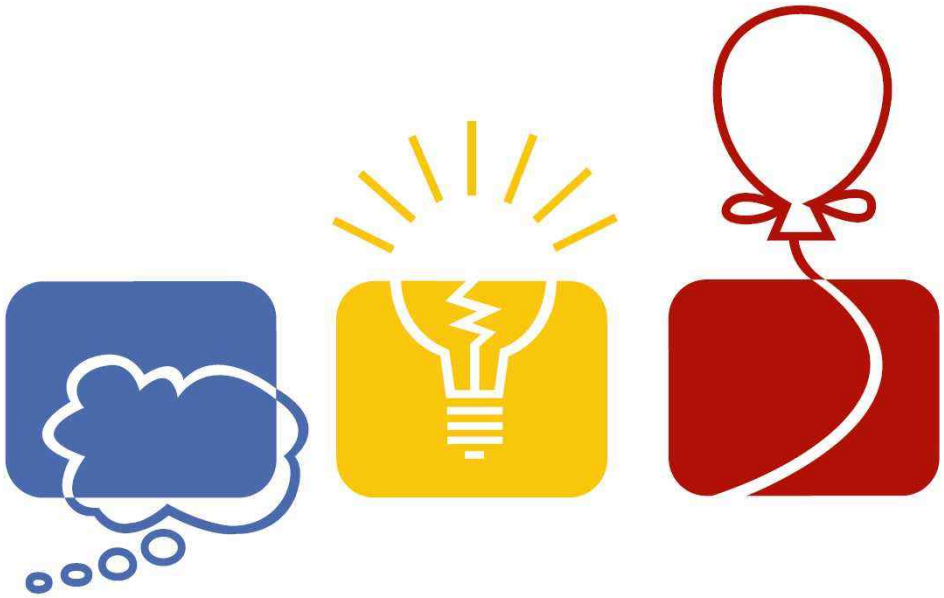

ACM/ICPC Template Manaual

ZheJiang GongShang University

Happy Otaku

July 12, 2019

Contents

0 头文件

```cpp
// 巨菜的ACMer-Happy233

#include <bits/stdc++.h>

using namespace std;

//-----
typedef long long ll;
typedef vector<int> vi;
typedef pair<int, int> pii;
#define pw(x) (1ll << (x))
#define sz(x) ((int)(x).size())
#define all(x) (x).begin(),(x).end()
#define rep(i, l, r) for(int i=(l);i<(r);++i)
#define per(i, l, r) for(int i=(r)-1;i>=(l);--i)
#define sf(x) scanf("%d", &(x))

using namespace std;

const double pi = acos(-1);
```

0.1 stdc++

```cpp
// C
#ifndef _GLIBCXX_NO_ASSERT
#include <cassert>
#endif
#include <cctype>
#include <cerrno>
#include <cfloat>
#include <ciso646>
#include <climits>
#include <clocale>
#include <cmath>
#include <csetjmp>
#include <csignal>
#include <cstdarg>
#include <cstddef>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>

#if __cplusplus >= 201103L
#include <ccomplex>
#include <cfenv>
#include <cinttypes>
#include <cstdalign>
#include <cstdbool>
#include <cstdint>
#include <ctgmath>
#include <cuchar>
#include <cwchar>
#include <cwctype>
#endif

```

```cpp
// C++
#include <algorithm>
#include <bitset>
#include <complex>
#include <deque>
#include <exception>
#include <fstream>
#include <functional>
#include <iomanip>
#include <ios>
#include <iosfwd>
#include <iostream>
#include <istream>
#include <iterator>
#include <limits>
#include <list>
#include <locale>
#include <map>
#include <memory>
#include <new>
#include <numeric>
#include <ostream>
#include <queue>
#include <set>
#include <sstream>
#include <stack>
#include <stdexcept>
#include <streambuf>
#include <string>
#include <typeinfo>
#include <utility>
#include <valarray>
#include <vector>

#if __cplusplus >= 201103L
#include <array>
#include <atomic>
#include <chrono>
#include <codecvt>
#include <condition_variable>
#include <forward_list>
#include <future>
#include <initializer_list>
#include <mutex>
#include <random>
#include <ratio>
#include <regex>
#include <scoped_allocator>
#include <system_error>
#include <thread>
#include <tuple>
#include <typeindex>
#include <type_traits>
#include <unordered_map>
#include <unordered_set>
#endif
#if __cplusplus >= 201402L
#include <shared_mutex>
#endif
```

1 字串符

1.1 KMP

```
template<class elemType>
inline void kmp_nxt(elemType &T, vector<int> &nxt) {
    nxt[0] = -1;
    for (int i = 1; i < T.size(); i++) {
        int j = nxt[i - 1];
        while (j >= 0 && T[i - 1] != T[j]) j = nxt[j];
        if (j >= 0 && T[i - 1] == T[j]) nxt[i] = j + 1;
        else nxt[i] = 0;
    }
}

template<class elemType>
inline int kmp_count(elemType &S, elemType &T) {
    vector<int> nxt(T.size());
    kmp_nxt(T, nxt);
    int index, count = 0;
    for (index = 0; index < S.size(); ++index) {
        int pos = 0;
        int iter = index;
        while (pos < T.size() && iter < S.size()) {
            if (S[iter] == T[pos]) {
                ++iter;
                ++pos;
            } else {
                if (pos == 0) ++iter;
                else pos = nxt[pos - 1] + 1;
            }
        }
        if (pos == T.size() && (iter - index) == T.size()) ++count;
    }
    return count;
}

template<class elemType>
inline void kmp_next(elemType T[], int count, vector<int> &nxt) {
    nxt[0] = -1;
    for (int i = 1; i < count; i++) {
        int j = nxt[i - 1];
        while (j >= 0 && T[i - 1] != T[j]) j = nxt[j];
        if (j >= 0 && T[i - 1] == T[j]) nxt[i] = j + 1;
        else nxt[i] = 0;
    }
}

template<class elemType>
inline int kmp_count(elemType S[], int c1, elemType T[], int c2) {
    vector<int> nxt(c2);
    kmp_nxt(T, c2, nxt);
    int index, count = 0;
    for (index = 0; index < c1; ++index) {
        int pos = 0;
        int iter = index;
        while (pos < c2 && iter < c1) {
            if (S[iter] == T[pos]) {
                ++iter;
```

```
56                  ++pos;
57              }
58              else {
59                  if (pos == 0) ++iter;
60                  else pos = nxt[pos - 1] + 1;
61              }
62          }
63          if (pos == c2 && (iter - index) == c2) ++count;
64      }
65      return count;
66  }
```

2 动态规划

2.1 01Bag

```
void dp(int n, int m) {
    // n=物品个数
    for (int i = 0; i < n; i++) {
        // m=背包最大容量
        for (int j = m; j >= wei[i]; j--)
            // wei=大小 val=价值
            f[j] = max(f[j], f[j - wei[i]] + val[i]);
    }
}
```

2.2 BagProblem

```
#define N 1000
// val=价值 wei=重量 num=数量
int val[N], wei[N], num[N], f[N];
// n=种类个数 m=背包最大值

// 01背包
void dp1(int n, int m) {
    for (int i = 0; i < n; i++) {
        for (int j = m; j >= wei[i]; j--)
            f[j] = max(f[j], f[j - wei[i]] + val[i]);
    }
}

// 完全背包
void dp2(int n, int m) {
    //初始化看要求
    for (int i = 0; i <= m; i++) {
        f[i] = INF;
    }
    f[0] = 0;
    //若要求恰好装满背包，那在初始化时除了f[0]=0其它f[1..V]均=-∞
    //若没要求背包装满，只希望价格大，初始化时应将f[0..V]=0)
    for (int i = 0; i < n; i++)
        for (int j = wei[i]; j <= m; j++)
            f[j] = max(f[j], f[j - wei[i]] + val[i]);
}

// 多重背包
void dp3(int n, int m) {
    for (int i = 0; i < n; i++)
        for (int k = 0; k < num[i]; k++)
            for (int j = m; j >= wei[i]; j--)
                f[j] = max(f[j], f[j - wei[i]] + val[i]);
}
```

