++)</pre>
84
85
             {
                  size[i] = father[i] = heavy_son[i] = 0;
86
87
                  deep[i] = top[i] = segid[i] = 0;
                  path[i].clear();
88
89
90
             for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
             for(int i = 1; i <= m; i++)</pre>
91
92
             {
93
                  int x, y; scanf("%d%d", &x, &y);
94
                  path[x].push back(y);
95
                  path[y].push_back(x);
             }
96
97
             dfs1(1, 0, 1);
             dfs2(1, 1);
98
             memset(add, 0, sizeof(add));
99
             for(int i = 1; i <= n; i++) change(i, i, a[i]);</pre>
100
             while (q--)
101
             {
102
103
                  char op; scanf("\u00e4%c", &op);
104
                  if(op == 'I' || op == 'D')
105
                  {
                      int a, b, c; scanf("%d%d%d", &a, &b, &c);
106
                      if(op == 'I') change(a, b, c);
107
                      else change(a, b, -c);
108
109
                  }
110
                  else {
111
                      int x; scanf("%d", &x);
112
                      printf("%d\n", query(1, 1, n, segid[x]));
                  }
113
             }
114
115
         }
116
         return 0;
117 | }
```

3.8 三维偏序 - CDQ 分治

```
1 #include <cstdio>
   |#include <cstring>
2
   #include <algorithm>
   #define lowbit(_X) ((_X)&(_(_X)))
   // SPOJ LIS2
5
6
   const int N = 100000 + 10;
7
   int n, f[N], idx[N], hash[N];
8
   struct Node{
9
10
       int x, y, z;
       void Read(int i)
11
12
       {
13
           scanf("%d%d", &y, &z);
14
           x = i; f[i] = 1; idx[i] = i;
```

```
15
       }
   }a[N];
16
17
   int maxp;
   int c[N]; // tree Array
18
19
   bool cmpx(int i, int j) { return a[i].x < a[j].x; }</pre>
20
21
   bool cmpy(int i, int j) { return a[i].y < a[j].y; }</pre>
22
   bool cmpz(int i, int j) { return a[i].z < a[j].z; }</pre>
23
24
   void discrete()
25
   {
26
       std::sort(idx+1, idx+1+n, cmpy); maxp = 0;
       for(int i = 1; i <= n; i++)</pre>
27
28
       {
29
            if(i == 1 \mid | a[idx[i]].y != a[idx[i-1]].y) hash[idx[i]] = ++
            else hash[idx[i]] = maxp;
30
       }
31
32
       for(int i = 1; i <= n; i++) a[idx[i]].y = hash[idx[i]];
       std::sort(idx+1, idx+1+n, cmpz); maxp = 0;
33
34
       for(int i = 1; i <= n; i++)</pre>
35
            if(i == 1 \mid | a[idx[i]].z != a[idx[i-1]].z) hash[idx[i]] = ++
36
               maxp;
37
            else hash[idx[i]] = maxp;
38
       for(int i = 1; i <= n; i++) a[idx[i]].z = hash[idx[i]];</pre>
39
40
41
42
   void insert(int a, int x)
43
   {
       for( ; a <= maxp; a += lowbit(a)) c[a] = std::max(c[a], x);
44
45
   int query(int a) // [1, a]
46
47
48
       int res = 0;
49
       for( ; a > 0; a = lowbit(a)) res = std::max(res, c[a]);
50
       return res;
51
52
53
   void solve(int l, int mid, int r)
54
   {
       std::sort(&idx[l], &idx[mid]+1, cmpy);
55
56
       std::sort(&idx[mid+1], &idx[r]+1, cmpy);
       // [l, mid] .. calculated ok
57
       // now calculating [mid+1, r]
58
59
       // f[i] = max\{f[j]\} + 1;
       int j = l;
60
61
       for(int i = mid + 1; i <= r; i++)</pre>
62
63
            for( ; j <= mid && a[idx[j]].y < a[idx[i]].y; j++)</pre>
64
                insert(a[idx[j]].z, f[a[idx[j]].x]);
            int tmp = query(a[idx[i]].z - 1);
65
```

ACM-ICPC Template GuessEver

```
66
            if(tmp + 1 > f[a[idx[i]].x]) f[a[idx[i]].x] = tmp + 1;
67
        }
68
        //memset(c, 0, sizeof(c));
        for(int i = l; i <= mid; i++)</pre>
69
70
71
            int b = a[idx[i]].z;
72
            for( ; b <= maxp; b += lowbit(b)) c[b] = 0;</pre>
73
        }
        std::sort(&idx[mid+1], &idx[r]+1, cmpx);
74
75
        // CDQ(mid+1, r) next, so sort back it
76
77
   void CDQ(int l, int r)
78
79
   {
80
       if(l == r) return;
        int mid = l + (r - l) / 2;
81
        CDQ(l, mid);
82
        solve(l, mid, r);
83
84
        CDQ(mid + 1, r);
85
86
87
   int main()
88
   {
89
        scanf("%d", &n);
90
        for(int i = 1; i <= n; i++) a[i].Read(i);</pre>
        discrete();
91
        std::sort(idx+1, idx+1+n, cmpx);
92
93
        CDQ(1, n);
94
        int res = 1;
95
        //for(int i = 1; i <= n; i++)    printf("%d ", f[i]);    puts("");
96
        for(int i = 1; i <= n; i++) if(f[i] > res) res = f[i];
97
        printf("%d\n", res);
98
        return 0;
99
   }
```

4 Graph

4.1 Shortest path

4.1.1 Dijkstra

```
void dijkstra()
1
2
   {
3
       memset(dist, 0x3f, sizeof(dist));
4
       dist[1] = 0; Q.push(make_pair(0, 1));
5
       while(!Q.empty())
       {
6
7
           int x = Q.top().second; Q.pop();
           if(done[x]) continue;
8
9
           done[x] = 1;
10
           for(Link p = head[x]; p; p = p->next)
                if(dist[p->y] > dist[x] + p->z)
11
```

19

```
ACM-ICPC Template
                                                                          GuessEver
12
                {
                     dist[p->y] = dist[x] + p->z;
13
14
                     Q.push(make_pair(-dist[p->y], p->y));
                }
15
16
        }
   }
17
   4.1.2 Spfa
   void spfa()
2
3
        memset(inQ, 0, sizeof(inQ));
        memset(dist, 0x3f, sizeof(dist));
4
5
        dist[S] = 0; Q.push(S); inQ[S] = 1; //S为源点
6
        while(!Q.empty())
7
        {
8
            int x = Q.front(); 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
                {
                     dist[p->y] = dist[x] + p->z;
12
                     if(!inQ[p->y])
13
14
15
                         Q.push(p->y);
16
                         inQ[p->y] = 1;
17
                     }
                }
18
        }
19
20 | }
   4.1.3 Floyd
1
   void floyd()
2
   {
        for(int k = 1; k \leftarrow n; k++) // 这里可以看作是一个加边的过程
3
4
            for(int i = 1; i <= n; i++)</pre>
                for(int j = 1; j <= n; j++)</pre>
5
                     map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
6
7
   }
8
   // 最小环
9
   void MinCircle()
10
11
   {
        cap[] = map[];
12
        int circle = 0x3f3f3f3f;
13
        for(int k = 1; k <= n; k++)</pre>
14
15
16
            for(int i = 1; i < k; i++)</pre>
17
                for(int j = i+1; j < k; j++)</pre>
                     circle = min(circle, map[i][j] + cap[j][k]+cap[k][i]);
18
```

for(int i = 1; i <= n; i++)</pre>

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

4.2 Minimum Spanning Tree

4.2.1 Prim

