

# ACM-ICPC Template



GuessEver

Last Modify at June 3, 2015

# Contents

<b>1</b>	<b>Dynamic Programming</b>	<b>3</b>
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
<b>2</b>	<b>Math</b>	<b>6</b>
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
<b>3</b>	<b>Datastructure</b>	<b>9</b>
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
<b>4</b>	<b>Graph</b>	<b>20</b>
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

<b>5</b>	<b>Geometry</b>	<b>30</b>
5.1	Convex Hull . . . . .	30
5.2	All . . . . .	31
<b>6</b>	<b>String</b>	<b>36</b>
6.1	Manacher . . . . .	36
6.2	KMP . . . . .	37
6.3	Suffix Array . . . . .	38
6.4	Aho-Corasick Automaton . . . . .	40
<b>7</b>	<b>Tools</b>	<b>42</b>
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

```

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

```

## 1.2 LIS - Longest Increasing Subsequence

```

1  int f[N];
2  int LIS() //  $O(N*N)$ 
3  {
4      for(int i = 1; i <= n; i++)
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]);
9      return res;
10 }
11
12 int c[N], len = 0;
13 int LIS() //  $(N\log N)$ 
14 {
15     for(int i = 1; i <= n; i++)
16     {
17         // -----find-----
18         int l = 1, r = len, mid;
19         while(l <= r)
20         {

```

```

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

```

### 1.3 Maximum Continuous Subsequence Sum

```

1 int MaxSubSum()
2 {
3     int f[N], res;
4     for(int i = 1; i <= n; i++)
5     {
6         f[i] = max(a[i], f[i-1] + a[i]);
7         res = max(res, f[i]);
8     }
9     return res;
10 }
11
12 int MaxSubSum()
13 {
14     int res = 0, now = 0;
15     for(int i = 1; i <= n; i++)
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

```

1 int _rmq[N][30];
2 void init_RMQ(int *_orig) // [1, n]
3 {
4     for(int i = 1; i <= n; i++) _rmq[i][0] = _orig[i];
5     for(int j = 1; j <= log(1.0 * n) / log(2.0); j++)
6         for(int i = 1; i <= n + 1 - (1 << j); i++)
7             _rmq[i][j] = std::max(_rmq[i][j-1], _rmq[i+(1<<(j-1))][j-1]);
8 }
9 int query_RMQ(int l, int r) // max{x E [l, r]}
10 {
11     int k = log(r - l + 1.0) / log(2.0);
12     return std::max(_rmq[l][k], _rmq[r-(1<<k)+1][k]);

```

13 }

## 1.5 数位 dp

```

1  #include <cstdio>
2  #include <cstring>
3  #include <algorithm>
4  // calculate the number of numbers in [l, r] which not contain '4' or
   '62'
5  long long l, r;
6  int k;
7  int L[100], R[100];
8  long long f[100][2][2][10];
9
10 int predo(long long a, int *num)
11 {
12     int len = 0;
13     do num[++len] = a % 10; while(a /= 10);
14     return len;
15 }
16
17 long long calc(int pos, bool d, bool u, int pre)
18 {
19     if(pos == 0) return 1;
20     long long &res = f[pos][d][u][pre];
21     if(res != -1) return res;
22     res = 0;
23     int st = d ? L[pos] : 0;
24     int ed = u ? R[pos] : 9;
25     for(int i = st; i <= ed; i++)
26     {
27         if(i == 4 || (pre == 6 && i == 2)) continue;
28         res += calc(pos-1, d && i == L[pos], u && i == R[pos], i);
29     }
30     return res;
31 }
32
33 int main()
34 {
35     while(scanf("%lld%lld", &l, &r) == 2 && (l || r))
36     {
37         memset(f, -1, sizeof(f));
38         memset(L, 0, sizeof(L));
39         memset(R, 0, sizeof(R));
40         int len = std::max(predo(l, L), predo(r, R));
41         printf("%lld\n", calc(len, 1, 1, 0));
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
2 | bx1 + (a%b)y1 = 1    ==>    bx + (a-a/b*b)y = 1
3 |    ==> ay1 + b(x1-a/b*y1) = 1
4 | 对应  ax  + by          = 1
5 |
6 | int egcd(int a, int b, int &x, int &y)
7 | {
8 |     if(b == 0)
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;
16 |     y = x1 - a / b * y1;
17 |     return e;
18 | }
```

## 2.2 Prime

### 2.2.1 Make Prime List

