ACM-ICPC Template

GuessEver

Last Modify at March 8, 2015

Contents

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 数位 dp

```
int predoing(LL a, int *num)
{
    int le = 0;
    while(a)
    {
        num[++le] = a % 10;
        a /= 10;
    }
    return le;
}
int calc(int pos, int d, int u, int last)
{
    if(pos == 0) return 1;
```

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

2 Math

2.1 GCD && LCM

2.1.1 GCD - Greatest Common Divisor

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

2.1.2 LCM - Least Common Multiple

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

2.1.3 E_GCD - Extended Greatest Common Divisor

```
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

```
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++)
        {
            h[i * pri[j]] = true;
            if(i % pri[j] == 0) break;
        }
    }
}
```

2.2.2 Prime Factor

```
void factor()
{
    make_prime_list();
    for(int j = 0; j < Cnt && pri[j]*pri[j] <= n; j++)
    {
        if(n % pri[j] == 0)
        {
            printf("%d␣", pri[j]);
            while(n % pri[j] == 0) n /= pri[j];
        }
    }
    if(n!=1) printf("%d",n);
}
```

2.3 Fast Power

```
//x^y  % mod
int mul(int x, LL y, int mod) // 递归
{
    if(y == 1) return x;
    if(y & 1) return (mul((x * (LL)x) % mod, y / 2, mod) * (LL)x)%mod;
    else return mul((x * (LL)x) % mod, y / 2, mod) % mod;
}
int mul(int x, int y, int mod) // 非递归
{
    int s = 1;
    int ss = x;
    while(y)
    {
        if(y & 1) s = s * ss;
        y /= 2;
        ss *= ss;
    }
    return s;
}
```

2.4 约瑟夫环、丢手绢问题

```
#include <cstdio>

int n, m, k;

int solve(int totalPeople, int nextNumber, int startIndex)
{
    int now = 0;
    for(int i = 2; i < totalPeople; i++)
        now = (now + nextNumber) % i;
    now = (now + startIndex) % n;
    return now + 1; // 1_Index
}

int main()
{
    while(scanf("%d%d%d", &n, &k, &m) == 3 && (n || m || k))
        printf("%d\n", solve(n, k, m));
    return 0;
}
```

3 Datastructure

3.1 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;
}
```

3.2 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);
}
```

3.3 Treap

3.3.1 @ 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;

```

```
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 }
```

3.3.2 @ 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;
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 }
```

3.4 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)
```

```
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  }
```

4 Graph

4.1 Shortest path

4.1.1 Dijkstra

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

4.1.2 Spfa

```
void spfa()
{
    memset(dist, 0x3f, sizeof(dist));
    Q.push(S);//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

```
void prime()
{
    memset(dist, 0, sizeof(dist));
    int res = 0;
    while(!Q.empty())
    {
        int x = Q.top().second;
        if(done[x]) {Q.pop(); continue;}
        res += Q.top().first;
        Q.pop();
        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));
            }
    }
}
```

4.2.2 Kruskal

```
void prime()
{
    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;
        merge(edge[i].x, edge[i].y);
        res += edge[i].z;
    }
}
```

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>

```

```
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

还不会...

4.5 Bipartite Graph

4.5.1 Maximal Matching - The Hungarian algorithm

```
1  int ttt = 0; // 全局时间戳变量
2
3  bool search(int x)
4  {
5      for(int i = 1; i <= m; i++)
6          if(map[x][i] && vis[i] != ttt)
7          {
8              vis[i] = ttt;
9              if(pre[i] == -1 || search(pre[i]))
10             {
11                 pre[i] = x;
12                 return 1;
13             }
14         }
15     return 0;
16 }
17
18 int match()
19 {
```

```
20      int res = 0;
21      for(int i = 1; i <= n; i++)
22      {
23          ++ttt; // 这里不用memset节省时间
24          res += search(i);
25      }
26      return res;
27  }
```

4.5.2 Optimal Matching - KM

不会... 用费用流解决

4.6 Network Flow

4.6.1 Maximum Flow - isap

