ACM/ICPC Template Manaual

ZheJiang GongShang University

Happy Otaku

March 20, 2019

Contents

0 头文件

```cpp
// 巨菜的ACMer-Happy233

#include <bits/stdc++.h>

using namespace std;

//-----
typedef double db;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#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))

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

0.1 stdc++

```cpp
// C++ includes used for precompiling -*- C++ -*-

// Copyright (C) 2003-2017 Free Software Foundation, Inc.
//
// This file is part of the GNU ISO C++ Library.  This library is free
// software; you can redistribute it and/or modify it under the
// terms of the GNU General Public License as published by the
// Free Software Foundation; either version 3, or (at your option)
// any later version.

// This library is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// Under Section 7 of GPL version 3, you are granted additional
// permissions described in the GCC Runtime Library Exception, version
// 3.1, as published by the Free Software Foundation.

// You should have received a copy of the GNU General Public License and
// a copy of the GCC Runtime Library Exception along with this program;
// see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see
// <http://www.gnu.org/licenses/>.

/** @file stdc++.h
 *  This is an implementation file for a precompiled header.
 */

// 17.4.1.2 Headers

// 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

// 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 NEXT(elemType &T, vector<int> &next) {
    next[0] = -1;
    for (int i = 1; i < T.size(); i++) {
        int j = next[i - 1];
        while (j >= 0 && T[i - 1] != T[j]) j = next[j];
        if (j >= 0 && T[i - 1] == T[j]) next[i] = j + 1;
        else next[i] = 0;
    }
}

template<class elemType>
inline int COUNT_KMP(elemType &S, elemType &T) {
    vector<int> next(T.size());
    NEXT(T, next);
    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 = next[pos - 1] + 1;
            }
        }
        if (pos == T.size() && (iter - index) == T.size()) ++count;
    }
    return count;
}

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

template<class elemType>
inline int COUNT_KMP(elemType S[], int c1, elemType T[], int c2) {
    vector<int> next(c2);
    NEXT(T, c2, next);
    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;
                ++pos;
            }
            else {
                if (pos == 0) ++iter;
                else pos = next[pos - 1] + 1;
            }
        }
        if (pos == c2 && (iter - index) == c2) ++count;
    }
    return count;
}
```

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){
    return x>y?x:y;
}
int main()  {
    int t,n,m,i,j,k;
    int w[N],pri[N],num[N],f[N];
    while(~scanf("%d",&t)){
        while(t--){
            memset(f,0,sizeof(f));
            scanf("%d%d",&n,&m);//n为总金额，m为大米种类
            for(i = 0 ; i < m ; i++){
                scanf("%d%d%d",&pri[i],&w[i],&num[i]);//num[i]为每种大米的袋数
            }
            for(i = 0 ; i < m ; i++){
                for(k = 0 ; k < num[i] ; k++){
                    for(j = n ; j >= pri[i]; j--){
                        f[j] = max(f[j],f[j-pri[i]]+w[i]);
                    }
                }
            }
            printf("%d\n",f[n]);
        }
    }
    return 0;
}
```

3 数据结构

3.1 BTree

```cpp
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;
    }
}

```

```
template<class T>
void postOrder(TreeNode<T> *root, queue<T> &out) {
    if (root) {
        postOrder(root->left, out);
        postOrder(root->right, out);
        out.push(root->value);
    } else {
        return;
    }
}

template<class T>
T *convertQueueToArray(queue<T> &out, int len) {
    T *list = new T[len];
    int now = 0;
    while (!out.empty() && now < len) {
        list[now] = out.front();
        out.pop();
        now++;
    }
    return list;
}

template<class T>
void destroyTree(TreeNode<T> *root) {
    if (root) {
        destroyTree(root->left);
        destroyTree(root->right);
        delete root;
    } else return;
}

template<class T>
void insertIntoBSTree(TreeNode<T> *root, const T &value) {
    if (!root) {
        return;
    }
    if (value < root->value) {
        if (root->left) {
            insertIntoTree(root->left, value);
        } else {
            root->left = new TreeNode<T>;
            root->left->value = value;
            root->left->left = NULL;
            root->left->right = NULL;
        }
    } else if (value > root->value) {
        if (root->right) {
            insertIntoTree(root->right, value);
        } else {
            root->right = new TreeNode<T>;
            root->right->value = value;
            root->right->left = NULL;
            root->right->right = NULL;
        }
    }
}

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

```
// 红黑树
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
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);
```

3.3 树状数组