2.3 FullBag

```
/*
完全背包问题的特点是，每种物品可以无限制的重复使用，可以选择放或不放。
完全背包问题描述:
有N物品和一个容量为V的背包。第i件物品的重量是wei[i]，价值是val[i]。
```

```
*/

#include <cstdio>
#define INF 0x3fffffff
#define N 10047
int f[N],val[N],wei[N];
int min(int a,int b)
{
    return x<y?x:y;
}
int main()
{
    int t,i,j,k,E,F,m,n;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d%d",&E,&F);
        int c = F-E;
        for(i = 0 ; i <= c ; i++)
            f[i]=INF;
        scanf("%d",&n);
        for(i = 0 ; i < n ; i++)
        {
            scanf("%d%d",&val[i],&wei[i]);//val[i]为面额，wei[i]为重量
        }
        f[0]=0;//因为此处假设的是小猪储钱罐 恰好装满 的情况
        //注意初始化（要求恰好装满背包，那么在初始化时除了f[0]为0其它f[1..V]均设为-∞，
        //这样就可以保证最终得到的f[N]是一种恰好装满背包的最优解。
        //如果并没有要求必须把背包装满，而是只希望价格尽量大，初始化时应该将f[0..V]全部设为0）
        for(i =0 ; i < n ; i++)
        {
            for(j = wei[i] ; j <= c ; j++)
            {
                f[j] = min(f[j],f[j-wei[i]]+val[i]);//此处求的是最坏的情况所以用min，确定最少的钱,当然最后就用max了，HEHE
            }
        }
        if(f[c] == INF)
            printf("This is impossible.\n");
        else
            printf("The minimum amount of money in the piggy-bank is %d.\n",f[c]);
    }
    return 0;
}
//此代码为HDU1114;
```

2.4 MultiBag

```
//多重背包(MultiplePack)：有N种物品和一个容量为V的背包。
//第i种物品最多有n[i]件可用，每件费用是c[i]，价值是w[i]。
//求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，
//且价值总和最大。
//HDU 2191

#include <cstdio>
#include <cstring>
#define N 247
int max(int x,int y){
```

```
11      return x>y?x:y;
12  }
13  int main()  {
14      int t,n,m,i,j,k;
15      int w[N],pri[N],num[N],f[N];
16      while(~scanf("%d",&t)){
17          while(t--){
18              memset(f,0,sizeof(f));
19              scanf("%d%d",&n,&m);//n为总金额，m为大米种类
20              for(i = 0 ; i < m ; i++){
21                  scanf("%d%d%d",&pri[i],&w[i],&num[i]);//num[i]为每种大米的袋数
22              }
23              for(i = 0 ; i < m ; i++){
24                  for(k = 0 ; k < num[i] ; k++){
25                      for(j = n ; j >= pri[i]; j--){
26                          f[j] = max(f[j],f[j-pri[i]]+w[i]);
27                      }
28                  }
29              }
30              printf("%d\n",f[n]);
31          }
32      }
33      return 0;
34  }
```

3 数据结构

3.1 BTree

```
template<class T>

struct TreeNode {
    T value;
    TreeNode *left;
    TreeNode *right;
};

template<class T>
TreeNode<T> *createTree(const T *pre, const T *in, const int len) {
    TreeNode<T> *t = NULL;
    if (len > 0) {
        t = new TreeNode<T>;
        t->value = pre[0];
        int index;
        for (index = 0; index < len; index++) {
            if (in[index] == pre[0]) {
                break;
            }
        }
        if (index == len) {
            index = -1;
        }
        t->left = createTree(pre + 1, in, index);
        t->right = createTree(pre + index + 1, in + index + 1, len - index - 1);
    }
    return t;
}

template<class T>
int preOrder(TreeNode<T> *root, queue<T> &out) {
    if (root) {
        int count = 1;
        out.push(root->value);
        count += preOrder(root->left, out);
        count += preOrder(root->right, out);
        return count;
    } else {
        return 0;
    }
}

template<class T>
int inOrder(TreeNode<T> *root, queue<T> &out) {
    if (root) {
        int count = 1;
        count += inOrder(root->left, out);
        out.push(root->value);
        count += inOrder(root->right, out);
        return count;
    } else {
        return 0;
    }
}

```

```
56  template<class T>
57  void postOrder(TreeNode<T> *root, queue<T> &out) {
58      if (root) {
59          postOrder(root->left, out);
60          postOrder(root->right, out);
61          out.push(root->value);
62      } else {
63          return;
64      }
65  }
66
67  template<class T>
68  T *convertQueueToArray(queue<T> &out, int len) {
69      T *list = new T[len];
70      int now = 0;
71      while (!out.empty() && now < len) {
72          list[now] = out.front();
73          out.pop();
74          now++;
75      }
76      return list;
77  }
78
79  template<class T>
80  void destroyTree(TreeNode<T> *root) {
81      if (root) {
82          destroyTree(root->left);
83          destroyTree(root->right);
84          delete root;
85      } else return;
86  }
87
88  template<class T>
89  void insertIntoBSTree(TreeNode<T> *root, const T &value) {
90      if (!root) {
91          return;
92      }
93      if (value < root->value) {
94          if (root->left) {
95              insertIntoTree(root->left, value);
96          } else {
97              root->left = new TreeNode<T>;
98              root->left->value = value;
99              root->left->left = NULL;
100             root->left->right = NULL;
101         }
102     } else if (value > root->value) {
103         if (root->right) {
104             insertIntoTree(root->right, value);
105         } else {
106             root->right = new TreeNode<T>;
107             root->right->value = value;
108             root->right->left = NULL;
109             root->right->right = NULL;
110         }
111     }
112 }
113
114 template<class T>
```

```
TreeNode<T> *createBSTree(T *list, int len) {
    if (len < 1) {
        return NULL;
    }
    TreeNode<T> *root = new TreeNode<char>;
    root->value = list[0];
    root->left = NULL;
    root->right = NULL;
    for (int i = 1; i < len; i++) {
        insertIntoBSTree(root, list[i]);
    }
    return root;
}
```

3.2 pbds-bbtree

```
// RBTree 红黑树
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
// 红黑树
__gnu_pbds::tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update> t;
// null_type无映射(低版本g++为null_mapped_type)
// 类似multiset
__gnu_pbds::tree<int, null_type, less_equal<int>, rb_tree_tag,
    tree_order_statistics_node_update> t;
find_by_order(size_t order);
// 结点更新
tree_order_statistics_node_update
insert(p);
erase(it);
// 求k在树中是第几大:
order_of_key(p);
// 找到第order小的迭代器
find_by_order(order);
// 前驱
lower_bound(p);
// 后驱
upper_bound(p);
// 合并
a.join(b);
// 分割 key小于等于v的元素属于a, 其余的属于b
a.split(v, b);

// 优先队列
#include <ext/pb_ds/priority_queue.hpp>
#include <ext/pb_ds/assoc_container.hpp>
// 操作类似于stl的优先队列
typedef __gnu_pbds::priority_queue<node, greater<node>, __gnu_pbds::thin_heap_tag> heap
    ;
heap::point_iterator; // 指向元素的指针
```