```
1   //   h[x] :   点 x 在第 h[x] 层
2   //   v[k] :   第 k 层有 v[k] 个点
3   int sap(int x, int flow)
4   {
5       if(x == n) return flow;
6       int res = 0;
7       for(int i = S; i <= T; i++)
8           if(g[x][i] && h[x] == h[i] + 1)
9           {
10              int t = sap(i, min(g[x][i], flow - res));
11              res += t; g[x][i] -= t; g[i][x] += t;
12              if(res == flow) return res;
13              if(h[S] >= T) return res;
14          }
15      //if(h[S] >= T) return res;
16      if((--v[h[x]]) == 0) h[S] = T;
17      ++v[++h[x]];
18      return res;
19  }
20  int main()
21  {
22      v[0] = T;
23      int maxflow = 0;
24      while(h[S] < T) maxflow += sap(1, inf);
25      reutrn 0;
26  }
```

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  //□°◆½◆◆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;
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 //求凸包 点list[0~n-1]
186 //凸包结果Stack[0~top-1]
187 const int Maxn = 100;////////////////here!!
188 Point list[Maxn]; //////////?!?!?!?!补全Maxn
    !?!?!?!?!?!?!?!?!?!?!
189 int Stack[Maxn],top;
190 bool _cmp (Point p1,Point p2)
191 {
192     double tmp=(p1-list[0])^(p2-list[0]);
193     if (fuhao(tmp)>0) return true;
194     else if (fuhao(tmp)==0&&fuhao(Dist(p1,list[0])-Dist(p2,list[0]))
        <=0)
195         return true;
196     else    return false;
```

```
}
void Graham(int n)
{
    Point p0;
    int k=0;
    p0=list[0];
    for (int i=1;i<n;++i)
    {
        if ((p0.y>list[i].y)||(p0.y==list[i].y&&p0.x>list[i].x))
        {
            p0=list[i];
            k=i;
        }
    }
    swap(list[k],list[0]);
    sort(list+1,list+n,_cmp);
    if (n==1)
    {
        top=1;
        Stack[0]=0;
        return;
    }
    if (n==2)
    {
        top=2;
        Stack[0]=0;
        Stack[1]=1;
        return;
    }
    Stack[0]=0;
    Stack[1]=1;
    top=2;
    for (int i=2;i<n;++i)
    {
        while (top>1 && fuhao((list[Stack[top-1]]-list[Stack[top-2]])^(
           list[i]-list[Stack[top-2]]))<=0)
            top--;
        Stack[top++]=i;
    }
}
//////////////////////////      E - N - D
   //////////////////////////


//////////////////////////          Area
   //////////////////////////
double PolygonArea(Point *pp, int nn) // pp[0, n-1]
{
    double ans_area = 0.0;
    for(int i = 1; i < nn-1; i++)
    {
        ans_area += (pp[i] - pp[0]) ^ (pp[i+1] - pp[0]);
    }
```

```
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     }
```

```
    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];

void getNext(char *t, int len, int *Next)
{
    memset(Next, 0, sizeof(Next)); Next[0] = -1;
    for(int j = 0, k = -1; j < len; )
    {
        if(k == -1 || t[j] == t[k]) Next[++j] = ++k;
        else k = Next[k];
    }
}
```

```
int kmp(char *s, int lens, char *t, int lent)
{
    int res = 0;
    getNext(t, lent, next);
    for(int i = 0, j = 0; i < lens; )
    {
        if(j == -1 || s[i] == t[j]) { i++; j++; }
        else j = next[j];
        if(j == lent) res++; // Bingo! [pos = j - lent]
    }
    return res;
}

int main()
{
    int T; scanf("%d", &T);
    while(T--)
    {
        scanf("%s%s", t, s);
        printf("%d\n", kmp(s, strlen(s), t, strlen(t)));
    }
    return 0;
}
```

