ACM-ICPC Template

GuessEver

Last Modify at July 27, 2015

Contents

1 Dynamic Programming **3**
1.1 LCS - Longest Common Subsequence . . . 3
1.2 LIS - Longest Increasing Subsequence . . . 3
1.3 Maximum Continuous Subsequence Sum . . . 4
1.4 RMQ - st . . . 4
1.5 数位 dp . . . 5
1.6 状压 dp . . . 6
1.6.1 枚举子集 . . . 6

2 Math **6**
2.1 GCD && LCM . . . 6
2.1.1 GCD - Greatest Common Divisor . . . 6
2.1.2 LCM - Least Common Multiple . . . 6
2.1.3 E_GCD - Extended Greatest Common Divisor . . . 6
2.2 Prime . . . 6
2.2.1 Make Prime List . . . 6
2.2.2 Prime Factor . . . 7
2.3 Fast Power . . . 7
2.4 约瑟夫环、丢手绢问题 . . . 7
2.5 康拓展开 Cantor . . . 8

3 Datastructure **9**
3.1 带权并查集 . . . 9
3.2 手写 Heap . . . 10
3.3 Leftist Tree . . . 11
3.4 Partition Tree . . . 12
3.5 Treap . . . 13
3.5.1 @ Array . . . 13
3.5.2 @ Pointer . . . 15
3.6 Size Balanced Tree . . . 17
3.7 三维偏序 - CDQ 分治 . . . 18

4 Graph **20**
4.1 Shortest path . . . 20
4.1.1 Dijkstra . . . 20
4.1.2 Spfa . . . 21
4.1.3 Floyd . . . 21
4.2 Minimum Spanning Tree . . . 22
4.2.1 Prim . . . 22
4.2.2 Kruskal . . . 22
4.3 Tarjan - Strong Union . . . 23
4.4 LCA . . . 24
4.4.1 @ Tarjan . . . 24
4.4.2 @ Doubling Algorithm . . . 25
4.5 Bipartite Graph . . . 27
4.5.1 Maximal Matching - The Hungarian algorithm . . . 27
4.5.2 Optimal Matching - KM . . . 28
4.6 Network Flow . . . 28
4.6.1 Maximum Flow - isap . . . 28
4.6.2 Minimum Cost Maximum Flow - spfa . . . 28

5 Geometry **30**
5.1 Convex Hull 30
5.2 All 31

6 String **36**
6.1 Manacher 36
6.2 KMP 37
6.3 Suffix Array 38
6.4 Aho-Corasick Automaton 40

7 Tools **42**
7.1 BigInteger - C++ 42
7.2 C++ 读入优化 46
7.3 C char* 46
7.4 C++ std::string 46
7.5 Java 47
7.5.1 The overall framework 47
7.5.2 Input and Output 47
7.5.3 BigInteger 48
7.5.4 String 48
7.5.5 Hexadecimal Conversion 48
7.5.6 function 49
7.6 Batch test 49
7.6.1 @Linux 49
7.6.2 @Windows 50
7.7 Vimrc Config For Linux 50

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
    int f[N][N];
    int res = 0;
    for(int i = 1; i < lena; i++)
        for(int j = 1; j < lenb; j++)
        {
            if(a[i] == a[j]) f[i][j] = f[i-1][j-1] + 1;
            else f[i][j] = max(f[i-1][j], f[i][j-1]);
            res = max(res, f[i][j]);
        }
    return res;
}

int LCS() // O(NlogN)
{//把 LCM 转化为 LIS 来做
// 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()
    //------change------
        //这里就要针对数据自己想尽办法转化了
        for(int i = 1; i <= n; i++) h[a[i]] = i;
        for(int i = 1; i <= n; i++) b[i] = h[b[i]];
    //------end------
    return LIS();
}
```

1.2 LIS - Longest Increasing Subsequence

```
int f[N];
int LIS()//O(N*N)
{
    for(int i = 1; i <= n; i++)
        for(int j = i-1; j >= 1; j--)
            if(a[i] > a[j]) f[i] = max(f[i], f[j] + 1);
    int res = 0;
    for(int i = 1; i <= n; i++) res = max(res, f[i]);
    return res;
}

int c[N], len = 0;
int LIS()//(NlogN)
{
    for(int i = 1; i <= n; i++)
    {
        //------find------
            int l = 1, r = len, mid;
            while(l <= r)
            {
```

```
                mid = (l + r) / 2;
                if(a[i] > c[mid]) l = mid + 1;
                else r = mid - 1;
            }
        //------end------
        c[l] = a[i];
        len = max(len, l);
    }
    return len;
}
```

1.3 Maximum Continuous Subsequence Sum

```
int MaxSubSum()
{
    int f[N], res;
    for(int i = 1; i <= n; i++)
    {
        f[i] = max(a[i], f[i-1] + a[i]);
        res = max(res, f[i]);
    }
    return res;
}

int MaxSubSum()
{
    int res = 0, now = 0;
    for(int i = 1; i <= n; i++)
    {
        now += a[i];
        res = max(res, now);
        if(now < 0) now = 0;
    }
    return res;
}
```

1.4 RMQ - st

```
int _rmq[N][30];
void init_RMQ(int *_orig) // [1, n]
{
    for(int i = 1; i <= n; i++) _rmq[i][0] = _orig[i];
    for(int j = 1; j <= log(1.0 * n) / log(2.0); j++)
        for(int i = 1; i <= n + 1 - (1 << j); i++)
            _rmq[i][j] = std::max(_rmq[i][j-1], _rmq[i+(1<<(j-1))][j
               -1]);
}
int query_RMQ(int l, int r) // max{x E [l, r]}
{
    int k = log(r - l + 1.0) / log(2.0);
    return std::max(_rmq[l][k], _rmq[r-(1<<k)+1][k]);
```

```
}
```

1.5 数位 dp

```
#include <cstdio>
#include <cstring>
#include <algorithm>
// calculate the number of numbers in [l, r] which not contain '4' or '62'
long long l, r;
int k;
int L[100], R[100];
long long f[100][2][2][10];

int predo(long long a, int *num)
{
    int len = 0;
    do num[++len] = a % 10; while(a /= 10);
    return len;
}

long long calc(int pos, bool d, bool u, int pre)
{
    if(pos == 0) return 1;
    long long &res = f[pos][d][u][pre];
    if(res != -1) return res;
    res = 0;
    int st = d ? L[pos] : 0;
    int ed = u ? R[pos] : 9;
    for(int i = st; i <= ed; i++)
    {
        if(i == 4 || (pre == 6 && i == 2)) continue;
        res += calc(pos-1, d && i == L[pos], u && i == R[pos], i);
    }
    return res;
}

int main()
{
    while(scanf("%lld%lld", &l, &r) == 2 && (l || r))
    {
        memset(f, -1, sizeof(f));
        memset(L, 0, sizeof(L));
        memset(R, 0, sizeof(R));
        int len = std::max(predo(l, L), predo(r, R));
        printf("%lld\n", calc(len, 1, 1, 0));
    }
    return 0;
}
```

1.6 状压 dp

1.6.1 枚举子集

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

2 Math

2.1 GCD && LCM

2.1.1 GCD - Greatest Common Divisor

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

2.1.2 LCM - Least Common Multiple

```
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
  ==> ay1 + b(x1-a/b*y1) = 1
对应  ax  + by          = 1

int egcd(int a, int b, int &x, int &y)
{
    if(b == 0)
    {
        x = 1; y = 0;
        return a;
    }
    int x1, y1;
    int e = egcd(b, a%b, x1, y1);
    x = y1;
    y = x1 - a / b * y1;
    return e;
}
```

2.2 Prime

2.2.1 Make Prime List

```
void make_prime_list(int maxp) // O(2*N)
{
    for(int i = 2; i <= maxp; i++)
    {
        if(!h[i]) pri[l++] = i;
        for(int j = 0; j < l && pri[j] <= maxp / i; j++)
```