3.3 树状数组

```
const int N = 1000005;
struct BITree {
    int n;
```

```
    ll c[N];

    void init(int _n) {
        n = _n;
        memset(c, 0, sizeof(ll) * ++n);
    }

    void change(int pos, ll v) {
        for (int i = pos; i < n; i += i & (-i))
            c[i] += v;
    }

    ll query(int x) {
        ll ans = 0;
        for (int i = x; i > 0; i -= i & (-i))
            ans += c[i];
        return ans;
    }

    void update(int l, int r, ll v) {
        change(l, v);
        change(r + 1, -v);
    }
};
```

3.4 二维树状数组

```
const int N = 2005;

inline int lowbit(const int &x) {
        return x & -x;
    }

struct TdBITree {
    int n, m;
    ll c[N][N];

    void init(int n, int m) {
        this->n = n;
        this->m = m;
        memset(c, 0, sizeof(c))
    }

    void init(int n, int m, ll v) {
        this->n = n;
        this->m = m;
        rep(x, 1, N) {
            rep(y, 1, N) {
                c[x][y] = (x * y + (x - lowbit(x)) * (y - lowbit(y)) - x * (y - lowbit(
    y)) - (x - lowbit(x)) * y) * v;
            }
        }
    }

    void change(int x, int y, ll v) {
        for (int i = x; i <= n; i += lowbit(i))
            for (int j = y; j <= m; j += lowbit(j))
                c[i][j] += v;
```

```
    }

    ll query(int x, int y) {
        ll ans = 0;
        for (int i = x; i >= 1; i -= lowbit(i))
            for (int j = y; j >= 1; j -= lowbit(j))
                ans += c[i][j];
        return ans;
    }

    ll solve(int x1, int y1, int x2, int y2) {
        return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) + query(x1 - 1, y1 - 1);
    }
};
```

3.5 线段树

```
const int N = 50005;

struct SegTree {
    ll c[N];
    ll ans[N << 2];
    ll laz[N << 2];

    void init(int n) {
        memset(c, 0, sizeof(ll) * (n + 1));
    }

    inline void up(int k) {
        ans[k] = ans[k << 1] + laz[k << 1] + ans[k << 1 | 1] + laz[k << 1 | 1];
    }

    inline void push(int k) {
        laz[k << 1] += laz[k];
        laz[k << 1 | 1] += laz[k];
        ans[k] += laz[k];
        laz[k] = 0;
    }

    void build(int x, int y, int k) {
        laz[k] = 0;
        if (x == y) {
            ans[k] = c[x];
            return;
        }
        int m = (x + y) >> 1;
        build(x, m, k << 1);
        build(m + 1, y, k << 1 | 1);
        up(k);
    }

    void change(int x, int y, int k, int l, int r, ll v) {
        if (x == l && y == r) {
            laz[k] += v;
            return;
        }
        push(k);
```

```
        int m = (x + y) >> 1;
        if (r <= m) change(x, m, k << 1, l, r, v);
        else if (l > m)change(m + 1, y, k << 1 | 1, l, r, v);
        else change(x, m, k << 1, l, m, v), change(m + 1, y, k << 1 | 1, m + 1, r, v);
        up(k);
    }

    ll query(int x, int y, int k, int l, int r) {
        if (x == l && y == r) {
            return ans[k] + laz[k];
        }
        int m = (x + y) >> 1;
        push(k);
        if (r <= m) return query(x, m, k << 1, l, r);
        else if (l > m) return query(m + 1, y, k << 1 | 1, l, r);
        else return query(x, m, k << 1, l, m) + query(m + 1, y, k << 1 | 1, m + 1, r);
    }
};
```

3.6 二维线段树

```

const int N = 1005;

struct SegTree {

    inline int son(int k, int x) {
        return (k << 2) - 2 + x;
    }

    struct node {
        int l, r;

        node() = default;

        node(int a, int b) : l(a), r(b) {}

        inline int mid() {
            return (l + r) >> 1;
        }

        inline node left() {
            return node(l, mid());
        }

        inline node right() {
            return node(mid() + 1, r);
        }

        inline bool in(int x) {
            return x >= l && x <= r;
        }

        inline bool more() {
            return l < r;
        }

        bool operator==(const node &t) {
```

```
38              return l == t.l && r == t.r;
39          }
40      };
41
42      ll c[N << 2][N << 2];
43      ll ans[N << 4];
44      ll laz[N << 4];
45
46      inline void up(int k, bool x, bool y) {
47          int s = (k << 2) - 2;
48          ll t = 0;
49          if (x) t += ans[s] + ans[s + 1] + laz[s] + laz[s + 1];
50          if (y) t += ans[s + 2] + ans[s + 3] + laz[s + 2] + laz[s + 3];
51          ans[k] = t;
52      }
53
54      inline void push(int k) {
55          int s = (k << 2) - 2;
56          laz[s] += laz[k];
57          laz[s + 1] += laz[k];
58          laz[s + 2] += laz[k];
59          laz[s + 3] += laz[k];
60          ans[k] += laz[k];
61          laz[k] = 0;
62      }
63
64      void build(node x, node y, int k) {
65          laz[k] = 0;
66          if (x.more() && y.more()) {
67              ans[k] = c[x.l][y.l];
68              return;
69          }
70          ans[k] = 0;
71          bool ax = false;
72          bool ay = false;
73          if (x.more()) {
74              build(x.left(), y, son(k, 0));
75              build(x.right(), y, son(k, 1));
76          }
77          if (y.more()) {
78              build(x, y.left(), son(k, 2));
79              build(x, y.right(), son(k, 3));
80          }
81          up(k, x.more(), y.more());
82      }
83
84      void change(node x, node y, int k, node l, node r, ll v) {
85          if (x == l && y == r) {
86              laz[k] += v;
87              return;
88          }
89          push(k);
90          if (x.more()) {
91              if (l.r <= x.mid()) {
92                  change(x.left(), y, son(k, 0), l, r, v);
93              } else if (l.l > x.mid()) {
94                  change(x.right(), y, son(k, 1), l, r, v);
95              } else {
96                  change(x.left(), y, son(k, 0), node(l.l, x.mid()), r, v);
```

```
                change(x.right(), y, son(k, 1), node(x.mid() + 1, l.r), r, v);
            }
        }
        if (y.more()) {
            if (r.l <= y.mid()) {
                change(x, y.left(), son(k, 2), l, r, v);
            } else if (r.r > y.mid()) {
                change(x, y.right(), son(k, 3), l, r, v);
            } else {
                change(x, y.left(), son(k, 2), l, node(r.l, y.mid()), v);
                change(x, y.right(), son(k, 3), l, node(y.mid() + 1, r.r), v);
            }
        }
        up(k, x.more(), y.more());
    }

    ll query(node x, node y, int k, node l, node r) {
        if (x == l && y == r) {
            return ans[k] + laz[k];
        }
        push(k);
        ll t = 0;
        if (x.more()) {
            if (l.r <= x.mid()) {
                t += query(x.left(), y, son(k, 0), l, r);
            } else if (l.l > x.mid()) {
                t += query(x.right(), y, son(k, 1), l, r);
            } else {
                t += query(x.left(), y, son(k, 0), node(l.l, x.mid()), r);
                t += query(x.right(), y, son(k, 1), node(x.mid() + 1, l.r), r);
            }
        }
        if (y.more()) {
            if (r.l <= y.mid()) {
                t += query(x, y.left(), son(k, 2), l, r);
            } else if (r.r > y.mid()) {
                t += query(x, y.right(), son(k, 3), l, r);
            } else {
                t += query(x, y.left(), son(k, 2), l, node(r.l, y.mid()));
                t += query(x, y.right(), son(k, 3), l, node(y.mid() + 1, r.r));
            }
        }
        return t;
    }
};
```

3.7 树状数组求逆序对

```
BITree t;
int n;
pii a[N];

void solve() {
    t.init(n);
    for (int i = 1; i <= n; i++) {
        int x;
        cin >> x;
        a[i] = make_pair(x, i);
```

```
    }
    sort(a + 1, a + n + 1);
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        t.change(a[i].second, 1);
        ans += (i - t.query(a[i].second));
    }
    cout << ans << endl;
}
```