```
1 | void make_prime_list(int maxp) // O(2*N)
2 | {
3 |     for(int i = 2; i <= maxp; i++)
4 |     {
5 |         if(!h[i]) pri[l++] = i;
6 |         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 <stdio>
2 //UVALive 4727
3 int n, m;

```



```

4
5 int Joseph(int totalPeople, int nextNumber, int startIndex, int lastIdx
6 )
7 { // All based on 0_Index , the Answer is the last `lastIdx` to leave
8   int now = (nextNumber - 1) % lastIdx + (startIndex - nextNumber);
9   for(int i = lastIdx + 1; i <= totalPeople; i++)
10     now = (now + nextNumber) % i;
11   return now;
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)
20           +1, Joseph(n, m, m, 1)+1);
21   }
22   return 0;
23 }

```

## 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 <stdio>
2 #include <stdlib>
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;

```

```

11     int tmp = getroot(f[x]);
12     g[x] += g[f[x]]; // update the value
13     return f[x] = tmp;
14 }
15
16 void merge(int x, int y) // merge x's set and y's set
17 { // 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
18     int fy = getroot(y);
19     g[x] += g[y] + abs(x - y) % 1000; // update the value
20     f[x] = fy;
21 }
22
23 int main()
24 {
25     scanf("%d", &n);
26     for(int i = 1; i <= n; i++) f[i] = i;
27     char op; int x, y;
28     while(scanf("%c", &op) == 1 && op != '0')
29     {
30         if(op == 'I')
31         {
32             scanf("%d%d", &x, &y);
33             if(getroot(x) == getroot(y)) continue;
34             merge(x, y);
35         }
36         else{
37             scanf("%d", &x);
38             getroot(x); // !!! update the value of x before output
39             printf("%d\n", g[x]);
40         }
41         //for(int i = 1; i <= n; i++) printf("%d ", f[i]); puts("");
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     {
12         std::swap(a[i], a[i/2]);
13         i /= 2;
14     }

```

```

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

```

### 3.3 Leftist Tree

```

1 //很多时候需要配合并查集一起使用
2 int getroot(int x){return f[x]==x ? x : f[x]=getroot(f[x]);}
3
4 //把x和y合并在一起，其实就是把y插入x
5 int merge(int x,int y)//返回合并后子树的根
6 {
7     if(!x || !y) return x|y;
8     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
9     R[x]=merge(R[x],y);//始终往右子树合并

```

```

10     f[R[x]] = x; //更新并查集
11     if(D[R[x]] > D[L[x]]) swap(L[x], R[x]); //保持左偏树性质
12     D[x] = D[R[x]] + 1;
13     若还有其他维护信息也需要更新;
14     return x; //返回根
15 }
16
17 int del(int x)
18 {
19     int t = merge(L[x], R[x]);
20     f[L[x]] = L[x]; f[R[x]] = R[x]; //更新并查集
21     L[x] = R[x] = D[x] = 0;
22     return t;
23 }

```

### 3.4 Partition Tree

```

1 struct Parti{int val, left;} val[30][N];
2 void build_tree(int d, int l, int r)
3 {
4     if(l == r) return;
5     int m = (l + r) >> 1, same = m - l + 1;
6     int lcnt = l, rcnt = m + 1;
7     for(int i = l; i <= r; i++)
8         if(val[d][i].val < sorted[m]) same--;
9     for(int i = l; i <= r; i++)
10    {
11        int flag = 0;
12        if((val[d][i].val < sorted[m]) || (val[d][i].val == sorted[m] &&
13            same))
14        {
15            flag = 1;
16            val[d + 1][lcnt++] = val[d][i];
17            if(val[d][i].val == sorted[m]) same--;
18        }
19        else val[d][rcnt++] = val[d][i];
20        val[d][i].left = val[d][i - 1].left + flag;
21    }
22    build_tree(d + 1, l, m);
23    build_tree(d + 1, m + 1, r);
24 }
25 int query(int d, int l, int r, int x, int y, int k)
26 {
27     if(l == r) return val[d][l].val;
28     int m = (l + r) >> 1;
29     int lx = val[d][x - 1].left - val[d][l - 1].left; //[l,x-1] to left
30     int ly = val[d][y].left - val[d][x - 1].left; //[x,y] to left
31     int rx = (x - 1 - l + 1) - lx; //[l,x-1] to right
32     int ry = (y - x + 1) - ly; //[x,y] to right
33     if(ly >= k) return query(d+1, l, m, l-1+lx+1, l-1+lx+ly, k);
34     else return query(d+1, m+1, r, m+1-1+rx+1, m+1-1+rx+ry, k-ly);
35 }