```
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();
4       for(int j = 0; j < Cnt && pri[j]*pri[j] <= n; j++)
5       {
6           if(n % pri[j] == 0)
7           {
8               printf("%d␣", pri[j]);
9               while(n % pri[j] == 0) n /= pri[j];
10          }
11      }
12      if(n!=1) printf("%d",n);
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;
5       if(y & 1) return (mul((x * (LL)x) % mod, y / 2, mod) * (LL)x)%mod;
6       else return mul((x * (LL)x) % mod, y / 2, mod) % mod;
7   }
8   int mul(int x, int y, int mod) // 非递归
9   {
10      int s = 1;
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>
2   //UVALive 4727
3   int n, m;
```

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

2.5 康拓展开 Cantor

```
1  #include <cstdio>
2  #include <cstring>
3
4  int fac[10], a[10];
5
6  bool Read(int *p)
7  {
8      for(int i = 0; i < 9; i++)
9      {
10         char chtmp;
11         if(scanf("␣%c", &chtmp) != 1) return 0;
12         p[i] = chtmp == 'x' ? 0 : chtmp - '0';
13     }
14     return 1;
15 }
16
17 int Cantor(int *p) // Eight puzzle status -> Integer
18 {
19     int res = 0;
20     for(int i = 0; i < 9; i++)
21     {
22         int cnt = 0;
23         for(int j = i + 1; j < 9; j++)
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 {
33     for(int i = 0, j = 0; i < 9; i++)
34     {
35         if(!used[i] && j == r) return i;
36         if(!used[i]) j++;
37     }
38 }
39 void getStatus(int cantor, int *p) // Integer -> Eight puzzle status
40 {
41     memset(used, 0, sizeof(used));
42     for(int i = 0; i < 9; i++)
43     {
44         p[i] = getRank(cantor / fac[9 - i - 1]);
45         used[p[i]] = 1;
46         cantor %= fac[9 - i - 1];
47     }
48 }
49
50 void PRINT(int *p)
51 {
52     int hash = Cantor(p);
53     printf("Cantor value = %d\n", hash);
54     getStatus(hash, p);
55     printf("Cantor Status = ");
56     for(int i = 0; i < 9; i++) printf("%d", p[i]); puts("");
57 }
58
59 int main()
60 {
61     fac[0] = 1; for(int i = 1; i < 10; i++) fac[i] = fac[i-1] * i;
62     while(Read(a)) PRINT(a);
63     return 0;
64 }
```

3 Datastructure

3.1 带权并查集

```
1 #include <cstdio>
2 #include <cstdlib>
3
4 const int N = 100000 + 10;
5
6 int n, f[N], g[N];
7
8 int getroot(int x)
9 {
10     if(f[x] == x) return x;
```

```
    int tmp = getroot(f[x]);
    g[x] += g[f[x]]; // update the value
    return f[x] = tmp;
}

void merge(int x, int y) // merge x's set and y's set
{ // Guarantee that the x must be the root of its set, which means x == getroot(x) is true, but it may not be same for y
    int fy = getroot(y);
    g[x] += g[y] + abs(x - y) % 1000; // update the value
    f[x] = fy;
}

int main()
{
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) f[i] = i;
    char op; int x, y;
    while(scanf("␣%c", &op) == 1 && op != 'O')
    {
        if(op == 'I')
        {
            scanf("%d%d", &x, &y);
            if(getroot(x) == getroot(y)) continue;
            merge(x, y);
        }
        else{
            scanf("%d", &x);
            getroot(x); // !!! update the value of x before output
            printf("%d\n", g[x]);
        }
        //for(int i = 1; i <= n; i++) printf("%d ", f[i]); puts("");
    }
    return 0;
}
```

3.2 手写 Heap

```
#include <cstdio>
#include <algorithm>

const int N = 250000;

int n, a[N], x, size = 0;

void update(int i)
{
    while(i > 1 && a[i] > a[i/2])
    {
        std::swap(a[i], a[i/2]);
        i /= 2;
    }
```

```
}

void pop()
{
    int i = 1; a[i] = 0;
    while(i * 2 <= size && (a[i] < a[i*2] || a[i] < a[i*2+1]))
    {
        if(i * 2 == size || (i * 2 < size && a[i*2] >= a[i*2+1]))
        {
            a[i] = a[i*2];
            a[i*2] = 0;
            i = i * 2;
        }
        else {
            a[i] = a[i*2+1];
            a[i*2+1] = 0;
            i = i * 2 + 1;
        }
    }
    a[i] = a[size]; size--;
    update(i);
}

int main()
{
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
    {
        scanf("%d", &x);
        a[++size] = x;
        update(size);
    }
    for(int i = 1; i <= n; i++)
    {
        printf("%d ", a[1]);
        pop();
    }
    return 0;
}
```

3.3 Leftist Tree

```
//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]);}

//把x和y合并在一起，其实就是把y插入x
int merge(int x,int y)//返回合并后子树的根
{
    if(!x || !y) return x|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，
       不如让x插入y
    R[x]=merge(R[x],y);//始终往右子树合并
```

```
    f[R[x]] = x;//更新并查集
    if(D[R[x]] > D[L[x]]) swap(L[x],R[x]);//保持左偏树性质
    D[x] = D[R[x]] + 1;
    若还有其他维护信息也需要更新;
    return x;//返回根
}

int del(int x)
{
    int t = merge(L[x],R[x]);
    f[L[x]] = L[x]; f[R[x]] = R[x];//更新并查集
    L[x] = R[x] = D[x] = 0;
    return t;
}
```

3.4 Partition Tree

```
struct Parti{int val, left;} val[30][N];
void build_tree(int d, int l, int r)
{
    if(l == r) return;
    int m = (l + r) >> 1, same = m - l + 1;
    int lcnt = l, rcnt = m + 1;
    for(int i = l; i <= r; i++)
        if(val[d][i].val < sorted[m]) same--;
    for(int i = l; i <= r; i++)
    {
        int flag = 0;
        if((val[d][i].val < soted[m]) || (val[d][i].val == sorted[m] &&
            same))
        {
            flag = 1;
            val[d + 1][lcnt++] = val[d][i];
            if(val[d][i].val == sorted[m]) same--;
        }
        else val[d][rcnt++] = val[d][i];
        val[d][i].left = val[d][i - 1].left + flag;
    }
    build_tree(d + 1, l, m);
    build_tree(d + 1, m + 1, r);
}
int query(int d, int l, int r, int x, int y, int k)
{
    if(l == r) return val[d][l].val;
    int m = (l + r) >> 1;
    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
    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

```
#include <cstdio>
#include <cstdlib>
#include <ctime>

const int N = 100000 + 10;

int m, Limit;
int L[N], R[N], S[N], fix[N], key[N];
int root, total, leave;

void rotate_left(int &p)
{
    int tmp = R[p];
    R[p] = L[tmp];
    int zsize = S[L[tmp]];
    S[p] = S[p] - S[tmp] + zsize;
    L[tmp] = p;
    S[tmp] = S[tmp] - zsize + S[p];
    p = tmp;
}
void rotate_right(int &p)
{
    int tmp = L[p];
    L[p] = R[tmp];
    int zsize = S[R[tmp]];
    S[p] = S[p] - S[tmp] + zsize;
    R[tmp] = p;
    S[tmp] = S[tmp] - zsize + S[p];
    p = tmp;
}

void insert(int &p, int x)
{
    if(!p)
    {
        p = ++total;
        L[p] = R[p] = 0;
        S[p] = 1;
        fix[p] = rand();
        key[p] = x;
        return;
    }
    S[p]++;
    if(x < key[p])
    {
        insert(L[p], x);
        if(fix[L[p]] > fix[p]) rotate_right(p);
    }
    else {
```

```
        insert(R[p], x);
        if(fix[R[p]] > fix[p]) rotate_left(p);
    }
}

void remove(int &p, int limit)
{
    if(!p) return;
    if(key[p] < limit)
    {
        leave += S[L[p]] + 1;
        p = R[p];
        remove(p, limit);
    }
    else{
        remove(L[p], limit);
        S[p] = S[L[p]] + S[R[p]] + 1;
    }
}

int kth(int &p, int k)
{
    if(k <= S[L[p]]) return kth(L[p], k);
    else if(k == S[L[p]] + 1) return key[p];
    else return kth(R[p], k - S[L[p]] - 1);
}

int main()
{
    srand(time(0));
    scanf("%d%d", &m, &Limit);
    int delta = 0;
    while(m--)
    {
        char op; int x;
        scanf("␣%c%d", &op, &x);
        if(op == 'I')
        {
            if(x < Limit) continue;
            insert(root, x - delta);
        }
        else if(op == 'A') delta += x;
        else if(op == 'S')
        {
            delta -= x;
            remove(root, Limit - delta);
        }
        else {
            x = S[root] - x + 1;
            if(x <= 0) puts("-1");
            else printf("%d\n", kth(root, x) + delta);
        }
    }
```

```
103     printf("%d\n", leave);
104     return 0;
105 }
```