```
int prim()
1
2
       memset(dist, 0x3f, sizeof(dist));
3
       dist[1] = 0; Q.push(make_pair(0, 1));
4
5
       int res = 0;
6
       while(!Q.empty())
7
       {
8
            int x = Q.top().second; Q.pop();
9
            if(done[x]) continue;
            res += dist[x]; done[x] = 1;
10
            for(Link p = head[x]; p; p = p->next)
11
12
                if(dist[p->y] > p->z)
13
                {
14
                    dist[p->y] = p->z;
15
                    Q.push(make_pair(-dist[p->y], p->y));
                }
16
17
       }
18
       return res;
19 | }
```

4.2.2 Kruskal

4.3 Tarjan - Strong Union

```
void dfs(int x)
1
2
3
       now[x] = low[x] = ++dfstime;
4
       hash[x] = 1;
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
14
                else if(inst[i]) low[x] = min(low[x], now[i]);
15
       if(low[x] == now[x])
16
17
       {
18
            while(!st.empty())
19
            {
20
                int u = st.top();
21
                st.pop(); inst[u] = 0;
22
                belong[u] = number;
                if(u == x) break;
23
24
25
            numer++;
26
       }
27
   void tarjan()
28
29
30
       for(int i = 1; i <= n; i++)</pre>
            if(!hash[i]) dfs(i);
31
                                               栈中还会剩下一个强连通分量
       if(!st.empty()) // 这是一个未知 bug
32
33
       {
34
            while!st.empty())
35
            {
36
                int u = st.top();
37
                st.pop();
                belong[u] = number;
38
39
40
            number++;
41
       }
42 | }
```

4.4 LCA

4.4.1 @ Tarjan

```
1 // poj 1330 (changed something)
   // LCA tarjan
2
3 #include <cstdio>
   #include <cstring>
4
5
   const int N = 10000 + 10;
6
7
8
   int n;
9
   struct Link{int y, idx; Link *next;}*head[N], *ask[N];
   int tx, ty;
10
   bool in[N], vis[N];
11
   int f[N];
12
   int ans[N]; // Query Answer
13
14
   void inLink(int x, int y)
15
16
17
        Link *p = new Link;
18
        p \rightarrow y = y;
        p -> next = head[x];
19
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
        p \rightarrow next = ask[x];
27
28
        ask[x] = p;
29
30
   int getroot(int x)
31
32
   {
        return f[x] == x ? x : f[x] = getroot(f[x]);
33
34
   }
35
   void LCA(int x)
36
37
38
        vis[x] = 1;
39
        f[x] = x;
        for(Link *p = ask[x]; p; p = p -> next)
40
41
            if(vis[p->y]) ans[p->idx] = getroot(p->y);
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
```

```
50
  int main()
51
   {
        int T; scanf("%d", &T);
52
53
        while (T--)
54
        {
            memset(head, 0, sizeof(head));
55
            memset(ask, 0, sizeof(ask));
56
57
            memset(in, 0, sizeof(in));
            memset(vis, 0, sizeof(vis));
58
            scanf("%d", &n);
59
            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;
64
                 scanf("%d%d", &x, &y);
65
                 inLink(x, y);
                 in[y] = 1;
66
            }
67
68
            int q = 1;// the number of query
            for(int i = 1; i <= q; i++)</pre>
69
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>
                 if(!in[i]) {root = i; break;}
76
77
            LCA(root);
78
            for(int i = 1; i <= q; i++)</pre>
79
                 printf("%d\n", ans[i]);
80
        return 0;
81
82 | }
```

4.4.2 @ Doubling Algorithm

```
1 #include <cstdio>
   #include <cstring>
2
   #include <algorithm>
   // POJ 1330
                 LCA_Doubling Algorithm
   const int N = 10000 + 10;
5
6
7
   const int UPDeepth = 14;
   int n;
8
9
   struct Link{
10
       int y;
       Link *next;
11
   }*head[N];
12
   bool in[N];
13
   int ancient[N][UPDeepth+1];
14
   int deep[N];
15
16
```

```
17
   void inLink(int x, int y)
18
19
        Link *p = new Link;
20
        p \rightarrow y = y;
21
        p -> next = head[x];
22
        head[x] = p;
23
24
   void dfs(int x, int deepth, int father)
25
26
27
        deep[x] = deepth;
28
        ancient[x][0] = father;
        for(Link *p = head[x]; p; p = p -> next)
29
30
            dfs(p \rightarrow y, deepth + 1, x);
31
   }
32
33
   void getLCA()
34
35
        for(int i = 1; i <= n; i++)</pre>
            if(!in[i]) dfs(i, 1, 0);
36
37
        for(int j = 1; j <= UPDeepth; j++)</pre>
38
            for(int i = 1; i <= n; i++)</pre>
                 ancient[i][j] = ancient[ancient[i][j-1]][j-1];
39
40
41
42
   int LCA(int x, int y)
43
44
        if(deep[x] > deep[y]) std::swap(x, y); // deep[x] \leftarrow deep[y]
45
        for(int j = UPDeepth; j >= 0; j--)
46
            if(deep[x] <= deep[ancient[y][j]]) y = ancient[y][j];</pre>
47
        if(x == y) return x;
        for(int j = UPDeepth; j >= 0; j--)
48
49
            if(ancient[x][j] != ancient[y][j])
            {
50
                 x = ancient[x][j];
51
52
                 y = ancient[y][j];
53
        return ancient[y][0];
54
55
56
57
   int main()
58
   {
59
        int T; scanf("%d", &T);
60
        while (T--)
        {
61
            memset(head, 0, sizeof(head));
62
            memset(ancient, 0, sizeof(ancient));
63
64
            memset(in, 0, sizeof(in));
65
            memset(deep, 0, sizeof(deep));
            scanf("%d", &n);
66
67
            for(int i = 1; i < n; i++)</pre>
            {
68
                 int x, y; scanf("%d%d", &x, &y);
69
```

4.5 Bipartite Graph

4.5.1 Maximal Matching - The Hungarian algorithm

```
1 | int timeStamp = 0;
   int n, m, g[N][N];
2
   int vis[N], pre[N];
3
5
   bool search(int x)
6
7
        for(int i = 1; i <= m; i++)</pre>
            if(g[x][i] && vis[i] != timeStamp)
8
            {
9
                 vis[i] = timeStamp;
10
11
                 if(pre[i] == -1 \mid | search(pre[i]))
                 {
12
13
                      pre[i] = x;
                      return 1;
14
15
                 }
16
17
        return 0;
18
19
20
   int maxMatch()
21
22
        int res = 0;
        memset(pre, -1, sizeof(pre));
23
        for(int i = 1; i <= n; i++)</pre>
24
25
        {
26
            ++timeStamp;
27
            res += search(i);
28
29
        return res;
30
   }
```

4.5.2 Optimal Matching - KM

不会... 用费用流解决

4.6 Network Flow

4.6.1 Maximum Flow - isap