```

### 3.5 Treap

#### 3.5.1 @ Array

```

1  #include <cstdio>
2  #include <cstdlib>
3  #include <ctime>
4
5  const int N = 100000 + 10;
6
7  int m, Limit;
8  int L[N], R[N], S[N], fix[N], key[N];
9  int root, total, leave;
10
11 void rotate_left(int &p)
12 {
13     int tmp = R[p];
14     R[p] = L[tmp];
15     int zsize = S[L[tmp]];
16     S[p] = S[p] - S[tmp] + zsize;
17     L[tmp] = p;
18     S[tmp] = S[tmp] - zsize + S[p];
19     p = tmp;
20 }
21 void rotate_right(int &p)
22 {
23     int tmp = L[p];
24     L[p] = R[tmp];
25     int zsize = S[R[tmp]];
26     S[p] = S[p] - S[tmp] + zsize;
27     R[tmp] = p;
28     S[tmp] = S[tmp] - zsize + S[p];
29     p = tmp;
30 }
31
32 void insert(int &p, int x)
33 {
34     if(!p)
35     {
36         p = ++total;
37         L[p] = R[p] = 0;
38         S[p] = 1;
39         fix[p] = rand();
40         key[p] = x;
41         return;
42     }
43     S[p]++;
44     if(x < key[p])
45     {
46         insert(L[p], x);
47         if(fix[L[p]] > fix[p]) rotate_right(p);
48     }
49     else {

```

```

50     insert(R[p], x);
51     if(fix[R[p]] > fix[p]) rotate_left(p);
52 }
53 }
54
55 void remove(int &p, int limit)
56 {
57     if(!p) return;
58     if(key[p] < limit)
59     {
60         leave += S[L[p]] + 1;
61         p = R[p];
62         remove(p, limit);
63     }
64     else{
65         remove(L[p], limit);
66         S[p] = S[L[p]] + S[R[p]] + 1;
67     }
68 }
69
70 int kth(int &p, int k)
71 {
72     if(k <= S[L[p]]) return kth(L[p], k);
73     else if(k == S[L[p]] + 1) return key[p];
74     else return kth(R[p], k - S[L[p]] - 1);
75 }
76
77 int main()
78 {
79     srand(time(0));
80     scanf("%d%d", &m, &Limit);
81     int delta = 0;
82     while(m--)
83     {
84         char op; int x;
85         scanf("%c%d", &op, &x);
86         if(op == 'I')
87         {
88             if(x < Limit) continue;
89             insert(root, x - delta);
90         }
91         else if(op == 'A') delta += x;
92         else if(op == 'S')
93         {
94             delta -= x;
95             remove(root, Limit - delta);
96         }
97         else {
98             x = S[root] - x + 1;
99             if(x <= 0) puts("-1");
100             else printf("%d\n", kth(root, x) + delta);
101         }
102     }

```

```

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

```

### 3.5.2 @ Pointer

```

1  #include <stdio>
2  #include <stdlib>
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;

```

```

100         remove(root, Limit - delta);
101     }
102     else {
103         x = root -> size - x + 1;
104         if(x <= 0) puts("-1");
105         else printf("%d\n", kth(root, x) + delta);
106     }
107 }
108 printf("%d\n", leave);
109 return 0;
110 }

```

### 3.6 Size Balanced Tree

```

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

```

```

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

```

### 3.7 三维偏序 - CDQ 分治

```

1 #include <cstdio>
2 #include <cstring>
3 #include <algorithm>
4 #define lowbit(_X) ((_X)&(-(_X)))
5 // SPOJ LIS2
6 const int N = 100000 + 10;
7