3.5.2 @ Pointer

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <ctime>
4
5  int m, Limit;
6  struct Treap{
7      int fix, key, size;
8      Treap *left, *right;
9  }*root, *null;
10 int leave;
11
12 void rotate_left(Treap *&p)
13 {
14     Treap *tmp = p -> right;
15     p -> right = tmp -> left;
16     int zsize = tmp -> left -> size;
17     p -> size = p -> size - tmp -> size + zsize;
18     tmp -> left = p;
19     tmp -> size = tmp -> size - zsize + p -> size;
20     p = tmp;
21 }
22 void rotate_right(Treap *&p)
23 {
24     Treap *tmp = p -> left;
25     p -> left = tmp -> right;
26     int zsize = tmp -> right -> size;
27     p -> size = p -> size - tmp -> size + zsize;
28     tmp -> right = p;
29     tmp -> size = tmp -> size - zsize + p -> size;
30     p = tmp;
31 }
32
33 void insert(Treap *&p, int x)
34 {
35     if(p == null)
36     {
37         p = new Treap;
38         p -> fix = rand();
39         p -> key = x;
40         p -> size = 1;
41         p -> left = null;
42         p -> right = null;
43         return;
44     }
45     if(x < p -> key)
46     {
```

```
47          insert(p -> left, x);
48          p -> size++;
49          if(p -> left -> fix > p -> fix) rotate_right(p);
50      }
51      else {
52          insert(p -> right, x);
53          p -> size++;
54          if(p -> right -> fix > p -> fix) rotate_left(p);
55      }
56  }
57
58  void remove(Treap *&p, int L)
59  {
60      if(p == null) return;
61      if(p -> key < L)
62      {
63          leave += p -> left -> size + 1;
64          p = p -> right;
65          remove(p, L);
66      }
67      else {
68          remove(p -> left, L);
69          p -> size = p -> left -> size + p -> right -> size + 1;
70      }
71  }
72
73  int kth(Treap *&p, int k)
74  {
75      int Lsize = p -> left -> size;
76      if(k <= Lsize) return kth(p -> left, k);
77      else if(k == Lsize + 1) return p -> key;
78      else return kth(p -> right, k - Lsize - 1);
79  }
80
81  int main()
82  {
83      srand(time(0));
84      null = new Treap; root = null;
85      scanf("%d%d", &m, &Limit);
86      int delta = 0;
87      while(m--)
88      {
89          char op; int x;
90          scanf("␣%c%d", &op, &x);
91          if(op == 'I')
92          {
93              if(x < Limit) continue;
94              insert(root, x - delta);
95          }
96          else if(op == 'A') delta += x;
97          else if(op == 'S')
98          {
99              delta -= x;
```

```
            remove(root, Limit - delta);
        }
        else {
            x = root -> size - x + 1;
            if(x <= 0) puts("-1");
            else printf("%d\n", kth(root, x) + delta);
        }
    }
    printf("%d\n", leave);
    return 0;
}
```

3.6 Size Balanced Tree

```
int A[N], S[N], L[N], R[N], root, total;
void rotate_left(int &x)
{
    int y = R[x];
    R[x] = L[y];
    L[y] = x;
    S[y] = S[x];
    S[x] = S[L[x]] + S[R[x]] + 1;
    x = y;
}
void rotate_right(int &x)
{
    int y = L[x];
    L[x] = R[y];
    R[y] = x;
    S[y] = S[x];
    S[x] = S[L[x]] + S[R[x]] + 1;
    x = y;
}

void maintain(int &p, bool flag)
{
    if(flag)//调整右边
    {
        if(S[R[R[p]]] > S[L[p]] rotate_left(p);
                else if(S[R[L[p]]] > S[L[p]])
        {
            rotate_right(R[p]);
                rotate_left(p);
            }
            else return;
            }
    else
    {
        if(S[L[L[p]]] > S[R[p]]) rotate_right(p);
        else if(S[L[R[p]]] > S[R[p]])
        {
            rotate_left(L[p]);
```

```
            rotate_right(p);
        }
        else return;
    }
    maintain(L[p], 0);
    maintain(R[p], 1);
    maintain(p, 0);
    maintain(p, 1);
}

void insert(int &p, int e)
{
    if(!p)
    {
        p = ++total;
        L[p] = R[p] = 0;
        A[p] = e; S[p] = 1;
        return;
    }
    S[p]++;
    if(e < A[p]) insert(L[p], e);
    else insert(R[p], e);
    maintain(p, k >= A[p]);
}

int getmin()
{
    for(int x = root; L[x]; x = L[x]);
    return A[x];
}
int getmax()
{
    for(int x = root; R[x]; x = R[x]);
    return A[x];
}
int kth(int &p, int k)
{
    int tmp = S[L[p]] + 1;
    if(k == tmp) return A[p];
    else if(k < tmp) return kth(L[p], k);
    else return kth(R[p], k - tmp);
}
```

3.7 三维偏序 - CDQ 分治

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#define lowbit(_X) ((_X)&(-(_X)))
// SPOJ LIS2
const int N = 100000 + 10;

```

```
 8 int n, f[N], idx[N], hash[N];
 9 struct Node{
10     int x, y, z;
11     void Read(int i)
12     {
13         scanf("%d%d", &y, &z);
14         x = i; f[i] = 1; idx[i] = i;
15     }
16 }a[N];
17 int maxp;
18 int c[N]; // tree Array
19
20 bool cmpx(int i, int j) { return a[i].x < a[j].x; }
21 bool cmpy(int i, int j) { return a[i].y < a[j].y; }
22 bool cmpz(int i, int j) { return a[i].z < a[j].z; }
23
24 void discrete()
25 {
26     std::sort(idx+1, idx+1+n, cmpy); maxp = 0;
27     for(int i = 1; i <= n; i++)
28     {
29         if(i == 1 || a[idx[i]].y != a[idx[i-1]].y) hash[idx[i]] = ++
              maxp;
30         else hash[idx[i]] = maxp;
31     }
32     for(int i = 1; i <= n; i++) a[idx[i]].y = hash[idx[i]];
33     std::sort(idx+1, idx+1+n, cmpz); maxp = 0;
34     for(int i = 1; i <= n; i++)
35     {
36         if(i == 1 || a[idx[i]].z != a[idx[i-1]].z) hash[idx[i]] = ++
              maxp;
37         else hash[idx[i]] = maxp;
38     }
39     for(int i = 1; i <= n; i++) a[idx[i]].z = hash[idx[i]];
40 }
41
42 void insert(int a, int x)
43 {
44     for( ; a <= maxp; a += lowbit(a)) c[a] = std::max(c[a], x);
45 }
46 int query(int a) // [1, a]
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 {
55     std::sort(&idx[l], &idx[mid]+1, cmpy);
56     std::sort(&idx[mid+1], &idx[r]+1, cmpy);
57     // [l, mid] .. calculated ok
58     // now calculating [mid+1, r]
```

```
59      // f[i] = max{f[j]} + 1;
60      int j = l;
61      for(int i = mid + 1; i <= r; i++)
62      {
63          for( ; j <= mid && a[idx[j]].y < a[idx[i]].y; j++)
64              insert(a[idx[j]].z, f[a[idx[j]].x]);
65          int tmp = query(a[idx[i]].z - 1);
66          if(tmp + 1 > f[a[idx[i]].x]) f[a[idx[i]].x] = tmp + 1;
67      }
68      //memset(c, 0, sizeof(c));
69      for(int i = l; i <= mid; i++)
70      {
71          int b = a[idx[i]].z;
72          for( ; b <= maxp; b += lowbit(b)) c[b] = 0;
73      }
74      std::sort(&idx[mid+1], &idx[r]+1, cmpx);
75      // CDQ(mid+1, r) next, so sort back it
76  }
77
78  void CDQ(int l, int r)
79  {
80      if(l == r) return;
81      int mid = l + (r - l) / 2;
82      CDQ(l, mid);
83      solve(l, mid, r);
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);
91      discrete();
92      std::sort(idx+1, idx+1+n, cmpx);
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