```
1 |#include <cstdio>
   #include <algorithm>
2
3
   const int N = 200 + 10;
4
5
   int n, m, g[N][N];
6
7
   int v[N], h[N];
8
   int S, T;
9
   int sap(int x, int flow)
10
11
       if(x == T) return flow;
12
13
       int res = 0;
14
       for(int i = S; i <= T; i++)</pre>
15
            if(g[x][i] && h[x] == h[i] + 1)
16
                int t = sap(i, std::min(g[x][i], flow - res));
17
                res += t; g[x][i] -= t; g[i][x] += t;
18
                if(res == flow) return res;
19
                if(h[S] >= T) return res;
20
21
            }
       //if(h[S] >= T) return res;
22
       if((--v[h[x]]) == 0) h[S] = T;
23
24
       ++v[++h[x]];
25
       return res;
26
   }
27
28
   int main()
29
30
       scanf("%d%d", &m, &n); // m = number of edges, n = number of points
       for(int i = 1; i <= m; i++)</pre>
31
32
33
            int x, y, z;
            scanf("%d%d%d", &x, &y, &z);
34
35
            g[x][y] += z;
36
       }
37
       v[0] = T; S = 1; T = n; // all idx started from `1`
38
       int maxflow = 0;
       while(h[S] < T) maxflow += sap(1, 0x3f3f3f3f);</pre>
39
       printf("%d\n", maxflow);
40
41
       return 0;
42 | }
```

4.6.2 Minimum Cost Maximum Flow - spfa

```
struct EG{int from,to,flow,cost,next;}edge[M];

void add_edge(int a,int b,int c,int d)

{
    edge[L]=(EG){a,b,c,+d,head[a]};
    head[a]=L++;
    edge[L]=(EG){b,a,0,-d,head[b]};
```

```
8
       head[b]=L++;
9
   }
10
   bool spfa()
11
12
       memset(inQ, 0, sizeof(inQ));
13
       memset(dist, 0x3f, sizeof(dist));
14
15
       dist[S] = 0;
16
       q.push(S);
17
       while(!q.empty())
18
19
            int x = q.front();
20
            q.pop();
            inQ[x] = 0;
21
22
            for(int i = head[x]; i != -1; i = edge[i].next)
                if(edge[i].flow && dist[edge[i].to] > dist[x] + edge[i].
23
                   cost)
                {
24
25
                    pre[edge[i].to] = i;
                    dist[edge[i].to] = dist[x] + edge[i].cost;
26
27
                    if(!inQ[edge[i].to])
28
                    {
                         inQ[edge[i].to] = 1;
29
30
                         q.push(edge[i].to);
                    }
31
                }
32
33
34
       return dist[T] != inf;
35
36
   void MFMC()
37
   {
       memset(head, -1, sizeof(head));
38
        建图调用 add_edge();
39
40
       int mincost = 0, maxflow = 0;
41
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
                edge[pre[i] ^ 1].flow += res;
52
53
            maxflow += res;
54
55
            mincost += res * dist[T];
56
       }
57
   }
```

5 Geometry

5.1 BASICS

```
1 #include <cstdio>
   #include <cmath>
2
   #include <algorithm>
   #include <vector>
5
   using std::vector;
6
7
   const double EPS = 1e-10, INF = 1e20;
8
   const double PI = acos(-1.0);
9
   const int sign(const double &x) {
       if(fabs(x) < EPS) return 0;</pre>
10
       return x < 0 ? -1 : 1;
11
12
   const int dcmp(const double &x, const double &y) {
13
14
       return sign(x - y);
15
   const double toDegree(const double &alpha) {
16
       return alpha * 180.0 / PI;
17
18
19
   const double toRad(const double &alpha) {
20
       return alpha * PI / 180.0;
21
   }
22
23
   struct Point {
       double x, y;
24
25
       Point() {}
       Point(const double \&_x, const double \&_y) { x = _x; y = _y; }
26
       void Read() { scanf("%lf%lf", &x, &y); }
27
28
       bool operator < (const Point &b) const {</pre>
29
           if(dcmp(x, b.x) == 0) return dcmp(y, b.y) < 0;
            return dcmp(x, b.x) < 0;
30
31
       }
32
       Point operator + (const Point &b) const {
33
            return Point(x + b.x, y + b.y);
34
35
       Point operator — (const Point &b) const {
36
            return Point(x - b.x, y - b.y);
37
       Point operator * (const double &k) const {
38
39
            return Point(x * k, y * k);
40
       Point operator / (const double &k) const {
41
42
            return Point(x / k, y / k);
43
       double operator * (const Point &b) const {
44
            return x * b.x + y * b.y;
45
46
47
       double operator ^ (const Point &b) const {
48
            return x * b.y - y * b.x;
49
       }
```

```
50
        bool operator == (const Point &b) const {
51
            return dcmp(x, b.x) == 0 && dcmp(y, b.y) == 0;
52
        double Abs() const {
53
54
            return sqrt(x * x + y * y);
55
56
        Point Rotate(const Point &o, const double &alpha) const {
57
            Point z = *this - o;
58
            double nx = z.x * cos(alpha) - z.y * sin(alpha);
59
            double ny = z.x * sin(alpha) + z.y * cos(alpha);
60
            return Point(nx, ny) + o;
61
        }
        double Angle() const {
62
63
            return atan2(y, x);
64
        }
65
    };
   typedef Point Vector;
66
    // 单位法向量
67
   const Vector getNormalVector(const Vector &P) {
68
        double L = P.Abs(); // `L` CANNOT BE `0` !!!!
69
70
        return Vector(-P.y / L, P.x / L);
71
   // 两向量夹角
72
   const double getAngle(const Vector &a, const Vector &b) {
73
74
        return acos(a * b / a.Abs() / b.Abs());
75
   }
76
77
   struct Line {
78
        Point s, e;
79
        Line() {}
80
        Line(const Point &_s, const Point &_e) { s = _s; e = _e; }
81
   };
82
   typedef Line Segment;
   // 判断两直线的关系
83
   const int getRelationBetweenLines(const Line &L1, const Line &L2) {
84
85
        if(sign((L1.e - L1.s) ^ (L2.e - L2.s)) == 0) {
            if(sign((L2.s - L1.s) ^ (L2.e - L1.s)) == 0) return 0; //
86
               coincidence
87
            else return 1; // parallel
88
        }
89
        return 2; // intersection
90
   // 直线交点
91
92
   const Point getLineIntersection(const Line &L1, const Line &L2) {
93
        Vector L1v = L1.e - L1.s, L2v = L2.e - L2.s;
94
        Vector u = L1.s - L2.s;
95
        double t = (L2v ^ u) / (L1v ^ L2v);
96
        return L1.s + L1v * t;
97
   // 点到直线的距离
98
   const double getDistanceFromPointToLine(const Point &P, const Line &L)
99
      {
100
        Vector v1 = L.e - L.s, v2 = P - L.s;
```