```

```

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));

```

```

5   while(!Q.empty())
6   {
7       int x = Q.top().second; Q.pop();
8       if(done[x]) continue;
9       done[x] = 1;
10      for(Link p = head[x]; p; p = p->next)
11          if(dist[p->y] > dist[x] + p->z)
12          {
13              dist[p->y] = dist[x] + p->z;
14              Q.push(make_pair(-dist[p->y], p->y));
15          }
16      }
17  }

```

#### 4.1.2 Spfa

```

1  void spfa()
2  {
3      memset(inQ, 0, sizeof(inQ));
4      memset(dist, 0x3f, sizeof(dist));
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)
10             if(dist[p->y] > dist[x] + p->z)
11             {
12                 dist[p->y] = dist[x] + p->z;
13                 if(!inQ[p->y])
14                 {
15                     Q.push(p->y);
16                     inQ[p->y] = 1;
17                 }
18             }
19      }
20  }

```

#### 4.1.3 Floyd

```

1  void floyd()
2  {
3      for(int k = 1; k <= n; k++) // 这里可以看作是一个加边的过程
4          for(int i = 1; i <= n; i++)
5              for(int j = 1; j <= n; j++)
6                  map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
7  }
8
9  // 最小环
10 void MinCircle()
11 {
12     cap[] = map[];

```

```

13     int circle = 0x3f3f3f3f;
14     for(int k = 1; k <= n; k++)
15     {
16         for(int i = 1; i < k; i++)
17             for(int j = i+1; j < k; j++)
18                 circle = min(circle, map[i][j] + cap[j][k]+cap[k][i]);
19         for(int i = 1; i <= n; i++)
20             for(int j = 1; j <= n; j++)
21                 map[i][j] = min(map[i][j], map[i][k] + map[k][j]);
22     }
23     return circle == 0x3f3f3f3f ? -1 : circle;
24 }
25
26 // floyd判圈法 ( 大白书 p44 )
27 void Circle()
28 {
29     int ans = k;
30     int k1 = k, k2 = k;
31     do{
32         k1 = next(k1);
33         k2 = next(k2); ans = max(ans, k2);
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

```

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

```

1 int kruskal()
2 {
3     sort(edge, edge+Cnt, cmp);
4     int res = 0;
5     for(int i = 0; i < Cnt; i++)
6     {
7         if(getroot(edge[i].x) == getroot(edge[i].y)) continue;
8         f[getroot(edge[i].x)] = getroot(edge[i].y);
9         res += edge[i].z;
10    }
11    return res;
12 }

```

### 4.3 Tarjan - Strong Union

```

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

```



```

38         belong[u] = number;
39     }
40     number++;
41 }
42 }

```

## 4.4 LCA

### 4.4.1 @ Tarjan

```

1 // poj 1330 (changed something)
2 // LCA tarjan
3 #include <cstdio>
4 #include <cstring>
5
6 const int N = 10000 + 10;
7
8 int n;
9 struct Link{int y, idx; Link *next;}*head[N], *ask[N];
10 int tx, ty;
11 bool in[N], vis[N];
12 int f[N];
13 int ans[N]; // Query Answer
14
15 void inLink(int x, int y)
16 {
17     Link *p = new Link;
18     p -> y = y;
19     p -> next = head[x];
20     head[x] = p;
21 }
22 void inAsk(int x, int y, int idx)
23 {
24     Link *p = new Link;
25     p -> y = y;
26     p -> idx = idx;
27     p -> next = ask[x];
28     ask[x] = p;
29 }
30
31 int getroot(int x)
32 {
33     return f[x] == x ? x : f[x] = getroot(f[x]);
34 }
35
36 void LCA(int x)
37 {
38     vis[x] = 1;
39     f[x] = x;
40     for(Link *p = ask[x]; p; p = p -> next)
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 {
52     int T; scanf("%d", &T);
53     while(T--)
54     {
55         memset(head, 0, sizeof(head));
56         memset(ask, 0, sizeof(ask));
57         memset(in, 0, sizeof(in));
58         memset(vis, 0, sizeof(vis));
59         scanf("%d", &n);
60         for(int i = 1; i <= n; i++) f[i] = i;
61         for(int i = 1; i < n; i++)
62         {
63             int x, y;
64             scanf("%d%d", &x, &y);
65             inLink(x, y);
66             in[y] = 1;
67         }
68         int q = 1; // the number of query
69         for(int i = 1; i <= q; i++)
70         {
71             int x, y; scanf("%d%d", &x, &y);
72             inAsk(x, y, i); inAsk(y, x, i);
73         }
74         int root = -1;
75         for(int i = 1; i <= n; i++)
76             if(!in[i]) {root = i; break;}
77         LCA(root);
78         for(int i = 1; i <= q; i++)
79             printf("%d\n", ans[i]);
80     }
81     return 0;
82 }