```
1  void dijkstra()
2  {
3      memset(dist, 0x3f, sizeof(dist));
4      dist[1] = 0; Q.push(make_pair(0, 1));
```

```
    while(!Q.empty())
    {
        int x = Q.top().second; Q.pop();
        if(done[x]) continue;
        done[x] = 1;
        for(Link p = head[x]; p; p = p->next)
            if(dist[p->y] > dist[x] + p->z)
            {
                dist[p->y] = dist[x] + p->z;
                Q.push(make_pair(-dist[p->y], p->y));
            }
    }
}
```

4.1.2 Spfa

```
void spfa()
{
    memset(inQ, 0, sizeof(inQ));
    memset(dist, 0x3f, sizeof(dist));
    dist[S] = 0; Q.push(S); inQ[S] = 1; //S为源点
    while(!Q.empty())
    {
        int x = Q.front(); Q.pop(); inQ[x] = 0;
        for(Link p = head[x]; p; p = p->next)
            if(dist[p->y] > dist[x] + p->z)
            {
                dist[p->y] = dist[x] + p->z;
                if(!inQ[p->y])
                {
                    Q.push(p->y);
                    inQ[p->y] = 1;
                }
            }
    }
}
```

4.1.3 Floyd

```
void floyd()
{
    for(int k = 1; k <= n; k++) // 这里可以看作是一个加边的过程
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= n; j++)
                map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
}

// 最小环
void MinCircle()
{
    cap[] = map[];
```

```
    int circle = 0x3f3f3f3f;
    for(int k = 1; k <= n; k++)
    {
        for(int i = 1; i < k; i++)
            for(int j = i+1; j < k; j++)
                circle = min(circle, map[i][j] + cap[j][k]+cap[k][i]);
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= n; j++)
                map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
    }
    return circle == 0x3f3f3f3f ? -1 : circle;
}

// floyd判圈法 （大白书 p44）
void Circle()
{
    int ans = k;
    int k1 = k, k2 = k;
    do{
        k1 = next(k1);
        k2 = next(k2); ans = max(ans, k2);
        k2 = next(k2); ans = max(ans, k2);
    }while(k1 != k2);
    return ans;
}
```

4.2 Minimum Spanning Tree

4.2.1 Prim

```
int prim()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[1] = 0; Q.push(make_pair(0, 1));
    int res = 0;
    while(!Q.empty())
    {
        int x = Q.top().second; Q.pop();
        if(done[x]) continue;
        res += dist[x]; done[x] = 1;
        for(Link p = head[x]; p; p = p->next)
            if(dist[p->y] > p->z)
            {
                dist[p->y] = p->z;
                Q.push(make_pair(-dist[p->y], p->y));
            }
    }
    return res;
}
```

4.2.2 Kruskal

```
int kruskal()
{
    sort(edge, edge+Cnt, cmp);
    int res = 0;
    for(int i = 0; i < Cnt; i++)
    {
        if(getroot(edge[i].x) == getroot(edge[i].y)) continue;
        f[getroot(edge[i].x)] = getroot(edge[i].y);
        res += edge[i].z;
    }
    return res;
}
```

4.3 Tarjan - Strong Union

```
void dfs(int x)
{
    now[x] = low[x] = ++dfstime;
    hash[x] = 1;
    st.push(x); inst[x] = 1;
    for(int i = 1; i <= n; i++)
        if(map[x][i])
        {
            if(!hash[i])
            {
                dfs(i);
                low[x] = min(low[x], low[i]);
            }
            else if(inst[i]) low[x] = min(low[x], now[i]);
        }
    if(low[x] == now[x])
    {
        while(!st.empty())
        {
            int u = st.top();
            st.pop(); inst[u] = 0;
            belong[u] = number;
            if(u == x) break;
        }
        numer++;
    }
}
void tarjan()
{
    for(int i = 1; i <= n; i++)
        if(!hash[i]) dfs(i);
    if(!st.empty()) // 这是一个未知bug   栈中还会剩下一个强连通分量
    {
        while!st.empty())
        {
            int u = st.top();
            st.pop();
```

```
            belong[u] = number;
        }
        number++;
    }
}
```

4.4 LCA

4.4.1 @ Tarjan

```
// poj 1330 (changed something)
// LCA tarjan
#include <cstdio>
#include <cstring>

const int N = 10000 + 10;

int n;
struct Link{int y, idx; Link *next;}*head[N], *ask[N];
int tx, ty;
bool in[N], vis[N];
int f[N];
int ans[N]; // Query Answer

void inLink(int x, int y)
{
    Link *p = new Link;
    p -> y = y;
    p -> next = head[x];
    head[x] = p;
}
void inAsk(int x, int y, int idx)
{
    Link *p = new Link;
    p -> y = y;
    p -> idx = idx;
    p -> next = ask[x];
    ask[x] = p;
}

int getroot(int x)
{
    return f[x] == x ? x : f[x] = getroot(f[x]);
}

void LCA(int x)
{
    vis[x] = 1;
    f[x] = x;
    for(Link *p = ask[x]; p; p = p -> next)
        if(vis[p->y]) ans[p->idx] = getroot(p->y);
    for(Link *p = head[x]; p; p = p -> next)
```

```
        if(!vis[p->y])
        {
            LCA(p->y);
            f[p->y] = x;
        }
}

int main()
{
    int T; scanf("%d", &T);
    while(T--)
    {
        memset(head, 0, sizeof(head));
        memset(ask, 0, sizeof(ask));
        memset(in, 0, sizeof(in));
        memset(vis, 0, sizeof(vis));
        scanf("%d", &n);
        for(int i = 1; i <= n; i++) f[i] = i;
        for(int i = 1; i < n; i++)
        {
            int x, y;
            scanf("%d%d", &x, &y);
            inLink(x, y);
            in[y] = 1;
        }
        int q = 1;// the number of query
        for(int i = 1; i <= q; i++)
        {
            int x, y; scanf("%d%d", &x, &y);
            inAsk(x, y, i); inAsk(y, x, i);
        }
        int root = -1;
        for(int i = 1; i <= n; i++)
            if(!in[i]) {root = i; break;}
        LCA(root);
        for(int i = 1; i <= q; i++)
            printf("%d\n", ans[i]);
    }
    return 0;
}
```

4.4.2 @ Doubling Algorithm