```
101
        return fabs(v1 ^ v2) / v1.Abs();
102
    // 点到直线最近的点
103
104
    const Point nearestPointToLine(const Point &P, const Line &L) {
        Point P2 = P + getNormalVector(L.e - L.s);
105
        return getLineIntersection(Line(P, P2), L);
106
107
    // 点到线段的距离
108
109
    const double getDistanceFromPointToSegment(const Point &P, const
      Segment &L) {
        if(L.s == L.e) return (P - L.s).Abs();
110
        Vector v1 = L.e - L.s, v2 = P - L.s, v3 = P - L.e;
111
        if(sign(v1 * v2) < 0) return v2.Abs();</pre>
112
113
        114
        return fabs(v1 ^ v2) / v1.Abs();
115
    // 点到线段最近的点
116
    const Point nearestPointToSegment(const Point &P, const Segment &L) {
117
        if(L.s == L.e) return L.s;
118
        Vector v1 = L.e - L.s, v2 = P - L.s, v3 = P - L.e;
119
120
        if(sign(v1 * v2) < 0) return L.s;
121
        if(sign(v1 * v3) > 0) return L.e;
122
        return nearestPointToLine(P, L);
123
    // 判断点在线端上
124
125
    const bool isPointOnSegment(const Point &P, const Segment &L) {
        Vector v1 = P - L.s, v2 = P - L.e;
126
127
        return sign(v1 ^ v2) == 0 && sign(v1 * v2) < 0;
128
    // 判断线段相交
129
130
    const bool isSegmentIntersection(const Segment &L1, const Segment &L2)
        return std::max(L1.s.x, L1.e.x) >= std::min(L2.s.x, L2.e.x)
131
            && std::max(L1.s.y, L1.e.y) >= std::min(L2.s.y, L2.e.y)
132
            && std::max(L2.s.x, L2.e.x) >= std::min(L1.s.x, L1.e.x)
133
134
            && std::max(L2.s.y, L2.e.y) >= std::min(L1.s.y, L1.e.y)
            && sign((L1.e - L1.s) ^ (L2.s - L1.s)) * sign((L1.e - L1.s) ^ (
135
               L2.e - L1.s)) <= 0
            && sign((L2.e - L2.s) ^ (L1.s - L2.s)) * sign((L2.e - L2.s) ^ (
136
              L1.e - L2.s) <= 0;
137
138
    // 判断直线和线段相交
    const bool isLineSegmentIntersection(const Line &L1, const Segment &L2)
139
        return sign((L1.s - L2.s) ^ (L1.e - L2.s)) * <math>sign((L1.s - L2.e) ^ (L1.e - L2.s)) 
140
          L1.e - L2.e) <= 0;
141
   }
142
143
    struct Circle {
144
        Point o; double r;
        Circle() {}
145
        Circle(const Point &_o, const double &_r) { o = _o; r = _r; }
146
        Point getPoint(const double &alpha) const {
147
```

```
148
            return Point(o.x + r * cos(alpha), o.y + r * sin(alpha));
149
        }
150
    };
    // 直线和圆的交(切)点, 返回交点个数, p1和p2为两个交点
151
    const int getLineCircleIntersection(const Line &L, const Circle &C,
152
       Point &p1, Point &p2) {
        double d = getDistanceFromPointToLine(C.o, L);
153
154
        if(dcmp(d, C.r) > 0) return 0;
        Point P = nearestPointToLine(C.o, L);
155
156
        if(dcmp(d, C.r) == 0) {
            p1 = p2 = P;
157
158
            return 1;
159
        Vector v = L.e - L.s; v = v / v.Abs();
160
        double length = sqrt(C.r * C.r - d * d);
161
        p1 = P + v * length; p2 = P - v * length;
162
163
        return 2;
164
    // 两个圆的交点,返回交点个数,p1和p2为两个交点
165
    const int getCircleIntersection(const Circle &C1, const Circle &C2,
166
      Point &p1, Point &p2) {
167
        double d = (C1.o - C2.o).Abs();
        if(sign(d) == 0) {
168
            if(dcmp(C1.r, C2.r) == 0) return -1; // 重合
169
170
            return 0;
171
        }
        if(dcmp(C1.r + C2.r, d) < 0) return 0;</pre>
172
173
        if(dcmp(fabs(C1.r - C2.r), d) > 0) return 0;
        double a = (C2.o - C1.o).Angle();
174
175
        double da = acos((C1.r * C1.r + d * d - C2.r * C2.r) / (2 * C1.r *
           d));
176
        p1 = C1.getPoint(a - da); p2 = C1.getPoint(a + da);
177
        if(p1 == p2) return 1; else return 2;
178
179
    const int getRelationBetweenCircles(const Circle &C1, const Circle &C2)
180
        double d = (C1.o - C2.o).Abs(), r1 = C1.r, r2 = C2.r;
        if(sign(d) == 0) {
181
            // 0 重合 - d == 0 && r1 == r2
182
            if(dcmp(r1, r2) == 0) return 0;
183
            // 1 同心圆 - d == 0 && r1 != r2
184
185
            else return 1;
186
        }
        // 2 内含 - |r1 - r2| > d
187
        if(dcmp(fabs(r1 - r2), d) > 0) return 2;
188
        // 3 内切 - /r1 - r2/ == d
189
        if (dcmp(fabs(r1 - r2), d) == 0) return 3;
190
        // 4 相交 - r1 + r2 > d && |r1 - r2| < d
191
192
        if(dcmp(r1 + r2, d) > 0 \&\& dcmp(fabs(r1 - r2), d) < 0) return 4;
        // 5 外切 - r1 + r2 == d
193
194
        if(dcmp(r1 + r2, d) == 0) return 5;
        // 6 相离 - r1 + r2 < d
195
        if(dcmp(r1 + r2, d) < 0) return 6;
196
```

```
197
   |}
198
    // 三角形外接圆
199
200
    const Circle getCircumscribedCircle(const Point &A, const Point &B,
       const Point &C) {
        Point AB = (A + B) / 2, AC = (A + C) / 2;
201
202
        Vector NAB = getNormalVector(B - A);
203
        Vector NAC = getNormalVector(C - A);
        Point O = getLineIntersection(Line(AB, AB + NAB), Line(AC, AC + NAC
204
        return Circle(0, (0 - A).Abs());
205
206
    // 角BAC的角平分线
207
    const Line getAngleDividingLine(const Point &B, const Point &A, const
208
       Point &C) {
        Vector AB = B - A, AC = C - A;
209
210
        return Line(A, C.Rotate(A, ((B - A).Angle() - (C - A).Angle()) / 2)
           );
211
    // 三角形内接圆
212
213
    const Circle getInscribedCircle(const Point &A, const Point &B, const
       Point &C) {
        Line BAC = getAngleDividingLine(B, A, C);
214
        Line ABC = getAngleDividingLine(A, B, C);
215
        Point 0 = getLineIntersection(BAC, ABC);
216
        return Circle(0, getDistanceFromPointToLine(0, Line(B, C)));
217
218
219
    // 多边形的有向面积
220
221
    const double getPolygonArea(const Point *poly, const int &n) {
222
        double area = 0;
223
        for(int i = 1; i < n - 1; i++)
224
            area += (poly[i] - poly[0]) ^ (poly[i+1] - poly[0]);
225
        return area / 2.0;
226
    // 判断点在多边形内
227
228
    const int isPointInPolygon(const Point &p, const Point *poly, const int
        &n) {
        int wn = 0;
229
        for(int i = 0; i < n; i++) {</pre>
230
            if(isPointOnSegment(p, Segment(poly[i], poly[(i+1)%n]))) return
231
                2; // on border
            int k = sign((poly[(i+1)%n] - poly[i]) ^ (p - poly[i]));
232
233
            int d1 = sign(poly[i].y - p.y);
            int d2 = sign(poly[(i+1)%n].y - p.y);
234
            if(k > 0 \&\& d1 <= 0 \&\& d2 > 0) wn++;
235
236
            if(k < 0 && d2 <= 0 && d1 > 0) wn—;
237
238
        if(wn != 0) return 1; // inside
239
        return 0; // outside
240
241
242 | int main() {
```