3.8 ST

```
struct ST {
    int ck[N];
    int rmq[N];
    int dp[20][N];

    void init(int n, int squ[]) {
        ck[0] = -1;
        for (int i = 1; i <= n; i++) {
            ck[i] = ck[i - 1] + ((i & (i - 1)) == 0 ? 1 : 0);
        }
        memcpy(rmq, squ, sizeof(int) * n);
        for (int i = 0; i < n; i++) {
            dp[0][i] = i;
        }
        for (int k = 1; k <= ck[k]; k++) {
            int dk = k - 1;
            for (int i = 0; i < n; i++) {
                int a = dp[dk][i];
                int b = dp[dk][i + (1 << dk)];
                dp[k][i] = rmq[a] < rmq[b] ? a : b;
            }
        }
    }

    int query(int l, int r) {
        if (l > r) swap(l, r);
        int k = ck[r - l + 1];
        int a = dp[k][l];
        int b = dp[k][r - (1 << k) + 1];
        return rmq[a] < rmq[b] ? a : b;
    }
};
```

4 图论

4.1 Graph

```
struct Edge {
    int e, nxt;
    ll v;
    Edge() = default;
    Edge(int a, ll b, int c = 0) : e(a), v(b), nxt(c) {}

    bool operator<(const Edge &a) const {
        return (a.v == v ? e < a.e : v < a.v);
    }
};

const ll INF = ll(1e11);
const int N = int(1e5 + 10);
const int M = int(3e5 + 10);

struct Graph {
    Edge eg[M];
    int head[N];
    int cnt;

    void init(int n) {
        memset(head, -1, sizeof(int) * ++n);
        cnt = 0;
    }

    inline void addEdge(int x, int y, ll v) {
        eg[cnt] = Edge(y, v, head[x]);
        head[x] = cnt++;
    }

    inline int begin(int p) {
        return head[p];
    }

    inline Edge &operator[](int i) {
        return eg[i];
    }

    inline int next(int i) {
        return eg[i].nxt;
    }
} gh;
```

4.2 Dijkstra

```
int dist[N];
int path[N];

void bfs(int s, int n) {
    n++;
    rep(i, 0, n) dist[i] = INF;
    memset(path, -1, sizeof(int) * n);
    dist[s] = 0;
    path[s] = s;
```

```
    // 注意优先队列默认less运算，但选择最大的作为top，注意cmp！！！
    priority_queue<Edge, vector<Edge>, greater<Edge>> q;
    q.push(Edge(s, dist[s]));
    while (!q.empty()) {
        Edge f = q.top();
        q.pop();
        for (int i = gh.head[f.e]; ~i; i = gh.eg[i].nxt) {
            Edge &t = gh.eg[i];
            if (dist[t.e] > f.v + t.v) {
                dist[t.e] = f.v + t.v;
                path[t.e] = f.e;
                q.push(Edge(t.e, dist[t.e]));
            }
        }
    }
}

#include <ext/pb_ds/priority_queue.hpp>
#include <ext/pb_ds/assoc_container.hpp>
typedef __gnu_pbds::priority_queue<Edge, greater<Edge>> heap;
// 使用该模板，需要注意因为使用了greater，所以需要重载大于运算
// 默认pairing_heap_tag
// push O(1), pop O(logn) modify O(logn) erase O(logn) join O(1)
// 可选thin_heap_tag
// push O(1), pop O(logn) modify O(1) erase O(logn) join O(n)

heap::point_iterator its[N];
int cnt[N];

void bfs(int s, int n) {
    n++;
    rep(i, 0, n) dist[i] = INF;
    memset(cnt, 0, sizeof(int) * n);
    dist[s] = 0;
    cnt[s] = 1;
    heap q;
    its[s] = q.push(Edge(s, dist[s]));
    while (!q.empty()) {
        Edge f = q.top();
        q.pop();
        for (int i = gh.head[f.e]; ~i; i = gh.eg[i].nxt) {
            Edge &t = gh.eg[i];
            its[t.e] = 0;
            int v = f.v + t.v;
            if (dist[t.e] > v) {
                dist[t.e] = v;
                if (its[t.e] != 0) {
                    q.modify(its[t.e], Edge(t.e, dist[t.e]));
                } else {
                    its[t.e] = q.push(Edge(t.e, dist[t.e]));
                }
                cnt[t.e] = cnt[f.e];
            } else if (dist[t.e] == v) {
                (cnt[t.e] += cnt[f.e]) %= 100003;
            }
        }
    }
}
```

4.3 spfa

```
vector<int> dist;
vector<vector<node>> eg;
vector<int> path;

bool spfa(int n, int start) {
    dist.assign(n, INF);
    dist[start] = 0;
    deque<int> q;
    q.push_back(start);
    path.assign(n, -1);
    vector<int> cnt(n, 0);
    vector<bool> flag(n, false);
    cnt[start] = flag[start] = true;
    while (!q.empty()) {
        const int now = q.front();
        q.pop_front();
        flag[now] = false;
        for (auto i: eg[now]) {
            if (dist[i.x] > dist[now] + i.d) {
                dist[i.x] = dist[now] + i.d;
                path[i.x] = now;
                if (!flag[i.x]) {
                    if (n == ++cnt[i.x]) return false;
                    //队列非空且优于队首 (SLF)
                    if (!q.empty() && dist[i.x] < dist[q.front()]) {
                        q.push_front(i.x);
                    } else {
                        q.push_back(i.x);
                    }
                    flag[i.x] = true;
                }
            }
        }
    }
    return true;
}
```

4.4 Dinic

```
struct Dinic {
    Graph gh;
    // 点的范围[0, n)
    int n;
    // 弧优化
    int cur[N], dis[N];

    Dinic(){};

    // 设置N
    void init(int _n) {
        n = _n;
        gh.init(n);
    }

    // 加流量
    void addFlow(int x, int y, ll f) {
```

```
        gh.addEdge(x, y, f);
        gh.addEdge(y, x, 0);
    }

    bool bfs(int s, int e) {
        memset(dis, -1, sizeof(int) * n);
        int q[N];
        int l, r;
        l = r = 0;
        dis[s] = 0;
        q[r++] = s;
        while (l < r) {
            int f = q[l++];
            for (int i = gh.head[f]; ~i; i = gh.eg[i].nxt) {
                if (gh.eg[i].v > 0 && dis[gh.eg[i].e] == -1) {
                    dis[gh.eg[i].e] = dis[f] + 1;
                    q[r++] = gh.eg[i].e;
                }
            }
        }
        return dis[e] > 0;
    }

    ll dfs(int s, int e, ll mx) {
        if (s == e || mx == 0) {
            return mx;
        }
        ll flow = 0;
        for (int &k = cur[s]; ~k; k = gh.eg[k].nxt) {
            auto &eg = gh.eg[k];
            ll a;
            if (eg.v > 0 && dis[eg.e] == dis[s] + 1 && (a = dfs(eg.e, e, min(eg.v, mx))
    )) {
                eg.v -= a;
                gh.eg[k ^ 1].v += a;
                flow += a;
                mx -= a;
                if (mx <= 0) break;
            }
        }
        return flow;
    }

    ll max_flow(int s, int e) {
        ll ans = 0;
        while (bfs(s, e)) {
            memcpy(cur, gh.head, sizeof(int) * n);
            ans += dfs(s, e, INF);
        }
        return ans;
    }
} dinic;
```