```
#include <cstdio>
#include <cstring>
#include <algorithm>
// POJ 1330   LCA_Doubling Algorithm
const int N = 10000 + 10;

const int UPDeepth = 14;
int n;
struct Link{
```

```
    int y;
    Link *next;
}*head[N];
bool in[N];
int ancient[N][UPDeepth+1];
int deep[N];

void inLink(int x, int y)
{
    Link *p = new Link;
    p -> y = y;
    p -> next = head[x];
    head[x] = p;
}

void dfs(int x, int deepth, int father)
{
    deep[x] = deepth;
    ancient[x][0] = father;
    for(Link *p = head[x]; p; p = p -> next)
        dfs(p -> y, deepth + 1, x);
}

void getLCA()
{
    for(int i = 1; i <= n; i++)
        if(!in[i]) dfs(i, 1, 0);
    for(int j = 1; j <= UPDeepth; j++)
        for(int i = 1; i <= n; i++)
            ancient[i][j] = ancient[ancient[i][j-1]][j-1];
}

int LCA(int x, int y)
{
    if(deep[x] > deep[y]) std::swap(x, y); // deep[x] <= deep[y]
    for(int j = UPDeepth; j >= 0; j--)
        if(deep[x] <= deep[ancient[y][j]]) y = ancient[y][j];
    if(x == y) return x;
    for(int j = UPDeepth; j >= 0; j--)
        if(ancient[x][j] != ancient[y][j])
        {
            x = ancient[x][j];
            y = ancient[y][j];
        }
    return ancient[y][0];
}

int main()
{
    int T; scanf("%d", &T);
    while(T--)
    {
        memset(head, 0, sizeof(head));
```

```
        memset(ancient, 0, sizeof(ancient));
        memset(in, 0, sizeof(in));
        memset(deep, 0, sizeof(deep));
        scanf("%d", &n);
        for(int i = 1; i < n; i++)
        {
            int x, y; scanf("%d%d", &x, &y);
            inLink(x, y); in[y] = 1;
        }
        getLCA();
        int x, y; scanf("%d%d", &x, &y);
        printf("%d\n", LCA(x, y));
    }
    return 0;
}
```

4.5 Bipartite Graph

4.5.1 Maximal Matching - The Hungarian algorithm

```
int timeStamp = 0;
int n, m, g[N][N];
int vis[N], pre[N];

bool search(int x)
{
    for(int i = 1; i <= m; i++)
        if(g[x][i] && vis[i] != timeStamp)
        {
            vis[i] = timeStamp;
            if(pre[i] == -1 || search(pre[i]))
            {
                pre[i] = x;
                return 1;
            }
        }
    return 0;
}

int maxMatch()
{
    int res = 0;
    memset(pre, -1, sizeof(pre));
    for(int i = 1; i <= n; i++)
    {
        ++timeStamp;
        res += search(i);
    }
    return res;
}
```

4.5.2 Optimal Matching - KM

不会... 用费用流解决

4.6 Network Flow

4.6.1 Maximum Flow - isap

```cpp
#include <cstdio>
#include <algorithm>

const int N = 200 + 10;

int n, m, g[N][N];
int v[N], h[N];
int S, T;

int sap(int x, int flow)
{
    if(x == n) return flow;
    int res = 0;
    for(int i = S; i <= T; i++)
        if(g[x][i] && h[x] == h[i] + 1)
        {
            int t = sap(i, std::min(g[x][i], flow - res));
            res += t; g[x][i] -= t; g[i][x] += t;
            if(res == flow) return res;
            if(h[S] >= T) return res;
        }
    //if(h[S] >= T) return res;
    if((--v[h[x]]) == 0) h[S] = T;
    ++v[++h[x]];
    return res;
}

int main()
{
    scanf("%d%d", &m, &n); // m = number of edges, n = number of points
    for(int i = 1; i <= m; i++)
    {
        int x, y, z;
        scanf("%d%d%d", &x, &y, &z);
        g[x][y] += z;
    }
    v[0] = T; S = 1; T = n;
    int maxflow = 0;
    while(h[S] < T) maxflow += sap(1, 0x3f3f3f3f);
    printf("%d\n", maxflow);
    return 0;
}
```

4.6.2 Minimum Cost Maximum Flow - spfa

```
1  struct EG{int from,to,flow,cost,next;}edge[M];
2
3  void add_edge(int a,int b,int c,int d)
4  {
5      edge[L]=(EG){a,b,c,+d,head[a]};
6      head[a]=L++;
7      edge[L]=(EG){b,a,0,-d,head[b]};
8      head[b]=L++;
9  }
10
11 bool spfa()
12 {
13     memset(inQ, 0, sizeof(inQ));
14     memset(dist, 0x3f, sizeof(dist));
15     dist[S] = 0;
16     q.push(S);
17     while(!q.empty())
18     {
19         int x = q.front();
20         q.pop();
21         inQ[x] = 0;
22         for(int i = head[x]; i != -1; i = edge[i].next)
23             if(edge[i].flow && dist[edge[i].to] > dist[x] + edge[i].
                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 }
36 void MFMC()
37 {
38     memset(head, -1, sizeof(head));
39     建图调用 add_edge();
40
41     int mincost = 0, maxflow = 0;
42     while(spfa())
43     {
44         int res = inf;
45         for(int i = T; i != S; i = edge[pre[i]].from)
46         {
47             res = min(res, edge[pre[i]].flow);
48         }
49         for(int i = T; i != S; i = edge[pre[i]].from)
50         {
51             edge[pre[i]].flow -= res;
52             edge[pre[i] ^ 1].flow += res;
```

```
53          }
54          maxflow += res;
55          mincost += res * dist[T];
56      }
57  }
```

5 Geometry

5.1 Convex Hull

```
1   //◆◆□°◆ μ◆list[0~n-1]
2   //□°◆¼◆◆stack[0~top-1]
3   Point list[Maxn];
4   int Stack[Maxn],top;
5   bool _cmp (Point p1,Point p2)
6   {
7       double tmp=(p1-list[0])^(p2-list[0]);
8       if (fuhao(tmp)>0) return true;
9       else if (fuhao(tmp)==0&&fuhao(dist(p1,list[0])-dist(p2,list[0]))
            <=0)
10          return true;
11      else    return false;
12  }
13  void Graham(int n)
14  {
15      Point p0;
16      int k=0;
17      p0=list[0];
18      for (int i=1;i<n;++i)
19      {
20          if ((p0.y>list[i].y)||(p0.y==list[i].y&&p0.x>list[i].x))
21          {
22              p0=list[i];
23              k=i;
24          }
25      }
26      swap(list[k],list[0]);
27      sort(list+1,list+n,_cmp);
28      if (n==1)
29      {
30          top=1;
31          stack[0]=0;
32          return;
33      }
34      if (n==2)
35      {
36          top=2;
37          stack[0]=0;
38          stack[1]=1;
39          return;
40      }
```

```
41      stack[0]=0;
42      stack[1]=1;
43      top=2;
44      for (int i=2;i<n;++i)
45      {
46          while (top>1 && fuhao((list[stack[top-1]]-list[stack[top-2]])^(
                list[i]-list[stack[top-2]]))<=0)
47              top--;
48          stack[top++]=i;
49      }
50  }
```

5.2 All

```
1   #include <cstdio>
2   #include <cstdlib>
3   #include <cstring>
4   #include <cmath>
5   #include <algorithm>
6   #include <utility>
7   using std::max;
8   using std::min;
9   using std::sort;
10  using std::swap;
11  using std::pair;
12  using std::make_pair;
13  const double eps = 1e-8, inf = 1e20;
14  const double pi = 4.0 * atan(1.0);
15  #define Degree(_rad) (180.0 / pi * (_rad))
16
17  int fuhao(double x)
18  {
19      if (fabs(x)<eps) return 0;
20      if (x<0) return -1;
21      else return 1;
22  }
23
24  ////////////////////////   Point && Vector
        ////////////////////////
25  struct Point{
26      double x, y;
27      Point (){}
28      Point (double _x,double _y):x(_x),y(_y){}
29      void init(double a, double b) { x = a; y = b; }
30
31      // basic calc
32          bool operator == (const Point &b) const
33          {
34              return !fuhao(x - b.x) && !fuhao(y - b.y);
35          }
36          Point operator + (const Point &b) const
37          {
```