```

#### 4.4.2 @ Doubling Algorithm

```

1 #include <cstdio>
2 #include <cstring>
3 #include <algorithm>
4 // POJ 1330 LCA_Doubling Algorithm
5 const int N = 10000 + 10;
6
7 const int UPDepth = 14;
8 int n;
9 struct Link{

```

```

10     int y;
11     Link *next;
12 }*head[N];
13 bool in[N];
14 int ancient[N][UPDepth+1];
15 int deep[N];
16
17 void inLink(int x, int y)
18 {
19     Link *p = new Link;
20     p -> y = y;
21     p -> next = head[x];
22     head[x] = p;
23 }
24
25 void dfs(int x, int depth, int father)
26 {
27     deep[x] = depth;
28     ancient[x][0] = father;
29     for(Link *p = head[x]; p; p = p -> next)
30         dfs(p -> y, depth + 1, x);
31 }
32
33 void getLCA()
34 {
35     for(int i = 1; i <= n; i++)
36         if(!in[i]) dfs(i, 1, 0);
37     for(int j = 1; j <= UPDepth; j++)
38         for(int i = 1; i <= n; i++)
39             ancient[i][j] = ancient[ancient[i][j-1]][j-1];
40 }
41
42 int LCA(int x, int y)
43 {
44     if(deep[x] > deep[y]) std::swap(x, y); // deep[x] <= deep[y]
45     for(int j = UPDepth; j >= 0; j--)
46         if(deep[x] <= deep[ancient[y][j]]) y = ancient[y][j];
47     if(x == y) return x;
48     for(int j = UPDepth; j >= 0; j--)
49         if(ancient[x][j] != ancient[y][j])
50         {
51             x = ancient[x][j];
52             y = ancient[y][j];
53         }
54     return ancient[y][0];
55 }
56
57 int main()
58 {
59     int T; scanf("%d", &T);
60     while(T--)
61     {
62         memset(head, 0, sizeof(head));

```

```

63     memset(ancient, 0, sizeof(ancient));
64     memset(in, 0, sizeof(in));
65     memset(deep, 0, sizeof(deep));
66     scanf("%d", &n);
67     for(int i = 1; i < n; i++)
68     {
69         int x, y; scanf("%d%d", &x, &y);
70         inLink(x, y); in[y] = 1;
71     }
72     getLCA();
73     int x, y; scanf("%d%d", &x, &y);
74     printf("%d\n", LCA(x, y));
75 }
76 return 0;
77 }

```

## 4.5 Bipartite Graph

### 4.5.1 Maximal Matching - The Hungarian algorithm

```

1  int timeStamp = 0;
2  int n, m, g[N][N];
3  int vis[N], pre[N];
4
5  bool search(int x)
6  {
7      for(int i = 1; i <= m; i++)
8          if(g[x][i] && vis[i] != timeStamp)
9              {
10                 vis[i] = timeStamp;
11                 if(pre[i] == -1 || search(pre[i]))
12                     {
13                         pre[i] = x;
14                         return 1;
15                     }
16             }
17      return 0;
18  }
19
20  int maxMatch()
21  {
22      int res = 0;
23      memset(pre, -1, sizeof(pre));
24      for(int i = 1; i <= n; i++)
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>
2  #include <algorithm>
3
4  const int N = 200 + 10;
5
6  int n, m, g[N][N];
7  int v[N], h[N];
8  int S, T;
9
10 int sap(int x, int flow)
11 {
12     if(x == n) return flow;
13     int res = 0;
14     for(int i = S; i <= T; i++)
15         if(g[x][i] && h[x] == h[i] + 1)
16         {
17             int t = sap(i, std::min(g[x][i], flow - res));
18             res += t; g[x][i] -= t; g[i][x] += t;
19             if(res == flow) return res;
20             if(h[S] >= T) return res;
21         }
22     //if(h[S] >= T) return res;
23     if((--v[h[x]]) == 0) h[S] = T;
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
31     for(int i = 1; i <= m; i++)
32     {
33         int x, y, z;
34         scanf("%d%d%d", &x, &y, &z);
35         g[x][y] += z;
36     }
37     v[0] = T; S = 1; T = n;
38     int maxflow = 0;
39     while(h[S] < T) maxflow += sap(1, 0x3f3f3f3f);
40     printf("%d\n", maxflow);
41     return 0;
42 }