4.5 hungry

```
#define N 105
#define M 10005
int n, m, k;
```

```
pii eg[M * 2];
int result[N * 2];
int head[N * 2];
int cnt = 0;

void addEdge(int x, int y) {
    eg[cnt].first = y;
    eg[cnt].second = head[x];
    head[x] = cnt++;
}

bool vis[M * 2] = {false};

int dfs(int x) {
    for (int i = head[x]; ~i; i = eg[i].second) {
        int y = eg[i].first;
        if (!vis[y]) {
            vis[y] = true;
            if (result[y] == -1 || dfs(result[y])) {
                result[y] = x;
                return 1;
            }
        }
    }
    return 0;
}

int MaxMatch() {
    int ans = 0;
    memset(result, -1, sizeof(result));
    rep(i, 1, n + 1) {
        memset(vis, 0, sizeof(vis));
        ans += dfs(i);
    }
    return ans;
}

void solve() {
    scanf("%d%d", &m, &k);
    memset(head, -1, sizeof(head));
    cnt = 0;
    rep(i, 0, k) {
        int x, y;
        scanf("%d%d", &x, &y);
        addEdge(x, y);
    }
    int ans = MaxMatch();
    printf("%d\n", ans);
}
```

4.6 MinSpanTree

```
/*
* Prim 求 MST
* 耗费矩阵 cost[][]，标号从 0 开始，0□n-1
* 返回最小生成树的权值，返回 -1 表示原图不连通
 */
const int INF = 0x3f3f3f3f;
```

```
const int N = 110;
bool vis[N];
int lowc[N]; //点是 0 n-1
int prim(int cost[][N], int n) {
    int ans = 0;
    memset(vis, false, sizeof(vis));
    vis[0] = true;
    for (int i = 1; i < n; i++)lowc[i] = cost[0][i];
    for (int i = 1; i < n; i++) {
        int minc = INF;
        int p =-1;
        19
        for (int j = 0; j < n; j++)
            if (!vis[j] && minc > lowc[j]) {
                minc = lowc[j];
                p = j;
            }
        if (minc == INF)return -1;//原图不连通
        ans += minc;
        vis[p] = true;
        for (int j = 0; j < n; j++)
            if (!vis[j] && lowc[j] > cost[p][j])
                lowc[j] = cost[p][j];
    }
    return ans;
}
```

4.7 MinCostMaxFlow

```
struct Edge {
    int e, nxt;
    ll flow, cost;

    Edge() {};

    Edge(int a, ll b, ll c, int d = 0) : e(a), flow(b), cost(c), nxt(d) {}
};

const ll INF = 1000000;
const int N = int(1e5 + 10);
const int M = int(1e5 + 10);

struct Graph {
    Edge eg[M];
    int head[N];
    int cnt;

    void init(int n) {
        memset(head, -1, sizeof(int) * ++n);
        cnt = 0;
    }

    inline void addEdge(int x, int y, ll v, ll c) {
        eg[cnt] = Edge(y, v, c, head[x]);
        head[x] = cnt++;
    }
};

```

```cpp
struct MinCostMaxFlow {
    Graph gh;
    // 点的范围[0, n)
    int n;

    // 设置N
    void init(int _n) {
        n = _n;
        gh.init(n);
    }

    // 加流量，反向是负的花费
    void addFlow(int x, int y, ll f, ll c) {
        // printf("%d->%d: %lld\t%lld\n", x, y, f, c); fflush(stdout);
        gh.addEdge(x, y, f, c);
        gh.addEdge(y, x, 0, -c);
    }

    // 该pre存的是边
    int pre[N];
    int dis[N];
    bool vis[N];

    bool spfa(int s, int e) {
        queue<int> q;
        for (int i = 0; i < n; i++) {
            dis[i] = INF;
            vis[i] = false;
            pre[i] = -1;
        }
        dis[s] = 0;
        vis[s] = true;
        q.push(s);
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            vis[u] = false;
            for (int i = gh.head[u]; ~i; i = gh.eg[i].nxt) {
                Edge &eg = gh.eg[i];
                if (eg.flow > 0 && dis[eg.e] > dis[u] + eg.cost) {
                    dis[eg.e] = dis[u] + eg.cost;
                    pre[eg.e] = i;
                    if (!vis[eg.e]) {
                        vis[eg.e] = true;
                        q.push(eg.e);
                    }
                }
            }
        }
        return pre[e] != -1;
    }

    pll cal(int s, int e) {
        ll flow = 0, cost = 0;
        while (spfa(s, e)) {
            ll f = INF;
            for (int i = pre[e]; ~i; i = pre[gh.eg[i ^ 1].e]) {
                f = min(f, gh.eg[i].flow);
            }
```

```
            for (int i = pre[e]; ~i; i = pre[gh.eg[i ^ 1].e]) {
                gh.eg[i].flow -= f;
                gh.eg[i ^ 1].flow += f;
                cost += gh.eg[i].cost;
            }
            flow += f;
        }
        return make_pair(flow, cost);
    }

} network;
```

4.8 ISAP

```
struct ISAP {
    Graph gh;
    // 点的范围[0, n)
    int n;
    // 弧优化
    int cur[N], dis[N];
    ISAP() {};
    // 设置N
    void init(int _n) {
        n = _n;
        gh.init(n);
    }

    // 加流量
    inline void addFlow(int x, int y, ll f) {
        gh.addEdge(x, y, f);
        gh.addEdge(y, x, 0);
    }

    int dep[N]; // 记录距离标号
    int gap[N]; // gap常数优化
    int q[N]; // 数组模拟队列

    void bfs(int s, int e) {
        memset(dep, -1, sizeof(int) * n);
        memset(gap, 0, sizeof(int) * n);
        gap[0] = 1;
        dep[e] = 0;
        int l = 0, r = 0;
        q[r++] = e;
        while (l < r) {
            int u = q[l++];
            for (int i = gh.head[u]; ~i; i = gh.eg[i].nxt) {
                int v = gh.eg[i].e;
                if (~dep[v]) continue;
                q[r++] = v;
                dep[v] = dep[u] + 1;
                gap[dep[v]]++;
            }
        }
    }

    ll st[N]; // 栈优化

```

```
    ll max_flow(int s, int e) {
        bfs(s, e);
        memcpy(cur, gh.head, sizeof(int) * n);
        int top = 0;
        int u = s;
        ll ans = 0;
        while (dep[s] < N) {
            if (u == e) {
                ll mf = INF;
                int sel = 0;
                for (int i = 0; i < top; i++) {
                    if (mf > gh.eg[st[i]].v) {
                        mf = gh.eg[st[i]].v;
                        sel = i;
                    }
                }

                for (int i = 0; i < top; i++) {
                    gh.eg[st[i]].v -= mf;
                    gh.eg[st[i] ^ 1].v += mf;
                }
                ans += mf;
                top = sel;
                u = gh.eg[st[top] ^ 1].e;
                continue;
            }
            bool flag = false;
            int v = 0;
            for (int i = cur[u]; ~i; i = gh.eg[i].nxt) {
                v = gh.eg[i].e;
                if (gh.eg[i].v > 0 && dep[v] + 1 == dep[u]) {
                    flag = true;
                    cur[u] = i;
                    break;
                }
            }
            if (flag) {
                st[top++] = cur[u];
                u = v;
                continue;
            }
            int mind = N;
            for (int i = gh.head[u]; ~i; i = gh.eg[i].nxt) {
                if (gh.eg[i].v > 0 && dep[gh.eg[i].e] < mind) {
                    mind = dep[gh.eg[i].e];
                    cur[u] = i;
                }
            }
            gap[dep[u]]--; // 当前层无法连通，降层
            if (!gap[dep[u]]) return ans; // 断层结束运算
            dep[u] = mind + 1; // 进入更高层
            gap[dep[u]]++;
            if (u != s)  u = gh.eg[st[--top] ^ 1].e;
        }
        return ans;
    }
} isap;
```