```
38              return Point(x + b.x, y + b.y);
39          }
40          Point operator – (const Point &b) const
41          {
42              return Point(x – b.x, y – b.y);
43          }
44          Point operator * (const double &b) const
45          {
46              return Point(x * b, y * b);
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;
56              return Point(nx, ny);
57          }
58
59      // Vector
60          double operator *(const Point &b)const
61          {// Dot
62              return x * b.x + y * b.y;
63          }
64          double operator ^ (const Point &b)const
65          {// Cross
66              return x * b.y – y * b.x;
67          }
68          double Abs() { return sqrt(x * x + y * y); }
69  };
70  double Dist(const Point &a, const Point &b) { return (a – b).Abs(); }
71  typedef Point Vector;
72
73  double Angle(Vector a, Vector b)
74  {
75      return acos(a * b / a.Abs() / b.Abs());
76  }
77  Vector Get_H(Vector A)
78  { // 求与向量垂直的单位向量   使用前确保不为0向量
79      // A != Vector(0.0, 0.0);
80      double L = A.Abs();
81      return Vector(–A.y / L, A.x / L);
82  }
83
84  ////////////////////////////      E – N – D
       ////////////////////////////
85
86
87  ////////////////////////////        Line
       ////////////////////////////
88  struct Line{
```

```
89      Point s,e;
90      Line() {}
91      Line(Point ss, Point ee)
92      {
93          s = ss; e = ee;
94      }
95
96      // 两直线的关系：重合0， 平行1， 相交2 并返回交点
97      pair<int,Point> operator &(const Line &b) const
98      {
99          Point ans = s;
100         if(fuhao((s-e)^(b.s-b.e))==0)
101         {
102             if (fuhao((s-b.e)^(b.s-b.e))==0)
103                 return make_pair(0,ans);//重合
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));
107         ans.x += (e.x-s.x) * t;
108         ans.y += (e.y-s.y) * t;
109         return make_pair(2,ans);//相交
110     }
111 };
112 ////////////////////////////      E - N - D
       ////////////////////////////
113
114 //判断线段相交
115 bool inter(Line l1,Line l2)
116 {
117     return
118     max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
119     max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
120     max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
121     max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
122     fuhao((l2.s-l1.e)^(l1.s-l1.e)) * fuhao((l2.e-l1.e)^(l1.s-l1.e))<=0
          &&
123     fuhao((l1.s-l2.e)^(l2.s-l2.e)) * fuhao((l1.e-l2.e)^(l2.s-l2.e))<=0;
124 }
125 //判断直线与线段相交
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)
          )<=0;
129 }
130 //点到直线距离
131 //返回点到直线最近的点
132 Point PointToLine(Point P,Line L)
133 {
134     Point ans;
135     double t=((P-L.s)*(L.e-L.s))/((L.e-L.s)*(L.e-L.s));
136     ans.x=L.s.x+(L.e.x-L.s.x)*t;
137     ans.y=L.s.y+(L.e.y-L.s.y)*t;
138     return ans;
```

```
}
//点到线段距离
//返回点到线段最近的点
Point NearestPointToLineSeg(Point P,Line L)
{
    Point ans;
    double t = ((P-L.s)*(L.e-L.s)) / ((L.e-L.s)*(L.e-L.s));
    if (t>=0&&t<=1)
    {
        ans.x = L.s.x + (L.e.x-L.s.x)*t;
        ans.y = L.s.y + (L.e.y-L.s.y)*t;
    }
    else {
        if (Dist(P,L.s)<Dist(P,L.e))
            ans = L.s;
        else    ans = L.e;
    }
    return ans;
}
//多边形面积
double CalcArea(Point p[],int n)
{
    double ans=0;
    for (int i=0;i<n;++i)
        ans+=(p[i]^p[(i+1)%n])/2;
    return fabs(ans);
}
//判断点在线段上
bool OnSeg(Point P,Line L)
{
    return
            fuhao((L.s-P)^(L.e-P))==0 &&
            fuhao((P.x-L.s.x)*(P.x-L.e.x))<=0 &&
            fuhao((P.y-L.s.y)*(P.y-L.e.y))<=0;
}
//三点求圆心坐标
Point waixin(Point a,Point b,Point c)
{
    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;
    double d=a1*b2-a2*b1;
    return Point(a.x+(c1*b2-c2*b1)/d,a.y+(a1*c2-a2*c1)/d);
}


//////////////////////////        Graham
   //////////////////////////
//求凸包 点list[0~n-1]
//凸包结果Stack[0~top-1]
const int Maxn = 100;////////////////here!!
Point list[Maxn];           //////////?!?!?!?!补全Maxn
   !?!?!?!?!?!?!?!?!?!?!
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];
203     for (int i=1;i<n;++i)
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     }
219     if (n==2)
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)
230     {
231         while (top>1 && fuhao((list[Stack[top-1]]-list[Stack[top-2]])^(
            list[i]-list[Stack[top-2]]))<=0)
232             top--;
233         Stack[top++]=i;
234     }
235 }
236 //////////////////////////      E - N - D
    //////////////////////////
237
238
```

```
239 ////////////////////////////          Area
       ////////////////////////////
240 double PolygonArea(Point *pp, int nn) // pp[0, n-1]
241 {
242     double ans_area = 0.0;
243     for(int i = 1; i < nn-1; i++)
244     {
245         ans_area += (pp[i] - pp[0]) ^ (pp[i+1] - pp[0]);
246     }
247     return fabs(ans_area / 2);
248 }
249 ////////////////////////////      E - N - D
       ////////////////////////////
250
251 ////////////////////////////     点在多边形内
       ////////////////////////////
252 int isPointInPolygon(Point p, Point *poly, int nn)
253 {
254     int w = 0;
255     for(int i = 0; i < n; i++)
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]));
259         int d1 = fuhao(poly[i].y - p.y);
260         int d2 = fuhao(poly[(i+1)%n].y - p.y);
261         if(k > 0 && d1 <= 0 && d2 > 0) wn++;
262         if(k < 0 && d1 > 0 && d2 <= 0) wn--;
263     }
264     if(wn != 0) return 1; //内部
265     return 0; // 外部
266 }
267 ////////////////////////////      E - N - D
       ////////////////////////////
268
269
270 int main()
271 {
272 }
```

6 String

6.1 Manacher

```
1 #include <cstdio>
2 #include <algorithm>
3 // HDU 3068
4 const int N = 110000 + 10;
5
6 char t[N], s[2*N];
7 int n, p[2*N];
8
```

```
void pre(char *origin, char *str, int &_len)
{
    _len = 0;
    str[_len++] = '$';
    for(int i = 0; origin[i]; i++)
    {
        str[_len++] = '#';
        str[_len++] = origin[i];
    }
    str[_len++] = '#';
    str[_len] = 0;
    //puts(str);
}

void getPi(char *str, int _len, int *_P)
{
    int mx = 0, id;
    for(int i = 1; i < _len; i++)
    {
        if(mx > i) _P[i] = std::min(_P[2*id-i], mx-i);
        else _P[i] = 1;
        for(; str[i+_P[i]] == str[i-_P[i]]; _P[i]++) ;
        if(_P[i] + i > mx)
        {
            mx = _P[i] + i;
            id = i;
        }
    }
}

int main()
{
    while(scanf("%s", t) == 1)
    {
        pre(t, s, n);
        getPi(s, n, p);
        int res = 1;
        for(int i = 1; i < n; i++)
            res = std::max(res, p[i]-1);
        printf("%d\n", res);
    }
    return 0;
}
```

6.2 KMP

```
#include <cstdio>
#include <cstring>
// POJ 3461 : Count the number of t occurrences in s
char s[1000000+10], t[1000000+10];
int next[1000000+10];

```

```
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     {
12         if(k == -1 || t[j] == t[k]) Next[++j] = ++k;
13         else k = Next[k];
14     }
15 }
16 int kmp(char *s, int lens, char *t, int lent)
17 {
18     int res = 0;
19     getNext(t, lent, next);
20     for(int i = 0, j = 0; i < lens; )
21     {
22         if(j == -1 || s[i] == t[j]) { i++; j++; }
23         else j = next[j];
24         if(j == lent) res++; // Bingo! [pos = j - lent]
25     }
26     return res;
27 }
28
29 int main()
30 {
31     int T; scanf("%d", &T);
32     while(T--)
33     {
34         scanf("%s%s", t, s);
35         printf("%d\n", kmp(s, strlen(s), t, strlen(t)));
36     }
37     return 0;
38 }
```

6.3 Suffix Array