```

### 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 // 初始化 μlist[0~n-1]
2 // 初始化 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]))
10         <=0)
11         return true;
12     else return false;
13 }
14 void Graham(int n)
15 {
16     Point p0;
17     int k=0;
18     p0=list[0];
19     for (int i=1;i<n;++i)
20     {
21         if ((p0.y>list[i].y)||((p0.y==list[i].y&&p0.x>list[i].x))
22         {
23             p0=list[i];
24             k=i;
25         }
26     }
27     swap(list[k],list[0]);
28     sort(list+1,list+n,_cmp);
29     if (n==1)
30     {
31         top=1;
32         stack[0]=0;
33         return;
34     }
35     if (n==2)
36     {
37         top=2;
38         stack[0]=0;
39         stack[1]=1;
40         return;
41     }

```

```

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 //////////////////////////////////////
26 struct Point{
27     double x, y;
28     Point (){}
29     Point (double _x,double _y):x(_x),y(_y){}
30     void init(double a, double b) { x = a; y = b; }
31
32     // basic calc
33     bool operator == (const Point &b) const
34     {
35         return !fuhao(x - b.x) && !fuhao(y - b.y);
36     }
37     Point operator + (const Point &b) const
38     {

```



```

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) //  $\alpha \in [0, +\infty)$  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     //  $A \neq \text{Vector}(0.0, 0.0)$ ;
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     //////////////////////////////////////
89 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     &&
124     fuhao((l1.s-l2.e)^(l2.s-l2.e)) * fuhao((l1.e-l2.e)^(l2.s-l2.e)) <=0;
125 }
126 //判断直线与线段相交
127 bool Seg_inter_line(Line l1,Line l2)//l1为直线 l2为线段
128 {
129     return fuhao((l2.s-l1.e)^(l1.s-l1.e))*fuhao((l2.e-l1.e)^(l1.s-l1.e))
130     <=0;
131 }
132 //点到直线距离
133 //返回点到直线最近的点
134 Point PointToLine(Point P,Line L)
135 {
136     Point ans;
137     double t=((P-L.s)*(L.e-L.s))/((L.e-L.s)*(L.e-L.s));
138     ans.x=L.s.x+(L.e.x-L.s.x)*t;
139     ans.y=L.s.y+(L.e.y-L.s.y)*t;
140     return ans;

```