```
243 | return 0;
244 |}
```

5.2 Convex Hull - Andrew

```
1 #include <cstdio>
   #include <cmath>
2
   #include <algorithm>
3
4
5
   const double EPS = 1e-10;
6
   int sign(double x) {
7
       if(fabs(x) < EPS) return 0;</pre>
       return x > 0 ? 1 : -1;
8
9
   int dcmp(double x, double y) {
10
11
       return sign(x - y);
12
   struct Point {
13
14
       double x, y;
15
       Point() {}
       Point(double _x, double _y) { x = _x; y = _y; }
16
       bool operator < (const Point &b) const {</pre>
17
18
           if(dcmp(x, b.x) == 0) return dcmp(y, b.y) < 0;
           return dcmp(x, b.x) < 0;
19
20
       }
       Point operator - (const Point &b) const {
21
           return Point(x - b.x, y - b.y);
22
23
       Point operator + (const Point &b) const {
24
25
           return Point(x + b.x, y + b.y);
26
27
       double operator ^ (const Point &b) const {
28
           return y * b.x - x * b.y;
29
       }
30
   };
      凸包Andrew算法,输入点p[],n个点,输出点ch,返回个数
31
   // p[]中不能有重复点, 执行完成后顺序被破坏
32
   // 两个 < 改成 <= 可以让凸包边上含有点
33
   int Andrew(Point *p, int n, Point *ch) {
34
35
       std::sort(p, p + n);
       // n = std::unique(p, p + n) - p;
36
       int m = 0;
37
38
       for(int i = 0; i < n; i++) {</pre>
39
           while(m > 1 && sign((ch[m-1]-ch[m-2]) ^{\circ} (p[i]-ch[m-2])) < 0) m
              --;
40
           ch[m++] = p[i];
41
       }
       int k = m;
42
43
       for(int i = n-2; i >= 0; i--) {
           while(m > k && sign((ch[m-1]-ch[m-2]) ^ (p[i]-ch[m-2])) < 0) m
44
45
           ch[m++] = p[i];
```

5.3 Halfplane Intersection

```
1 #include <cstdio>
   #include <cstring>
2
   #include <cmath>
3
   #include <vector>
5
   #include <algorithm>
6
   using std::vector;
7
8
   const double EPS = 1e-10, INF = 1e20;
9
   const double PI = acos(-1.0);
   const int sign(const double &x) {
10
       if(fabs(x) < EPS) return 0;</pre>
11
       return x < 0 ? -1 : 1:
12
13
14
   const int dcmp(const double &x, const double &y) {
15
       return sign(x - y);
16
   struct Point {
17
18
       double x, y;
19
       Point() {}
20
       Point(const double &_x, const double &_y) { x = _x; y = _y; }
       void Read() { scanf("%lf%lf", &x, &y); }
21
       bool operator < (const Point &b) const {</pre>
22
           if(dcmp(x, b.x) == 0) return dcmp(y, b.y) < 0;
23
            return dcmp(x, b.x) < 0;
24
25
       Point operator + (const Point &b) const {
26
            return Point(x + b.x, y + b.y);
27
28
29
       Point operator — (const Point &b) const {
            return Point(x - b.x, y - b.y);
30
31
       }
32
       Point operator * (const double &k) const {
33
            return Point(x * k, y * k);
34
35
       Point operator / (const double &k) const {
            return Point(x / k, y / k);
36
37
       double operator * (const Point &b) const {
38
39
            return x * b.x + y * b.y;
40
41
       double operator ^ (const Point &b) const {
```

```
42
           return x * b.y - y * b.x;
43
       }
44
       bool operator == (const Point &b) const {
           return dcmp(x, b.x) == 0 \&\& dcmp(y, b.y) == 0;
45
46
       double Abs() const {
47
48
           return sqrt(x * x + y * y);
49
       }
50
   };
   typedef Point Vector;
51
   // 单位法向量
52
   const Vector getNormalVector(const Vector &P) {
53
       double L = P.Abs(); // `L` CANNOT BE `0` !!!!
54
55
       return Vector(-P.y / L, P.x / L);
56
57
   struct Line {
       Point P; // 直线上任意一点
58
59
       Vector v; // 直线向量, 左边为对应半平面
60
       double angle;
61
       Line() {}
       Line(const Point &_P, const Vector &_v) {
62
63
           P = P; v = v;
64
           angle = atan2(v.y, v.x);
65
       bool operator < (const Line &L) const {</pre>
66
67
           return angle < L.angle;</pre>
68
       }
69
   };
70
71
   // 判断点在直线左边(线上不算)
   const bool isPointOnLineLeft(const Point &P, const Line &L) {
72
73
       return sign(L.v ^ (P - L.P)) > 0;
74
75
   // 两直线交点 ( 假设交点唯一存在 )
76
   const Point getLineIntersection(const Line &L1, const Line &L2) {
77
       Vector u = L1.P - L2.P;
78
       double t = (L2.v ^ u) / (L1.v ^ L2.v);
       return L1.P + L1.v * t;
79
80
   // 半平面交 ( 结果在 poly , 返回顶点数 )
81
   int HalfplaneIntersection(Line *L, int n, Point *poly) {
82
       std::sort(L, L + n);
83
84
       int first, last;
85
       Point *p = new Point[n];
       Line *q = new Line[n];
86
       q[first=last=0] = L[0];
87
       for(int i = 1; i < n; i++) {</pre>
88
           while(first < last && !isPointOnLineLeft(p[last-1], L[i])) last</pre>
89
90
           while(first < last && !isPointOnLineLeft(p[first], L[i])) first</pre>
              ++;
91
           q[++last] = L[i];
           if(sign(q[last].v ^ q[last-1].v) == 0) {
92
```

```
ACM-ICPC Template
93
                last--;
94
                if(isPointOnLineLeft(L[i].P, q[last])) q[last] = L[i];
95
            if(first < last) p[last-1] = getLineIntersection(q[last-1], q[
96
               last]);
97
        while(first < last && !isPointOnLineLeft(p[last-1], q[first])) last</pre>
98
99
        if(last - first <= 1) return 0; // 空集
100
        p[last] = getLineIntersection(q[last], q[first]); //
           计算首位两个半平面交点
        // 保存答案
101
        for(int i = first; i <= last; i++) poly[i-first] = p[i];</pre>
102
        return last - first + 1;
103
104
105
106
    int main() {
107
        return 0;
108
   | }
```

String 6

6.1 HASH

```
P = 102929; mod1 = 10000000000 + 7; mod2 = 10000000000 + 9;
```

Minimum Representation - 最小表示法 6.2

```
1
   namespace MinimumRepresentation{
       int get(int *s, int l)
2
3
            int i = 0, j = 1, k = 0, t;
4
5
            while(i < l && j < l && k < l) {
6
                t = s[(i + k) >= l ? i + k - l : i + k] - s[(j + k) >= l ?
                   j + k - l : j + k];
                if(!t) k++;
7
                else{
8
9
                     if(t > 0) i = i + k + 1;
                     else j = j + k + 1;
10
                    if(i == j) ++ j;
11
                    k = 0;
12
                }
13
14
            }
            return (i < j ? i : j);
15
16
       }
17 | }
```

Manacher 6.3