6.3 Suffix Array

```
#include <cstdio>
#include <algorithm>
#include <map>
using std::map;
// POJ 3261 找重复了K次的最长子串
const int N = 20000 + 10;
/*
    sa[rank[i]] = i
    sa[i] = j       : rank i is s[j, n)
    rank[j] = i     : s[j, n) is rank i
    height[i] = j   : the longest common prefix of string rank _i and
       _i-1
*/

int sa[N], rank[N];
int c[N], tmp[N];
int height[N];

bool cmp(int *r, int a, int b, int l)
{
    return r[a] == r[b] && r[a+l] == r[b+l];
}

void DA(int *s, int n, int m) // s[0...n-1] E [1, m)
{
    int i, j, p, *x = rank, *y = tmp;
```

```
    for(i = 0; i < m; i++) c[i] = 0;
    for(i = 0; i < n; i++) c[x[i] = s[i]]++;
    for(i = 1; i < m; i++) c[i] += c[i-1];
    for(i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
    for(j = 1, p = 0; p < n; j *= 2, m = p)
    {
        for(p = 0, i = n-j; i < n; i++) y[p++] = i;
        for(i = 0; i < n; i++) if(sa[i] >= j) y[p++] = sa[i] - j;
        for(i = 0; i < m; i++) c[i] = 0;
        for(i = 0; i < n; i++) c[x[y[i]]]++;
        for(i = 1; i < m; i++) c[i] += c[i-1];
        for(i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
        for(std::swap(x, y), p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
            x[sa[i]] = cmp(y, sa[i], sa[i-1], j) ? p - 1 : p++;
    }
    for(i = 0; i < n; i++) rank[sa[i]] = i;

    int k = 0; height[0] = 0;
    for(i = 0; i < n; height[rank[i++]] = k) if(rank[i])
        for(k ? k-- : 0, j = sa[rank[i]-1]; s[j+k] == s[i+k]; k++);
}

int n, K, a[N];
map<int, int> hash;

bool check(int len)
{
    int cnt = 0;
    for(int i = 1; i < n; i++)
    {
        if(height[i] >= len) cnt++;
        else cnt = 0;
        if(cnt >= K - 1) return 1;
    }
    return 0;
}

int Solve()
{
    int low = 0, high = n, ans = 0;
    while(low <= high)
    {
        int mid = low + (high - low) / 2;
        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 ++;
}
```

```
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;
```

```
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)
    {
        int k = 1, L = strlen(s);
        for(int i = L-1; i >= 0; i--)
        {
            c[len] += (s[i]-'0') * k;
            k *= 10;
            if(k == base)
            {
                k = 1;
```

```
                len++;
            }
        }
    }
    void Read()
    {
        char s[5000] = {0};
        scanf("%s", s);
        writein(s);
    }
    void Print()
    {
        if(sign) printf("-");
        printf("%d", c[len]);
        for(int i = len-1; i >= 1; i--) printf("%04d", c[i]);
        printf("\n");
    }
    BigInt operator = (int a)
    {
        char s[100] = {0};
        sprintf(s, "%d", a);
        writein(s);
        return *this;
    }
    bool operator > (const BigInt &b)
    {
        if(len != b.len) return len > b.len;
        for(int i = len; i >= 1; i--)
        {
            if(c[i] != b.c[i]) return c[i] > b.c[i];
        }
        return 0;
    }
    bool operator < (const BigInt &b)
    {
        if(len != b.len) return len < b.len;
        for(int i = len; i >= 1; i--)
        {
            if(c[i] != b.c[i]) return c[i] < b.c[i];
        }
        return 0;
    }
    bool operator == (const BigInt &b)
    {
        if(len != b.len) return 0;
        for(int i = 1; i <= len; i++)
            if(c[i] != b.c[i]) return 0;
        return 1;
    }
    bool operator == (const int &a)
    {
        BigInt b; b = a;
        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 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
```

7.3 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 ）
```

7.4 Batch test

7.4.1 @Linux

```
1 mkdata=mk
2 filea=a
3 fileb=b
```

```
4
5  g++ $mkdata.cpp -o $mkdata
6  g++ $filea.cpp -o $filea
7  g++ $fileb.cpp -o $fileb
8  cas=0
9  while true; do
10     ./$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
```

7.4.2 @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
```

7.5 Vimrc Config For Linux

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