```
1  #include <cstdio>
2  #include <algorithm>
3  #include <map>
4  using std::map;
5  // POJ 3261 找重复了K次的最长子串
6  const int N = 20000 + 10;
7  /*
8      sa[rank[i]] = i
9      sa[i] = j       : rank i is s[j, n)
10     rank[j] = i     : s[j, n) is rank i
11     height[i] = j   : the longest common prefix of string rank _i and
           _i-1
12 */
13
14 int sa[N], rank[N];
15 int c[N], tmp[N];
16 int height[N];
```

```
17
18 bool cmp(int *r, int a, int b, int l)
19 {
20     return r[a] == r[b] && r[a+l] == r[b+l];
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;
26     for(i = 0; i < m; i++) c[i] = 0;
27     for(i = 0; i < n; i++) c[x[i] = s[i]]++;
28     for(i = 1; i < m; i++) c[i] += c[i-1];
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;
33         for(i = 0; i < n; i++) if(sa[i] >= j) y[p++] = sa[i] - j;
34         for(i = 0; i < m; i++) c[i] = 0;
35         for(i = 0; i < n; i++) c[x[y[i]]]++;
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;
42
43     int k = 0; height[0] = 0;
44     for(i = 0; i < n; height[rank[i++]] = k) if(rank[i])
45         for(k ? k-- : 0, j = sa[rank[i]-1]; s[j+k] == s[i+k]; k++);
46 }
47
48 int n, K, a[N];
49 map<int, int> hash;
50
51 bool check(int len)
52 {
53     int cnt = 0;
54     for(int i = 1; i < n; i++)
55     {
56         if(height[i] >= len) cnt++;
57         else cnt = 0;
58         if(cnt >= K - 1) return 1;
59     }
60     return 0;
61 }
62
63 int Solve()
64 {
65     int low = 0, high = n, ans = 0;
66     while(low <= high)
67     {
68         int mid = low + (high - low) / 2;
69         if(check(mid)) { low = mid + 1; ans = mid; }
```

```
        else high = mid - 1;
    }
    return ans;
}

int main()
{
    //------Read-------
    scanf("%d%d", &n, &K);
    for(int i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
        tmp[i] = a[i];
    }
    std::sort(tmp, tmp+n);
    int cnt = 0;
    for(int i = 0; i < n; i++)
        if(i == 0 || tmp[i] != tmp[i-1]) hash[tmp[i]] = ++cnt;
    for(int i = 0; i < n; i++) a[i] = hash[a[i]];
    a[n++] = 0; ///////////
    DA(a, n, cnt+1);
/*  for(int i = 0; i < n; i++)
    {
        printf("rank = %d -> [%d, %d) [%d] :", i, sa[i], n, height[i]);
        for(int j = sa[i]; j < n; j++) printf(" %d", a[j]);
        puts("");
    }   */
    printf("%d\n", Solve());
    return 0;
}
```

6.4 Aho-Corasick Automaton

```
#include <cstdio>
#include <cstring>
#include <queue>
using std::queue;
// HDU 2222 查询 n 个模式串中有几个在原串 str 中出现了
struct ACG{
    int count;
    ACG *fail, *next[26];
    ACG()
    {
        fail = 0;
        count = 0;
        for(int i = 0; i < 26; i++) next[i] = 0;
    }
}*root;
queue<ACG*> Q;

void insert(char *str, ACG *p)
{
```

```
    int len = strlen(str);
    for(int i = 0; i < len; i++)
    {
        int x = str[i] - 'a';
        if(!p -> next[x]) p -> next[x] = new ACG;
        p = p -> next[x];
    }
    p -> count ++;
}

void build_acg()
{
    while(!Q.empty()) Q.pop();
    Q.push(root);
    while(!Q.empty())
    {
        ACG *p = Q.front(); Q.pop();
        for(int i = 0; i < 26; i++)
        {
            if(p -> next[i])
            {
                if(p == root) p -> next[i] -> fail = root;
                else{
                    ACG *temp = p -> fail;
                    while(temp)
                    {
                        if(temp -> next[i])
                        {
                            p -> next[i] -> fail = temp -> next[i];
                            break;
                        }
                        temp = temp -> fail;
                    }
                    if(!temp) p -> next[i] -> fail = root;
                }
                Q.push(p -> next[i]);
            }
        }
    }
}

int query(char *str, ACG *p)
{
    int len = strlen(str), res = 0;
    for(int i = 0; i < len; i++)
    {
        int x = str[i] - 'a';
        while(!p -> next[x] && p != root) p = p -> fail;
        p = p -> next[x];
        if(!p) p = root;
        ACG *temp = p;
        while(temp != root && temp -> count != -1)
        {
```

```
            res += temp -> count;
            temp -> count = -1;
            temp = temp -> fail;
        }
    }
    return res;
}

int n;
char tmp[1000000+10];

int main()
{
    int T; scanf("%d", &T);
    while(T--)
    {
        root = new ACG;
        scanf("%d", &n);
        for(int i = 1; i <= n; i++)
        {
            scanf("%s", tmp);
            insert(tmp, root);
        }
        build_acg();
        scanf("%s", tmp);
        printf("%d\n", query(tmp, root));
    }
    return 0;
}
```

7 Tools

7.1 BigInteger - C++

```
//程序中全部为正整数之间的操作
#include <cstdio>
#include <cstring>
#include <algorithm>
using std::max;

const int base = 10000; // 压4位

struct BigInt{
    int c[1000], len, sign;
    BigInt() { memset(c, 0, sizeof(c)); len = 1; sign = 0; }
    void Zero()
    {
        while(len > 1 && c[len] == 0) len--;
        if(len == 1 && c[len] == 0) sign = 0;
    }
    void writein(char *s)
```

```
18      {
19          int k = 1, L = strlen(s);
20          for(int i = L-1; i >= 0; i--)
21          {
22              c[len] += (s[i]-'0') * k;
23              k *= 10;
24              if(k == base)
25              {
26                  k = 1;
27                  len++;
28              }
29          }
30      }
31      void Read()
32      {
33          char s[5000] = {0};
34          scanf("%s", s);
35          writein(s);
36      }
37      void Print()
38      {
39          if(sign) printf("-");
40          printf("%d", c[len]);
41          for(int i = len-1; i >= 1; i--) printf("%04d", c[i]);
42          printf("\n");
43      }
44      BigInt operator = (int a)
45      {
46          char s[100] = {0};
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;
54          for(int i = len; i >= 1; i--)
55          {
56              if(c[i] != b.c[i]) return c[i] > b.c[i];
57          }
58          return 0;
59      }
60      bool operator < (const BigInt &b)
61      {
62          if(len != b.len) return len < b.len;
63          for(int i = len; i >= 1; i--)
64          {
65              if(c[i] != b.c[i]) return c[i] < b.c[i];
66          }
67          return 0;
68      }
69      bool operator == (const BigInt &b)
70      {
```

```
71          if(len != b.len) return 0;
72          for(int i = 1; i <= len; i++)
73              if(c[i] != b.c[i]) return 0;
74          return 1;
75      }
76      bool operator == (const int &a)
77      {
78          BigInt b; b = a;
79          return *this == b;
80      }
81      BigInt operator + (const BigInt &b)
82      {
83          BigInt r; r.len = max(len, b.len) + 1;
84          for(int i = 1; i <= r.len; i++)
85          {
86              r.c[i] += c[i] + b.c[i];
87              r.c[i+1] += r.c[i] / base;
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
101         a = *this; c = b;
102         if(a < c)
103         {
104             std::swap(a, c);
105             a.sign = 1;
106         }
107         for(int i = 1; i <= len; i++)
108         {
109             a.c[i] -= c.c[i];
110             if(a.c[i] < 0)
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;
122         return *this - b;
123     }
```