```
1 | #include <cstdio>
   #include <algorithm>
2
   // HDU 3068
3
   const int N = 110000 + 10;
4
5
   char t[N], s[2*N];
6
7
   int n, p[2*N];
8
   void pre(char *origin, char *str, int &_len)
9
10
       _len = 0;
11
       str[_len++] = '$';
12
       for(int i = 0; origin[i]; i++)
13
14
       {
            str[_len++] = '#';
15
            str[_len++] = origin[i];
16
17
       }
       str[_len++] = '#';
18
19
       str[len] = 0;
       //puts(str);
20
21
   }
22
   void getPi(char *str, int _len, int *_P)
23
24
25
       int mx = 0, id;
26
       for(int i = 1; i < _len; i++)</pre>
27
28
            if(mx > i) _P[i] = std::min(_P[2*id-i], mx-i);
29
            else _P[i] = 1;
            for(; str[i+_P[i]] == str[i-_P[i]]; _P[i]++);
30
            if(_P[i] + i > mx)
31
32
            {
                mx = P[i] + i;
33
                id = i;
34
35
            }
36
       }
37
38
   int main()
39
40
       while(scanf("%s", t) == 1)
41
42
       {
43
            pre(t, s, n);
44
            getPi(s, n, p);
45
            int res = 1;
            for(int i = 1; i < n; i++)</pre>
46
                res = std::max(res, p[i]-1);
47
48
            printf("%d\n", res);
49
       }
50
       return 0;
51 | }
```

6.4 KMP

```
1 #include <cstdio>
   #include <cstring>
   // POJ 3461 : Count the number of t occurrences in s
   char s[1000000+10], t[1000000+10];
4
   int next[1000000+10];
5
6
7
   void getNext(char *t, int len, int *Next)
8
9
       memset(Next, 0, sizeof(Next)); Next[0] = -1;
10
       for(int j = 0, k = -1; j < len; )
11
            if(k == -1 \mid | t[j] == t[k]) Next[++j] = ++k;
12
            else k = Next[k];
13
14
15
   int kmp(char *s, int lens, char *t, int lent)
16
17
       int res = 0;
18
       getNext(t, lent, next);
19
       for(int i = 0, j = 0; i < lens; )</pre>
20
21
       {
            if(j == -1 \mid \mid s[i] == t[j]) \{ i++; j++; \}
22
23
            else j = next[j];
24
            if(j == lent) res++; // Bingo! [pos = j - lent]
25
26
       return res;
27
   }
28
29
   int main()
30
       int T; scanf("%d", &T);
31
       while (T--)
32
33
       {
34
            scanf("%s%s", t, s);
            printf("%d\n", kmp(s, strlen(s), t, strlen(t)));
35
36
37
       return 0;
38
  | }
```

6.5 Suffix Array

```
1 #include <cstdio>
 #include <algorithm>
2
  #include <map>
  using std::map;
4
  // POJ 3261 找重复了K次的最长子串
  const int N = 20000 + 10;
6
7
  /*
8
      sa[rank[i]] = i
                      : rank i is s[j, n)
9
      sa[i] = j
```

```
10
       rank[j] = i
                     : s[j, n) is rank i
       height[i] = j : the longest common prefix of string rank _i and
11
          _i-1
   */
12
13
   int sa[N], rank[N];
14
15
   int c[N], tmp[N];
16
   int height[N];
17
   bool cmp(int *r, int a, int b, int l)
18
19
       return r[a] == r[b] && r[a+l] == r[b+l];
20
21
22
23
   void DA(int *s, int n, int m) // s[0...n-1] E [1, m)
24
25
       int i, j, p, *x = rank, *y = tmp;
       for(i = 0; i < m; i++) c[i] = 0;
26
27
       for(i = 0; i < n; i++) c[x[i] = s[i]]++;
       for(i = 1; i < m; i++) c[i] += c[i-1];
28
29
       for(i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
30
       for(j = 1, p = 0; p < n; j *= 2, m = p)
31
       {
32
            for(p = 0, i = n-j; i < n; i++) y[p++] = i;
           for(i = 0; i < n; i++) if(sa[i] >= j) y[p++] = sa[i] - j;
33
           for(i = 0; i < m; i++) c[i] = 0;
34
35
           for(i = 0; i < n; i++) c[x[y[i]]]++;</pre>
36
           for(i = 1; i < m; i++) c[i] += c[i-1];
37
           for(i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
38
           for(std::swap(x, y), p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
39
                x[sa[i]] = cmp(y, sa[i], sa[i-1], j) ? p - 1 : p++;
40
41
       for(i = 0; i < n; i++) rank[sa[i]] = i;</pre>
42
43
       int k = 0; height[0] = 0;
44
       for(i = 0; i < n; height[rank[i++]] = k) if(rank[i])</pre>
45
           for(k ? k— : 0, j = sa[rank[i]-1]; s[j+k] == s[i+k]; k++);
46
47
   int n, K, a[N];
48
49
   map<int, int> hash;
50
   bool check(int len)
51
52
53
       int cnt = 0;
       for(int i = 1; i < n; i++)</pre>
54
55
56
           if(height[i] >= len) cnt++;
57
           else cnt = 0;
           if(cnt >= K - 1) return 1;
58
59
60
       return 0;
61 | }
```

```
62
   int Solve()
63
64
   {
        int low = 0, high = n, ans = 0;
65
66
        while(low <= high)</pre>
        {
67
68
            int mid = low + (high - low) / 2;
69
            if(check(mid)) { low = mid + 1; ans = mid; }
            else high = mid - 1;
70
71
72
        return ans;
73
   }
74
75
   int main()
76
   {
77
        //----Read--
        scanf("%d%d", &n, &K);
78
79
        for(int i = 0; i < n; i++)</pre>
80
            scanf("%d", &a[i]);
81
82
            tmp[i] = a[i];
83
        }
84
        std::sort(tmp, tmp+n);
        int cnt = 0;
85
86
        for(int i = 0; i < n; i++)</pre>
            if(i == 0 \mid | tmp[i] != tmp[i-1]) hash[tmp[i]] = ++cnt;
87
        for(int i = 0; i < n; i++) a[i] = hash[a[i]];</pre>
88
89
        a[n++] = 0; /////////
90
        DA(a, n, cnt+1);
91
        for(int i = 0; i < n; i++)
92
        {
            printf("rank = %d \rightarrow [%d, %d) [%d] :", i, sa[i], n, height[i]);
93
            for(int j = sa[i]; j < n; j++) printf(" %d", a[j]);
94
95
            puts("");
96
            */
97
        printf("%d\n", Solve());
98
        return 0;
99 | }
```

6.6 Aho-Corasick Automaton

```
1 #include <cstdio>
  #include <cstring>
2
3
  #include <queue>
4
   using std::queue;
   // HDU 2222 查询 n 个模式串中有几个在原串 str 中出现了
5
   struct ACG{
6
7
       int count;
       ACG *fail, *next[26];
8
9
       ACG()
10
       {
11
          fail = 0;
```