5 博弈

5.1 GameProblem

```
// 巴什博奕，是否先手必胜
inline bool bash_game(int n, int m) {
    //一堆东西，n个物品,最多选m个
    return n % (m + 1);
}

// 威佐夫博弈，是否先手必胜
// 有两堆各若干的物品，两人轮流从其中一堆取至少一件物品，至多不限，或从两堆中同时取相同件物品，规定最后取完者胜利。
inline bool wythoff_game(int n, int m) {
    if (n > m) {
        swap(n, m);
    }
    int temp = floor((n2 - n1) * (1 + sqrt(5.0)) / 2.0);
    return temp != n1;
}
// SG函数
#define N 1001
//f[]: 可以取走的石子个数
//sg[]:0~n的SG函数值
int f[N], sg[N], mex[N];

void getSG(int n) {
    int i, j;
    memset(sg, 0, sizeof(sg));
    for (i = 1; i <= n; i++) {
        memset(mex, 0, sizeof(mex));
        for (j = 1; f[j] <= i; j++)
            mex[sg[i - f[j]]] = 1;
        for (j = 0; j <= n; j++) { //求mes{}中未出现的最小的非负整数
            if (mex[j] == 0) {
                sg[i] = j;
                break;
            }
        }
    }
}

// Auti-nim 反尼姆游戏
// 当先拿完所有石子时候输
// 当如下条件时，先手必胜
// □: 所有堆的石子数均=1，且有偶数堆。
// □: 至少有一个堆的石子数>1，且石子堆的异或和≠0。
```

6 分治

6.1 IntegerFastPower

```
ll fpow(ll x, ll k) {
    ll base = x, r = 1;
    for (; k; k >>= 1) {
        if (k & 1) r = r * base;
        base = base * base;
    }
    return r;
}
```

6.2 MatrixFastPower

```
#define MAX_N 10
#define mod_num 9973

struct Mat {
    long long mat[MAX_N][MAX_N];
    long long n;
    Mat() {
        memset(mat, 0, sizeof(mat));
        n = 0;
    }
    Mat(long long n) {
        memset(mat, 0, sizeof(mat));
        this->n = n;
    }
    void init() {
        for (int i = 0; i < n; ++i) {
            mat[i][i] = 1;
        }
    }
    Mat(const long long ** list, long long n) {
        this->n = n;
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                mat[i][j] = list[i][j];
            }
        }
    }
};

Mat operator * (Mat a, Mat b) {
    long long n = a.n;
    Mat c(n);
    memset(c.mat, 0, sizeof(c.mat));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            for (int k = 0; k < n; ++k) {
                c.mat[i][j] += (a.mat[i][k] * b.mat[k][j]) % mod_num;
                c.mat[i][j] %= mod_num;
            }
        }
    }
    return c;
}
```

```
44
45 Mat operator ^ (Mat a, int k) {
46     long long n = a.n;
47     Mat c(n);
48     c.init();
49     for (; k; k >>= 1) {
50         if (k & 1)   c = c * a;
51         a = a * a;
52     }
53     return c;
54 }
```