```
124     BigInt operator * (const BigInt &b)
125     {
126         BigInt r; r.len = len + b.len + 2;
127         for(int i = 1; i <= len; i++)
128         {
129             for(int j = 1; j <= b.len; j++)
130             {
131                 r.c[j+i-1] += c[i] * b.c[j];
132             }
133         }
134         for(int i = 1; i <= r.len; i++)
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     }
147     BigInt operator / (BigInt b)//整除
148     {
149         BigInt t, r;
150         if(b == 0) return r;
151         r.len = len;
152         for(int i = len; i >= 1; i--)
153         {
154             t = t * base + c[i];
155             int div;
156             //-------try-------
157                 int up = 10000, down = 0;
158                 while(up >= down)
159                 {
160                     int mid = (up + down) / 2;
161                     BigInt ccc ; ccc = b * mid;
162                     if(ccc > t) up = mid - 1;
163                     else {
164                         down = mid + 1;
165                         div = mid;
166                     }
167                 }
168             //-------end-------
169             r.c[i] = div;
170             t = t - b * div;
171         }
172         //最后的t为余数，要用的自己想办法传出去
173         r.Zero();
174         return r;
175     }
176     BigInt operator / (const int &a)
```

```
177     {
178         BigInt b; b = a;
179         return *this / b;
180     }
181     BigInt operator % (const BigInt &b)
182     {//其实可以复制上面除法的，这里换一种写法
183         return *this - *this / b * b;
184     }
185     BigInt operator % (const int &a)
186     {
187         BigInt b; b = a;
188         return *this % b;
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;
4     while(!isdigit(ch) && ch != '-') ch = getchar();
5     if(ch == '-') { sign = 1; ch = getchar(); }
6     do res = (res << 1) + (res << 3) + ch - '0';
7     while(isdigit(ch = getchar()));
8     return sign ? -res : res;
9 }
```

7.3 C char*

```
1 头文件cstring
2 strlen(s);//获取长度  O(N)
3 strcpy(a+2,b+1)//从b+1开始全部赋值给a+2开始的字符串
4 strncpy(a+2,b+1,2)//从b+1开始赋值2个给a+2开始的字符串
5 strcmp(a,b)//比较a和b的大小，相等返回0，a>b返回正整数
6 strcat(a,b)//相当于string类的 a += b;
7 strstr(a,b)-a;//返回b在a中第一次出现的位置，不存在返回NULL(即0)，由于-a
    ,所以最后应该是-a
```

7.4 C++ std::string

```
1 //====初始化====
2 头文件string并加上std::
3 string s(str);//相当于string s=str;
4 string s(cstr);//把char数组类型的字符串cstr作为s的初值
5 s.clear();//清空，相当于 s="";
```

```
6
7  //====长度====
8  s.length();//获取s的长度，O(1)
9  s.size();//一样
10
11 //====插入删除====
12 s.insert(2, "a"); //在s的位置2插入string类字符串"a"
13 s.erase(2, 3); //从s的位置2开始删除3个字符
14
15 //====查找====
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）
17 //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{
5      public static void main(String args[])
6      {
7      }
8  }
```

7.5.2 Input and Output

```
1  Scanner cin = new Scanner(System.in);
2  Scanner cin = new Scanner(new BufferedInputStream(System.in));
3  Scanner cin = new Scanner(new File("data.in"));
4
5  PrintWriter cout = new PrintWriter(System.out);
6  PrintWriter cout = new PrintWriter(new BufferedOutputStream(System.out)
       );
7  PrintWriter cout = new PrintWriter(new File("data.out"));
8
9  int n = cin.nextInt();
10 String s = cin.next();
11 double m = cin.nextDouble();
12 String line = cin.nextLine(); // 读一整行
13 BigInteger c = cin.nextBigInteger();
14 while(cin.hasNext()) {};
15
16 //PrintWriter 用 cout.println(...);
17 System.out.println(n + "-->" + s "-->" + m);
18
19 //使用format控制格式,与C/C++一样,double用%f,
20 System.out.format("%03d", c).println();
21 System.out.format("%.3f", c).println();
```

```java

//变量声明
int a, b[] = new int[100];
double a, b[] = new double[100];
int a[][] = new int[100][100];
String ...
BigInteger/BigDecimal ...
```

7.5.3 BigInteger

```java
BigInteger a = BigInteger.valueOf(100);
BigInteger b = BigInteger.valueOf(50);
BigInteger ONE = BigInteger.ONE;
BigInteger TWO = BigInteger.valueOf(2);
a = a.add(ONE).subtract(b);
a = a.multiply(TWO).divide(TWO);
a = a.mod(TWO);
a.compareTo(ONE); // 大于1, 小于-1, 等于0
//BigDecimal 为高精小数
```

7.5.4 String

```java
String s = "abcdefg"; // 注意0下标!
char c = s.charAt(2);// 相当于`char c = s[2]`(C++)(c = 'c')
char ch[];
ch = s.toCharArray(); // 字符串转换为字符数组
for(int i = 0; i < ch.length; i++) ch[i] += 2;
System.out.println(ch); // 输出cdefghi
String tmp1 = s.substring(1); // bcdefg
String tmp2 = s.substring(2, 4); // cd
```

7.5.5 Hexadecimal Conversion

```java
import java.io.*;
import java.util.*;
import java.math.*;
// Binary, Octal, Decimal(Integer/BigInteger), Hexadecimal
public class Main{
    public static void main(String args[])
    {
        //Decimal(123) to Others
        String a1 = Integer.toBinaryString(123);
        String a2 = Integer.toOctalString(123);
        String a3 = Integer.toHexString(123);
        //Others to Decimal(123)
        int b1 = Integer.valueOf("1111011", 2);
        int b2 = Integer.valueOf("173", 8);
        int b3 = Integer.valueOf("7b", 16);
        // Others to BigInteger(Decimal(123))
        BigInteger c1 = new BigInteger("1111011", 2);
```

```
        BigInteger c2 = new BigInteger("173", 8);
        BigInteger c3 = new BigInteger("7B", 16);
    }
}
```

7.5.6 function

```
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
Arrays.sort(a, l, r, new cmp());

import java.io.*;
import java.util.*;
import java.math.*;
class INT{
    int s;
    public INT(int x) { s = x; }// 构造函数 INT a = new INT(3);
}
class cmp implements Comparator<INT>{
    public int compare(INT a, INT b)
    {
        return a.s - b.s;
    }
}
public class Main{
    public static void main(String args[])
    {
        Scanner cin = new Scanner(System.in);
        int n;
        INT a[] = new INT[100];
        for(int i = 1; i <= 10; i++) a[i] = new INT(11 - i);
        Arrays.sort(a, 1, 11, new cmp());
    }
}
//a[i].s排序前10 9 8 7 6 5 4 3 2 1
//a[i].s排序后1 2 3 4 5 6 7 8 9 10

String s = Integer.toString(n, B); // 把十进制数n转换成B进制数
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
fileb=b

g++ $mkdata.cpp -o $mkdata
```

```
g++ $filea.cpp -o $filea
g++ $fileb.cpp -o $fileb
cas=0
while true; do
    ./$mkdata > $filea.in
    ./$filea < $filea.in > $filea.out
    ./$fileb < $filea.in > $fileb.out
    if ! diff $filea.out $fileb.out
    then
        echo "␣Wrong␣Answer"
        break
    fi
    echo $((cas=cas+1)) "␣Accepted"
done
```

7.6.2 @Windows

```
:loop
    mk > A.in
    A < A.in > A.out
    p < A.in > p.out
    fc A.out p.out
    if errorlevel 1 goto end
    goto loop
:end
    pause
```

7.7 Vimrc Config For Linux

```
filetype on
filetype indent on
set nobackup
set nu
set st=4
set ts=4
set sw=4

map <F7> <Esc>:w<CR>:!javac %:r.java<CR>:!java %:r<CR>
imap <F7> <Esc>:w<CR>:!javac %:r.java<CR>:!java %:r<CR>
map <F8> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!gdb %:r<CR>
imap <F8> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!gdb %:r<CR>
map <F9> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!./%:r<CR>
imap <F9> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!./%:r<CR>
map <c-a> <Esc>gg"+yG
imap␣<c-a>␣<Esc>gg"+yG
```