```
12
            count = 0;
            for(int i = 0; i < 26; i++) next[i] = 0;</pre>
13
14
   }*root;
15
   queue < ACG*> Q;
16
17
   void insert(char *str, ACG *p)
18
19
        int len = strlen(str);
20
21
        for(int i = 0; i < len; i++)</pre>
22
23
             int x = str[i] - 'a';
            if(!p \rightarrow next[x]) p \rightarrow next[x] = new ACG;
24
25
            p = p -> next[x];
26
        }
27
        p -> count ++;
28
29
30
   void build_acg()
31
   {
32
        while(!Q.empty()) Q.pop();
33
        Q.push(root);
        while(!Q.empty())
34
35
        {
            ACG *p = Q.front(); Q.pop();
36
            for(int i = 0; i < 26; i++)
37
38
39
                 if(p -> next[i])
40
                 {
41
                      if(p == root) p -> next[i] -> fail = root;
42
                      else{
                          ACG *temp = p \rightarrow fail;
43
44
                          while(temp)
45
                          {
                               if(temp -> next[i])
46
47
48
                                    p -> next[i] -> fail = temp -> next[i];
49
                                    break;
50
51
                               temp = temp -> fail;
52
53
                          if(!temp) p -> next[i] -> fail = root;
54
55
                      Q.push(p -> next[i]);
                 }
56
            }
57
58
        }
59
60
61
   int query(char *str, ACG *p)
62
        int len = strlen(str), res = 0;
63
        for(int i = 0; i < len; i++)</pre>
64
```

ACM-ICPC Template GuessEver

```
65
         {
             int x = str[i] - 'a';
66
             while(!p -> next[x] && p != root) p = p -> fail;
67
             p = p \rightarrow next[x];
68
69
             if(!p) p = root;
70
             ACG *temp = p;
71
             while (temp != root \&\& temp -> count != -1)
72
73
                  res += temp -> count;
74
                  temp \rightarrow count = -1;
75
                  temp = temp -> fail;
             }
76
77
78
         return res;
79
80
81
    int n;
    char tmp[1000000+10];
82
83
    int main()
84
85
    {
86
         int T; scanf("%d", &T);
         while (T--)
87
         {
88
89
             root = new ACG;
             scanf("%d", &n);
90
             for(int i = 1; i <= n; i++)</pre>
91
92
             {
93
                  scanf("%s", tmp);
94
                  insert(tmp, root);
95
96
             build_acg();
97
             scanf("%s", tmp);
             printf("%d\n", query(tmp, root));
98
99
         }
100
         return 0;
101
```

7 Tools

7.1 BigInteger - C++

```
1 //程序中全部为正整数之间的操作
2 #include <cstdio>
3 #include <cstring>
4 #include <algorithm>
5 using std::max;
6
7 const int base = 10000; // 压4位
8
9 struct BigInt{
```

```
10
        int c[1000], len, sign;
        BigInt() { memset(c, 0, sizeof(c)); len = 1; sign = 0; }
11
        void Zero()
12
        {
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
                c[len] += (s[i]-'0') * k;
22
23
                k *= 10;
24
                if(k == base)
25
26
                     k = 1;
27
                     len++;
28
                }
            }
29
30
        }
31
        void Read()
32
        {
            char s[5000] = \{0\};
33
            scanf("%s", s);
34
35
            writein(s);
36
37
        void Print()
38
        {
39
            if(sign) printf("-");
40
            printf("%d", c[len]);
            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
47
            sprintf(s, "%d", a);
48
            writein(s);
49
            return *this;
50
51
        bool operator > (const BigInt &b)
52
        {
53
            if(len != b.len) return len > b.len;
            for(int i = len; i >= 1; i--)
54
55
            {
                if(c[i] != b.c[i]) return c[i] > b.c[i];
56
57
58
            return 0;
59
60
        bool operator < (const BigInt &b)</pre>
        {
61
            if(len != b.len) return len < b.len;</pre>
62
```

```
63
             for(int i = len; i >= 1; i--)
64
                 if(c[i] != b.c[i]) return c[i] < b.c[i];</pre>
65
66
67
             return 0;
68
69
        bool operator == (const BigInt &b)
70
71
             if(len != b.len) return 0;
             for(int i = 1; i <= len; i++)</pre>
72
73
                 if(c[i] != b.c[i]) return 0;
74
             return 1;
75
        bool operator == (const int &a)
76
77
        {
78
             BigInt b; b = a;
79
             return *this == b;
80
        BigInt operator + (const BigInt &b)
81
82
83
             BigInt r; r.len = max(len, b.len) + 1;
84
             for(int i = 1; i <= r.len; i++)</pre>
             {
85
                 r.c[i] += c[i] + b.c[i];
86
                 r.c[i+1] += r.c[i] / base;
87
                 r.c[i] %= base;
88
89
90
             r.Zero();
91
             return r;
92
93
        BigInt operator + (const int &a)
94
95
             BigInt b; b = a;
             return *this + b;
96
97
        }
98
        BigInt operator - (const BigInt &b)
99
        {
100
             BigInt a, c;// a-c
             a = *this; c = b;
101
             if(a < c)
102
103
104
                 std::swap(a, c);
105
                 a.sign = 1;
106
             for(int i = 1; i <= len; i++)</pre>
107
108
109
                 a.c[i] -= c.c[i];
                 if(a.c[i] < 0)
110
111
                 {
112
                      a.c[i] += base;
113
                      a.c[i+1]--;
114
                 }
             }
115
```

```
116
             a.Zero();
117
             return a;
118
119
         BigInt operator - (const int &a)
120
121
             BigInt b; b = a;
             return *this - b;
122
123
124
         BigInt operator * (const BigInt &b)
125
126
             BigInt r; r.len = len + b.len + 2;
             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
134
             for(int i = 1; i <= r.len; i++)</pre>
135
136
                 r.c[i+1] += r.c[i] / base;
137
                 r.c[i] %= base;
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;
150
             if(b == 0) return r;
151
             r.len = len;
             for(int i = len; i >= 1; i--)
152
153
             {
154
                 t = t * base + c[i];
155
                 int div;
                 //----try--
156
157
                      int up = 10000, down = 0;
158
                      while(up >= down)
159
160
                          int mid = (up + down) / 2;
                          BigInt ccc ; ccc = b * mid;
161
                          if(ccc > t) up = mid -1;
162
                          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
173
            r.Zero();
174
            return r;
175
176
        BigInt operator / (const int &a)
177
        {
178
            BigInt b; b = a;
            return *this / b;
179
180
        }
        BigInt operator % (const BigInt &b)
181
        {//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
187
            BigInt b; b = a;
            return *this % b;
188
189
        }
190
    };
191
192
    int main()
193
194
        return 0;
195 | }
```

7.2 C++ 读入优化

```
1
 inline int nextInt()
2
  {
3
      char ch = getchar(); int res = 0; bool sign = 0;
      while(!isdigit(ch) && ch != '-') ch = getchar();
4
5
      if(ch == '-') { sign = 1; ch = getchar(); }
      do res = (res << 1) + (res << 3) + ch - '0';</pre>
6
7
      while(isdigit(ch = getchar()));
      return sign ? -res : res;
8
9
  }
```

7.3 C char*

7.4 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)
8
  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
16 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"));
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 )
```

7.5 Java

7.5.1 The overall framework

```
1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4 public class Main{
    public static void main(String args[])
6 {
7 }
8 }
```