7 其他

7.1 BigInteger

```
// base and base_digits must be consistent
constexpr int base = 1000000000;
constexpr int base_digits = 9;

struct bigint {
    // value == 0 is represented by empty z
    vector<int> z; // digits

    // sign == 1 <==> value >= 0
    // sign == -1 <==> value < 0
    int sign;

    bigint() : sign(1) {}

    bigint(ll v) { *this = v; }

    bigint &operator=(ll v) {
        sign = v < 0 ? -1 : 1;
        v *= sign;
        z.clear();
        for (; v > 0; v = v / base) z.push_back((int) (v % base));
        return *this;
    }

    bigint(const string &s) { read(s); }

    bigint &operator+=(const bigint &other) {
        if (sign == other.sign) {
            for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
                if (i == z.size())
                    z.push_back(0);
                z[i] += carry + (i < other.z.size() ? other.z[i] : 0);
                carry = z[i] >= base;
                if (carry)
                    z[i] -= base;
            }
        } else if (other != 0 /* prevent infinite loop */) {
            *this -= -other;
        }
        return *this;
    }

    friend bigint operator+(bigint a, const bigint &b) { return a += b; }

    bigint &operator-=(const bigint &other) {
        if (sign == other.sign) {
            if (sign == 1 && *this >= other || sign == -1 && *this <= other) {
                for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
                    z[i] -= carry + (i < other.z.size() ? other.z[i] : 0);
                    carry = z[i] < 0;
                    if (carry)
                        z[i] += base;
                }
                trim();
            } else {
```

```
56                 *this = other - *this;
57                 this->sign = -this->sign;
58             }
59         } else {
60             *this += -other;
61         }
62         return *this;
63     }
64
65     friend bigint operator-(bigint a, const bigint &b) {
66         return a -= b;
67     }
68
69     bigint &operator*=(int v) {
70         if (v < 0) sign = -sign, v = -v;
71         for (int i = 0, carry = 0; i < z.size() || carry; ++i) {
72             if (i == z.size()) z.push_back(0);
73             ll cur = (ll) z[i] * v + carry;
74             carry = (int) (cur / base);
75             z[i] = (int) (cur % base);
76         }
77         trim();
78         return *this;
79     }
80
81     bigint operator*(int v) const { return bigint(*this) *= v; }
82
83     friend pair<bigint, bigint> divmod(const bigint &a1, const bigint &b1) {
84         int norm = base / (b1.z.back() + 1);
85         bigint a = a1.abs() * norm;
86         bigint b = b1.abs() * norm;
87         bigint q, r;
88         q.z.resize(a.z.size());
89
90         for (int i = (int) a.z.size() - 1; i >= 0; i--) {
91             r *= base;
92             r += a.z[i];
93             int s1 = b.z.size() < r.z.size() ? r.z[b.z.size()] : 0;
94             int s2 = b.z.size() - 1 < r.z.size() ? r.z[b.z.size() - 1] : 0;
95             int d = (int) (((ll) s1 * base + s2) / b.z.back());
96             r -= b * d;
97             while (r < 0) r += b, --d;
98             q.z[i] = d;
99         }
100
101         q.sign = a1.sign * b1.sign;
102         r.sign = a1.sign;
103         q.trim();
104         r.trim();
105         return {q, r / norm};
106     }
107
108     friend bigint sqrt(const bigint &a1) {
109         bigint a = a1;
110         while (a.z.empty() || a.z.size() % 2 == 1) a.z.push_back(0);
111
112         int n = a.z.size();
113
114         int firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
```

```
115         int norm = base / (firstDigit + 1);
116         a *= norm;
117         a *= norm;
118         while (a.z.empty() || a.z.size() % 2 == 1) a.z.push_back(0);
119 
120         bigint r = (ll) a.z[n - 1] * base + a.z[n - 2];
121         firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
122         int q = firstDigit;
123         bigint res;
124 
125         for (int j = n / 2 - 1; j >= 0; j--) {
126             for (;; --q) {
127                 bigint r1 = (r - (res * 2 * base + q) * q) * base * base +
128                             (j > 0 ? (ll) a.z[2 * j - 1] * base + a.z[2 * j - 2] : 0);
129                 if (r1 >= 0) {
130                     r = r1;
131                     break;
132                 }
133             }
134             (res *= base) += q;
135 
136             if (j > 0) {
137                 int d1 = res.z.size() + 2 < r.z.size() ? r.z[res.z.size() + 2] : 0;
138                 int d2 = res.z.size() + 1 < r.z.size() ? r.z[res.z.size() + 1] : 0;
139                 int d3 = res.z.size() < r.z.size() ? r.z[res.z.size()] : 0;
140                 q = (int) (((ll) d1 * base * base + (ll) d2 * base + d3) / (firstDigit
    * 2));
141             }
142         }
143 
144         res.trim();
145         return res / norm;
146     }
147 
148     bigint operator/(const bigint &v) const {
149         return divmod(*this, v).first;
150     }
151 
152     bigint operator%(const bigint &v) const {
153         return divmod(*this, v).second;
154     }
155 
156     bigint &operator/=(int v) {
157         if (v < 0) sign = -sign, v = -v;
158         for (int i = (int) z.size() - 1, rem = 0; i >= 0; --i) {
159             ll cur = z[i] + rem * (ll) base;
160             z[i] = (int) (cur / v);
161             rem = (int) (cur % v);
162         }
163         trim();
164         return *this;
165     }
166 
167     bigint operator/(int v) const {
168         return bigint(*this) /= v;
169     }
170 
171     int operator%(int v) const {
172         if (v < 0) v = -v;
```

```
        int m = 0;
        for (int i = (int) z.size() - 1; i >= 0; --i)
            m = (int) ((z[i] + m * (ll) base) % v);
        return m * sign;
    }

    bigint &operator*=(const bigint &v) {
        return *this = *this * v;;
    }

    bigint &operator/=(const bigint &v) {
        return *this = *this / v;
    }

    bool operator<(const bigint &v) const {
        if (sign != v.sign)
            return sign < v.sign;
        if (z.size() != v.z.size())
            return z.size() * sign < v.z.size() * v.sign;
        for (int i = (int) z.size() - 1; i >= 0; i--)
            if (z[i] != v.z[i])
                return z[i] * sign < v.z[i] * sign;
        return false;
    }

    bool operator>(const bigint &v) const { return v < *this; }

    bool operator<=(const bigint &v) const { return !(v < *this); }

    bool operator>=(const bigint &v) const { return !(*this < v); }

    bool operator==(const bigint &v) const { return !(*this < v) && !(v < *this); }

    bool operator!=(const bigint &v) const { return *this < v || v < *this; }

    void trim() {
        while (!z.empty() && z.back() == 0) z.pop_back();
        if (z.empty()) sign = 1;
    }

    bool isZero() const {
        return z.empty();
    }

    friend bigint operator-(bigint v) {
        if (!v.z.empty()) v.sign = -v.sign;
        return v;
    }

    bigint abs() const {
        return sign == 1 ? *this : -*this;
    }

    ll longValue() const {
        ll res = 0;
        for (int i = (int) z.size() - 1; i >= 0; i--)
            res = res * base + z[i];
        return res * sign;
    }
```

```
232
233     friend bigint gcd(const bigint &a, const bigint &b) {
234         return b.isZero() ? a : gcd(b, a % b);
235     }
236
237     friend bigint lcm(const bigint &a, const bigint &b) {
238         return a / gcd(a, b) * b;
239     }
240
241     void read(const string &s) {
242         sign = 1;
243         z.clear();
244         int pos = 0;
245         while (pos < s.size() && (s[pos] == '-' || s[pos] == '+')) {
246             if (s[pos] == '-') sign = -sign;
247             ++pos;
248         }
249         for (int i = (int) s.size() - 1; i >= pos; i -= base_digits) {
250             int x = 0;
251             for (int j = max(pos, i - base_digits + 1); j <= i; j++)
252                 x = x * 10 + s[j] - '0';
253             z.push_back(x);
254         }
255         trim();
256     }
257
258     friend istream &operator>>(istream &stream, bigint &v) {
259         string s;
260         stream >> s;
261         v.read(s);
262         return stream;
263     }
264
265     friend ostream &operator<<(ostream &stream, const bigint &v) {
266         if (v.sign == -1)
267             stream << '-';
268         stream << (v.z.empty() ? 0 : v.z.back());
269         for (int i = (int) v.z.size() - 2; i >= 0; --i)
270             stream << setw(base_digits) << setfill('0') << v.z[i];
271         return stream;
272     }
273
274     static vector<int> convert_base(const vector<int> &a, int old_digits, int
    new_digits) {
275         vector<ll> p(max(old_digits, new_digits) + 1);
276         p[0] = 1;
277         for (int i = 1; i < p.size(); i++)
278             p[i] = p[i - 1] * 10;
279         vector<int> res;
280         ll cur = 0;
281         int cur_digits = 0;
282         for (int v : a) {
283             cur += v * p[cur_digits];
284             cur_digits += old_digits;
285             while (cur_digits >= new_digits) {
286                 res.push_back(int(cur % p[new_digits]));
287                 cur /= p[new_digits];
288                 cur_digits -= new_digits;
289             }
```

```
        }
        res.push_back((int) cur);
        while (!res.empty() && res.back() == 0) res.pop_back();
        return res;
    }

    typedef vector<ll> vll;

    static vll karatsubaMultiply(const vll &a, const vll &b) {
        int n = a.size();
        vll res(n + n);
        if (n <= 32) {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    res[i + j] += a[i] * b[j];
            return res;
        }

        int k = n >> 1;
        vll a1(a.begin(), a.begin() + k);
        vll a2(a.begin() + k, a.end());
        vll b1(b.begin(), b.begin() + k);
        vll b2(b.begin() + k, b.end());

        vll a1b1 = karatsubaMultiply(a1, b1);
        vll a2b2 = karatsubaMultiply(a2, b2);

        for (int i = 0; i < k; i++) a2[i] += a1[i];
        for (int i = 0; i < k; i++) b2[i] += b1[i];

        vll r = karatsubaMultiply(a2, b2);
        for (int i = 0; i < a1b1.size(); i++) r[i] -= a1b1[i];
        for (int i = 0; i < a2b2.size(); i++) r[i] -= a2b2[i];

        for (int i = 0; i < r.size(); i++) res[i + k] += r[i];
        for (int i = 0; i < a1b1.size(); i++) res[i] += a1b1[i];
        for (int i = 0; i < a2b2.size(); i++) res[i + n] += a2b2[i];
        return res;
    }

    bigint operator*(const bigint &v) const {
        vector<int> a6 = convert_base(this->z, base_digits, 6);
        vector<int> b6 = convert_base(v.z, base_digits, 6);
        vll a(a6.begin(), a6.end());
        vll b(b6.begin(), b6.end());
        while (a.size() < b.size()) a.push_back(0);
        while (b.size() < a.size()) b.push_back(0);
        while (a.size() & (a.size() - 1)) a.push_back(0), b.push_back(0);
        vll c = karatsubaMultiply(a, b);
        bigint res;
        res.sign = sign * v.sign;
        for (int i = 0, carry = 0; i < c.size(); i++) {
            ll cur = c[i] + carry;
            res.z.push_back((int) (cur % 1000000));
            carry = (int) (cur / 1000000);
        }
        res.z = convert_base(res.z, 6, base_digits);
        res.trim();
        return res;
```

```
    }
};
```