```

139 }
140 //点到线段距离
141 //返回点到线段最近的点
142 Point NearestPointToLineSeg(Point P,Line L)
143 {
144     Point ans;
145     double t = ((P-L.s)*(L.e-L.s)) / ((L.e-L.s)*(L.e-L.s));
146     if (t>=0&&t<=1)
147     {
148         ans.x = L.s.x + (L.e.x-L.s.x)*t;
149         ans.y = L.s.y + (L.e.y-L.s.y)*t;
150     }
151     else {
152         if (Dist(P,L.s)<Dist(P,L.e))
153             ans = L.s;
154         else ans = L.e;
155     }
156     return ans;
157 }
158 //多边形面积
159 double CalcArea(Point p[],int n)
160 {
161     double ans=0;
162     for (int i=0;i<n;++i)
163         ans+=(p[i]^p[(i+1)%n])/2;
164     return fabs(ans);
165 }
166 //判断点在线段上
167 bool OnSeg(Point P,Line L)
168 {
169     return
170         fuhao((L.s-P)^(L.e-P))==0 &&
171         fuhao((P.x-L.s.x)*(P.x-L.e.x))<=0 &&
172         fuhao((P.y-L.s.y)*(P.y-L.e.y))<=0;
173 }
174 //三点求圆心坐标
175 Point waixin(Point a,Point b,Point c)
176 {
177     double a1=b.x-a.x,b1=b.y-a.y,c1=(a1*a1+b1*b1)/2;
178     double a2=c.x-a.x,b2=c.y-a.y,c2=(a2*a2+b2*b2)/2;
179     double d=a1*b2-a2*b1;
180     return Point(a.x+(c1*b2-c2*b1)/d,a.y+(a1*c2-a2*c1)/d);
181 }
182
183
184 ////////////////////////////////////////          Graham
185 ////////////////////////////////////////
186 //求凸包 点list[0~n-1]
187 //凸包结果Stack[0~top-1]
188 const int Maxn = 100;////////////////////////////////////here!!
189 Point list[Maxn];          //////////////////////////////////////?!?!?!?! 补全Maxn
190     !?!?!?!?!?!?!?!?!?!?!
191 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]))
195             <=0)
196         return true;
197     else return false;
198 }
199 void Graham(int n)
200 {
201     Point p0;
202     int k=0;
203     p0=list[0];
204     for (int i=1;i<n;++i)
205     {
206         if ((p0.y>list[i].y)||((p0.y==list[i].y&&p0.x>list[i].x))
207         {
208             p0=list[i];
209             k=i;
210         }
211     }
212     swap(list[k],list[0]);
213     sort(list+1,list+n,_cmp);
214     if (n==1)
215     {
216         top=1;
217         Stack[0]=0;
218         return;
219     }
220     if (n==2)
221     {
222         top=2;
223         Stack[0]=0;
224         Stack[1]=1;
225         return;
226     }
227     Stack[0]=0;
228     Stack[1]=1;
229     top=2;
230     for (int i=2;i<n;++i)
231     {
232         while (top>1 && fuhao((list[Stack[top-1]]-list[Stack[top-2]])^(
233             list[i]-list[Stack[top-2]]))<=0)
234             top--;
235         Stack[top++]=i;
236     }
237 }
238 ////////////////////////////////////// E - N - D
239 //////////////////////////////////////

```

```

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

```

```

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

```

## 6.2 KMP

```

1 #include <cstdio>
2 #include <cstring>
3 // POJ 3461 : Count the number of t occurrences in s
4 char s[1000000+10], t[1000000+10];
5 int next[1000000+10];
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    {
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; }

```



```

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

```

## 6.4 Aho-Corasick Automaton

```

1  #include <cstdio>
2  #include <cstring>
3  #include <queue>
4  using std::queue;
5  // HDU 2222 查询 n 个模式串中有几个在原串 str 中出现了
6  struct ACG{
7      int count;
8      ACG *fail, *next[26];
9      ACG()
10     {
11         fail = 0;
12         count = 0;
13         for(int i = 0; i < 26; i++) next[i] = 0;
14     }
15 }*root;
16 queue<ACG*> Q;
17
18 void insert(char *str, ACG *p)
19 {

```

```

20     int len = strlen(str);
21     for(int i = 0; i < len; i++)
22     {
23         int x = str[i] - 'a';
24         if(!p -> next[x]) p -> next[x] = new ACG;
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);
34     while(!Q.empty())
35     {
36         ACG *p = Q.front(); Q.pop();
37         for(int i = 0; i < 26; i++)
38         {
39             if(p -> next[i])
40             {
41                 if(p == root) p -> next[i] -> fail = root;
42                 else{
43                     ACG *temp = p -> fail;
44                     while(temp)
45                     {
46                         if(temp -> next[i])
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 {
63     int len = strlen(str), res = 0;
64     for(int i = 0; i < len; i++)
65     {
66         int x = str[i] - 'a';
67         while(!p -> next[x] && p != root) p = p -> fail;
68         p = p -> next[x];
69         if(!p) p = root;
70         ACG *temp = p;
71         while(temp != root && temp -> count != -1)
72         {

```