7.5.2 Input and Output

```
1 | Scanner cin = new Scanner(System.in);
  Scanner cin = new Scanner(new BufferedInputStream(System.in));
2
  Scanner cin = new Scanner(new File("data.in"));
3
5
   PrintWriter cout = new PrintWriter(System.out);
   PrintWriter cout = new PrintWriter(new BufferedOutputStream(System.out)
6
7
   PrintWriter cout = new PrintWriter(new File("data.out"));
8
9
   int n = cin.nextInt();
   String s = cin.next();
10
   double m = cin.nextDouble();
   String line = cin.nextLine(); // 读一整行
   BigInteger c = cin.nextBigInteger();
14 | while(cin.hasNext()) {};
```

```
15
   //PrintWriter 用 cout.println(...);
16
   System.out.println(n + "-->" + s "-->" + m);
17
18
   //使用format控制格式,与C/C++一样,double用%f,
19
   System.out.format("%03d", c).println();
20
  System.out.format("%.3f", c).println();
21
22
   //变量声明
23
24
  int a, b[] = new int[100];
  |double a, b[] = new double[100];
  |int a[][] = new int[100][100];
26
27
  String ...
28 | BigInteger / BigDecimal ...
   7.5.3 BigInteger
1 | BigInteger a = BigInteger.valueOf(100);
  BigInteger b = BigInteger.valueOf(50);
2
3
  BigInteger ONE = BigInteger.ONE;
4 | BigInteger TWO = BigInteger.valueOf(2);
5 a = a.add(ONE).subtract(b);
  a = a.multiply(TWO).divide(TWO);
7
  a = a.mod(TWO);
  8
9 | //BigDecimal 为高精小数
   7.5.4 String
1 | String s = "abcdefg"; // 注意 0下标!
  |char c = s.charAt(2);// 相当于 `char c = s[2] `(C++)(c = 'c')
2
3
  char ch[];
  ch = s.toCharArray(); // 字符串转换为字符数组
5 | for(int i = 0; i < ch.length; i++) ch[i] += 2;
  System.out.println(ch); // 输出cdefghi
7 | String tmp1 = s.substring(1); // bcdefg
8 | String tmp2 = s.substring(2, 4); // cd
   7.5.5 Hexadecimal Conversion
1 | import java.io.*;
2 | import java.util.*;
   import java.math.*;
  // Binary, Octal, Decimal(Integer/BigInteger), Hexadecimal
5
   public class Main{
       public static void main(String args[])
6
7
8
           //Decimal(123) to Others
9
           String a1 = Integer.toBinaryString(123);
           String a2 = Integer.toOctalString(123);
10
```

ACM-ICPC Template GuessEver

```
11
           String a3 = Integer.toHexString(123);
           //Others to Decimal(123)
12
           int b1 = Integer.valueOf("1111011", 2);
13
           int b2 = Integer.valueOf("173", 8);
14
           int b3 = Integer.valueOf("7b", 16);
15
           // Others to BigInteger(Decimal(123))
16
           BigInteger c1 = new BigInteger("1111011", 2);
17
18
           BigInteger c2 = new BigInteger("173", 8);
           BigInteger c3 = new BigInteger("7B", 16);
19
20
       }
21 | }
```

7.5.6 function

```
1 Arrays.fill(a, x); // for(int i = 0; i < N; i++) a[i] = x;
   Arrays.fill(a, l, r, x); // for(int i = l; i < r; i++) a[i] = x;
  Arrays.sort(a); // 给a的所有元素排序 升序
   Arrays.sort(a, l, r); // 给a的[l, r)元素排序 升序
4
  Arrays.sort(a, l, r, new cmp());
5
6
   import java.io.*;
7
   import java.util.*;
8
   import java.math.*;
9
10
   class INT{
11
       int s;
       public INT(int x) { s = x; }// 构造函数 INT a = new INT(3);
12
13
   class cmp implements Comparator<INT>{
14
       public int compare(INT a, INT b)
15
       {
16
17
           return a.s - b.s;
18
       }
19
20
   public class Main{
       public static void main(String args[])
21
22
       {
           Scanner cin = new Scanner(System.in);
23
24
           int n;
25
           INT a[] = new INT[100];
           for(int i = 1; i <= 10; i++) a[i] = new INT(11 - i);</pre>
26
27
           Arrays.sort(a, 1, 11, new cmp());
       }
28
29
   //a[i].s排序前10 9 8 7 6 5 4 3 2 1
30
31
   //a[i].s排序后1 2 3 4 5 6 7 8 9 10
32
  |String s = Integer.toString(n, B); // 把十进制数 n转换成 B进制数
33
34 | int b = Integer.parseInt(s, B); // 把B进制数 s转换成 10进制数
```

7.6 Batch test

7.6.1 @Linux

```
|mkdata=mk
   filea=a
2
   fileb=b
3
4
   g++ $mkdata.cpp —o $mkdata
5
   g++ $filea.cpp -o $filea
   g++ $fileb.cpp —o $fileb
7
8
   cas=0
9
   while true; do
10
        ./$mkdata > $filea.in
       ./$filea < $filea.in > $filea.out
11
        ./$fileb < $filea.in > $fileb.out
12
       if ! diff $filea.out $fileb.out
13
14
       then
            echo "uWronguAnswer"
15
16
            break
17
       fi
       echo $((cas=cas+1)) "⊔Accepted"
18
19
   done
```

7.6.2 @Windows

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

7.7 Vimrc Config For Linux

```
1
  filetype on
   filetype indent on
2
   set nobackup
3
   set nu
4
   set st=4
   set ts=4
6
7
   set sw=4
8
   map <F7> <Esc>:w<CR>:!javac %:r.java<CR>:!java %:r<CR>
9
   imap <F7> <Esc>:w<CR>:!javac %:r.java<CR>:!java %:r<CR>
10
   map <F8> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!gdb %:r<CR>
11
   imap <F8> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!gdb %:r<CR>
12
   map <F9> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!./%:r<CR>
```

ACM-ICPC Template GuessEver

```
14 | imap <F9> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!./%:r<CR>
15 | map <c-a> <Esc>gg"+yG
16 | imap<sub>\(\sigma\)</sub><c-a><sub>\(\sigma\)</sub><Esc>gg"+yG
```

8 WHAT THE FUCK!!!

8.1 不用递归的 DFS - dfsWithoutDfs.cpp

```
void dfs()
1
2
3
        st.clear();
        DFS.push(std::make_pair(root, 0));
4
5
        int END = 3;
        while(!DFS.empty())
6
7
        {
            pair < int , int > & now = DFS.top();
8
            if(now.second != END)
9
10
            {
                 ++now.second;
11
                 DFS.push(std::make_pair(SON[now.first], 0));
12
13
14
            else DFS.pop();
15
        }
16 | }
```

8.2 时间结构体

```
1 | const char mon[][5] = {"", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "
      Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
   const int days[] = {0, 31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31};
2
   struct Date{
3
       int year, month, day, hour;
4
5
       bool check(int y)
       {
6
7
           return (y % 4 == 0 && y % 100 != 0) || y % 400 == 0;
8
       int getHours() // to 2000.01.01 0 o'clock
9
10
           int hours = 1;
11
           for(int i = 2000; i < year; i++) hours += 24 * (365 + check(i))</pre>
12
           for(int i = 1; i < month; i++)
13
                hours += 24 * (days[i] + (i == 2 && check(year)));
14
           return hours += 24 * (day - 1) + hour;
15
       }
16
17 | };
```