7.2 FastIO

```
/*
 * FastIO
 * 代码模板 !
 * 如有雷同 !
 * 纯属巧合 !
 */
namespace FastIO {
#define BUF_SIZE 10000000
#define OUT_SIZE 10000000
#define ll long long
    //fread->read
    bool IOerror = 0;

    inline char nc() {
        static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
        if (p1 == pend) {
            p1 = buf;
            pend = buf + fread(buf, 1, BUF_SIZE, stdin);
            if (pend == p1) {
                IOerror = 1;
                return -1;
            }
            //{printf("IO error!\n");system("pause");for (;;);exit(0);}
        }
        return *p1++;
    }

    inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch ==
    '\t'; }

    inline void read(int &x) {
        bool sign = 0;
        char ch = nc();
        x = 0;
        for (; blank(ch); ch = nc());
        if (IOerror)return;
        if (ch == '-')sign = 1, ch = nc();
        for (; ch >= '0' && ch <= '9'; ch = nc())x = x * 10 + ch - '0';
        if (sign)x = -x;
    }

    inline void read(ll &x) {
        bool sign = 0;
        char ch = nc();
        x = 0;
        for (; blank(ch); ch = nc());
        if (IOerror)return;
        if (ch == '-')sign = 1, ch = nc();
        for (; ch >= '0' && ch <= '9'; ch = nc())x = x * 10 + ch - '0';
        if (sign)x = -x;
    }

    inline void read(double &x) {
```

```
53          bool sign = 0;
54          char ch = nc();
55          x = 0;
56          for (; blank(ch); ch = nc());
57          if (IOerror)return;
58          if (ch == '-')sign = 1, ch = nc();
59          for (; ch >= '0' && ch <= '9'; ch = nc())x = x * 10 + ch - '0';
60          if (ch == '.') {
61              double tmp = 1;
62              ch = nc();
63              for (; ch >= '0' && ch <= '9'; ch = nc())tmp /= 10.0, x += tmp * (ch - '0')
    ;
64          }
65          if (sign)x = -x;
66      }
67
68      inline void read(char *s) {
69          char ch = nc();
70          for (; blank(ch); ch = nc());
71          if (IOerror)return;
72          for (; !blank(ch) && !IOerror; ch = nc())*s++ = ch;
73          *s = 0;
74      }
75
76      inline void read(char &c) {
77          for (c = nc(); blank(c); c = nc());
78          if (IOerror) {
79              c = -1;
80              return;
81          }
82      }
83
84      //fwrite->write
85      struct Ostream_fwrite {
86          char *buf, *p1, *pend;
87          Ostream_fwrite() {
88              buf = new char[OUT_SIZE];
89              p1 = buf;
90              pend = buf + OUT_SIZE;
91          }
92          void out(char ch) {
93              if (p1 == pend) {
94                  fwrite(buf, 1, OUT_SIZE, stdout);
95                  p1 = buf;
96              }
97              *p1++ = ch;
98          }
99          void print(int x) {
100             static char s[15], *s1;
101             s1 = s;
102             if (!x)*s1++ = '0';
103             if (x < 0)out('-'), x = -x;
104             while (x)*s1++ = x % 10 + '0', x /= 10;
105             while (s1-- != s)out(*s1);
106         }
107         void println(int x) {
108             static char s[15], *s1;
109             s1 = s;
110             if (!x)*s1++ = '0';
```

```
111             if (x < 0)out('-'), x = -x;
112             while (x)*s1++ = x % 10 + '0', x /= 10;
113             while (s1-- != s)out(*s1);
114             out('\n');
115         }
116         void print(ll x) {
117             static char s[25], *s1;
118             s1 = s;
119             if (!x)*s1++ = '0';
120             if (x < 0)out('-'), x = -x;
121             while (x)*s1++ = x % 10 + '0', x /= 10;
122             while (s1-- != s)out(*s1);
123         }
124         void println(ll x) {
125             static char s[25], *s1;
126             s1 = s;
127             if (!x)*s1++ = '0';
128             if (x < 0)out('-'), x = -x;
129             while (x)*s1++ = x % 10 + '0', x /= 10;
130             while (s1-- != s)out(*s1);
131             out('\n');
132         }
133         void print(double x, int y) {
134             static ll mul[] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000,
    100000000,
135                                1000000000, 10000000000LL, 100000000000LL, 1000000000000
    LL, 10000000000000LL,
136                                100000000000000LL, 1000000000000000LL, 10000000000000000
    LL, 100000000000000000LL};
137             if (x < -1e-12)out('-'), x = -x;
138             x *= mul[y];
139             ll x1 = (ll) floor(x);
140             if (x - floor(x) >= 0.5)++x1;
141             ll x2 = x1 / mul[y], x3 = x1 - x2 * mul[y];
142             print(x2);
143             if (y > 0) {
144                 out('.');
145                 for (size_t i = 1; i < y && x3 * mul[i] < mul[y]; out('0'), ++i);
146                 print(x3);
147             }
148         }
149         void println(double x, int y) {
150             print(x, y);
151             out('\n');
152         }
153         void print(char *s) { while (*s)out(*s++); }
154         void println(char *s) {
155             while (*s)out(*s++);
156             out('\n');
157         }
158         void flush() {
159             if (p1 != buf) {
160                 fwrite(buf, 1, p1 - buf, stdout);
161                 p1 = buf;
162             }
163         }
164         ~Ostream_fwrite() { flush(); }
165     } Ostream;
166     inline void print(int x) { Ostream.print(x); }
```

```
    inline void println(int x) { Ostream.println(x); }
    inline void print(char x) { Ostream.out(x); }
    inline void println(char x) {
        Ostream.out(x);
        Ostream.out('\n');
    }
    inline void print(ll x) { Ostream.print(x); }
    inline void println(ll x) { Ostream.println(x); }
    inline void print(double x, int y) { Ostream.print(x, y); }
    inline void println(double x, int y) { Ostream.println(x, y); }
    inline void print(char *s) { Ostream.print(s); }
    inline void println(char *s) { Ostream.println(s); }
    inline void println() { Ostream.out('\n'); }
    inline void flush() { Ostream.flush(); }
};
using namespace FastIO;
```

7.3 InputOutputSpeedUp

```
//适用于正负整数
template <class T>
inline bool scan_d (T &ret) {
    char c; int sgn;
    if( c = getchar(), c == EOF)    return 0; //EOF
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    sgn = (c == '-') ? -1 : 1;
    ret = (c == '-') ? 0 : (c - '0');
    while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
    ret *= sgn;
    return 1;
}
//适用于正负整数
template <class T>
inline void outU (T x) {
    if (x < 0)  putchar('-'), x = -x;
    if (x > 9) out (x / 10);
    putchar (x % 10 + '0');
}
```

7.4 gcd

```
ll gcd(ll x, ll y) { // 循环版
    ll t;
    while (y){
        t = x % y;
        x = y;
        y = t;
    }
    return  x;
}

ll gcd(ll a, ll b) { // 递归版
    return b == 0 ? a : gcd(b, a % b);
}

// 扩展欧几里得
ll exgcd(ll a, ll b, ll &x, ll &y) {
```

```
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    ll q = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return q;
}
```

7.5 myItoa

```
char * myItoa(int value, char* result, int base = 10);

char * myItoa(int value, char* result, int base) {
    // check that the base if valid

    if (base < 2 || base > 16) { *result = 0; return result; }
    char* out = result;
    int quotient = abs(value);
    do {
        const int tmp = quotient / base;
        *out = "0123456789abcdef"[quotient - (tmp*base)];
        ++out;
        quotient = tmp;
    } while (quotient);
    // Apply negative sign
    if (value < 0) *out++ = '-';
    std::reverse(result, out);
    *out = 0;
    return result;
}
```

7.6 Permutation

```
// 错排问题
// D(n) = n! [(-1)^2/2! + ··· + (-1)^(n-1)/(n-1)! + (-1)^n/n!].
long long table[1000] = {0, 0, 1};
void init() {
    for (int i = 3; i <= 20; i++) {
        table[i] = (i - 1) * (table[i - 1] + table[i - 2]);
    }
}
```

7.7 prime

```
#define prime_max 1000000

int prime_count = 0;
bool prime_list[prime_max] = { false };//元素值为0代表是素数
int prime_table[prime_max] = { 0 };

void initPrime() {
    for (int i = 2; i < prime_max; i++) {
        if (!prime_list[i])
            prime_table[prime_count++] = i;
        for (int j = 0, e = prime_max / i;
```

```
12              j < prime_count && prime_table[j] <= e; j++) {
13             prime_list[i * prime_table[j]] = 1;
14             if (i % prime_table[j] == 0) break;
15         }
16     }
17 }
```