```

73         res += temp -> count;
74         temp -> count = -1;
75         temp = temp -> fail;
76     }
77 }
78     return res;
79 }
80
81 int n;
82 char tmp[1000000+10];
83
84 int main()
85 {
86     int T; scanf("%d", &T);
87     while(T--)
88     {
89         root = new ACG;
90         scanf("%d", &n);
91         for(int i = 1; i <= n; i++)
92         {
93             scanf("%s", tmp);
94             insert(tmp, root);
95         }
96         build_acg();
97         scanf("%s", tmp);
98         printf("%d\n", query(tmp, root));
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;
11     BigInt() { memset(c, 0, sizeof(c)); len = 1; sign = 0; }
12     void Zero()
13     {
14         while(len > 1 && c[len] == 0) len--;
15         if(len == 1 && c[len] == 0) sign = 0;
16     }
17     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 // 一定记住最后 cin.close();
4
5 // 一般直接用 System.out.println();
6 PrintWriter cout = new PrintWriter(System.out);
7 PrintWriter cout = new PrintWriter(new BufferedOutputStream(System.out)
8     ); // 一定记住最后 cout.close();
9
10 int n = cin.nextInt();
11 String s = cin.next();
12 double m = cin.nextDouble();
13 String line = cin.nextLine(); // 读一整行
14 BigInteger c = cin.nextBigInteger();
15 while(cin.hasNext()) {};
16
17 // PrintWriter 用 cout.println(...);
18 System.out.println(n + "—>" + s + "—>" + m);
19
20 // 使用 format 控制格式, 与 C/C++ 一样, double 用 %f,
21 System.out.format("%03d", c).println();
22 System.out.format("%.3f", c).println();

```



```

22
23 //变量声明
24 int a, b[] = new int[100];
25 double a, b[] = new double[100];
26 int a[][] = new int[100][100];
27 String ...
28 BigInteger/BigDecimal ...

```

### 7.5.3 BigInteger

```

1 BigInteger a = BigInteger.valueOf(100);
2 BigInteger b = BigInteger.valueOf(50);
3 BigInteger ONE = BigInteger.ONE;
4 BigInteger TWO = BigInteger.valueOf(2);
5 a = a.add(ONE).subtract(b);
6 a = a.multiply(TWO).divide(TWO);
7 a = a.mod(TWO);
8 a.compareTo(ONE); // 大于1, 小于-1, 等于0
9 //BigDecimal 为高精小数

```

### 7.5.4 String

```

1 String s = "abcdefg"; // 注意0下标!
2 char c = s.charAt(2); // 相当于 `char c = s[2]` (C++)(c = 'c')
3 char ch[];
4 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;
6 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.*;
3 import java.math.*;
4 // Binary, Octal, Decimal(Integer/BigInteger), Hexadecimal
5 public class Main{
6     public static void main(String args[])
7     {
8         //Decimal(123) to Others
9         String a1 = Integer.toBinaryString(123);
10        String a2 = Integer.toOctalString(123);
11        String a3 = Integer.toHexString(123);
12        //Others to Decimal(123)
13        int b1 = Integer.valueOf("1111011", 2);
14        int b2 = Integer.valueOf("173", 8);
15        int b3 = Integer.valueOf("7b", 16);
16        // Others to BigInteger(Decimal(123))
17        BigInteger c1 = new BigInteger("1111011", 2);

```

```

18         BigInteger c2 = new BigInteger("173", 8);
19         BigInteger c3 = new BigInteger("7B", 16);
20     }
21 }

```

### 7.5.6 function

```

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

```

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

```

```

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

```

### 7.6.2 @Windows

```

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

```

## 7.7 Vimrc Config For Linux

```

1 filetype on
2 filetype indent on
3 set nobackup
4 set nu
5 set st=4
6 set ts=4
7 set sw=4
8
9 map <F7> <Esc>:w<CR>:!javac %:r.java<CR>:!java %:r<CR>
10 imap <F7> <Esc>:w<CR>:!javac %:r.java<CR>:!java %:r<CR>
11 map <F8> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!gdb %:r<CR>
12 imap <F8> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!gdb %:r<CR>
13 map <F9> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!./%:r<CR>
14 imap <F9> <Esc>:w<CR>:!g++ -g %:r.cpp -o %:r<CR>:!./%:r<CR>
15 map <c-a> <Esc>gg"+yG
16 imap␣<c-a>␣<Esc>gg"+yG

```