```
const int N = 50005;

struct BITree {
    int n;
    ll c[N * 2];

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

    int change(int pos, ll v) {
        for (int i = pos; i <= n; i += i & (-i))
            c[i] += v;
        return 0;
    }

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

3.4 二维树状数组

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

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

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

    int change(int x, int y, ll v) {
        p[x][y] += v;
        for (int i = x; i <= n; i += lowbit(i))
            for (int j = y; j <= m; j += lowbit(j))
                c[i][j] += v;
        return 0;
    }

    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) {
            return l == t.l && r == t.r;
        }
    };

    ll c[N << 2][N << 2];
    ll ans[N << 4];
    ll laz[N << 4];

    inline void up(int k, bool x, bool y) {
        int s = (k << 2) - 2;
        ll t = 0;
        if (x) t += ans[s] + ans[s + 1] + laz[s] + laz[s + 1];
        if (y) t += ans[s + 2] + ans[s + 3] + laz[s + 2] + laz[s + 3];
        ans[k] = t;
    }

    inline void push(int k) {
        int s = (k << 2) - 2;
        laz[s] += laz[k];
        laz[s + 1] += laz[k];
        laz[s + 2] += laz[k];
        laz[s + 3] += laz[k];
        ans[k] += laz[k];
        laz[k] = 0;
    }

    void build(node x, node y, int k) {
        laz[k] = 0;
        if (x.more() && y.more()) {
            ans[k] = c[x.l][y.l];
            return;
        }
        ans[k] = 0;
        bool ax = false;
        bool ay = false;
        if (x.more()) {
            build(x.left(), y, son(k, 0));
            build(x.right(), y, son(k, 1));
        }
        if (y.more()) {
            build(x, y.left(), son(k, 2));
            build(x, y.right(), son(k, 3));
        }
        up(k, x.more(), y.more());
    }

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

4 图论

4.1 Dijkstra

```cpp
#define INF 10000000

struct node {
    int x, d;
    node() = default;
    node(int a, int b): x(a), d(b) {}
    bool operator<(const node &a) const {
        return (a.d == d ? x < a.x : d < a.d);
    }
};

#define N_MAX 150
vector<node> eg[N_MAX];
int dist[N_MAX];
vector<int> path[N_MAX];

void dfs(int s, int n) {
    for (int i = 1; i <= n; i++) {
        dist[i] = INF;
    }
    dist[s] = 0;
    priority_queue<node> list;
    list.push(node(s, dist[s]));
    path[s].push_back(s);
    while (!list.empty()) {
        node x = list.top();
        list.pop();
        for(const auto &y: eg[x.x]){
            if (dist[y.x] > x.d + y.d) {
                dist[y.x] = x.d + y.d;
                path[y.x] = path[x.x];
                path[y.x].push_back(y.x);
                list.push(node(y.x, dist[y.x]));
            }
        }
    }
}
```

4.2 spfa

```cpp
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;
}
```

4.3 Dinic

```
#define N 2005
#define INF 0x7fffffff

struct dinic {

    struct node {
        int e;
        ll f;

        node() = default;

        node(int a, ll b) : e(a), f(b) {}
    };

    // 点的范围[0, n)
    int n;
    vector<node> eg;
    vector<int> head[N];
    // 弧优化
    int cur[N], dis[N];

    dinic() = default;

    // 设置N
    void setN(int n) {
        this->n = n;
    }

    inline void addEdge(int x, int y, ll f) {
        //printf("%d->%d: %lld\n", x, y, f);
        head[x].push_back(static_cast<int &&>(eg.size()));
        eg.push_back({y, f});
    }

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

    bool bfs(int s, int e) {
        fill_n(dis, n, -1);
        int q[N];
        int l, r;
        l = r = 0;
        dis[s] = 0;
        q[r++] = s;
        while (l < r) {
            int f = q[l++];
            for (const auto &i: head[f]) {
                if (eg[i].f > 0 && dis[eg[i].e] == -1) {
                    dis[eg[i].e] = dis[f] + 1;
                    q[r++] = eg[i].e;
                }
            }
        }
        return dis[e] > 0;
    }

    ll dfs(int s, int e, ll mx) {
        if (s == e || mx == 0) {
            return mx;
        }
        int flow = 0;
        for (int &k = cur[s]; k < head[s].size(); k++) {
            int &i = head[s][k];
            auto &te = eg[i];
            ll a;
            if (te.f > 0 && dis[te.e] == dis[s] + 1 && (a = dfs(te.e, e, min(te.f, mx))
    )) {
                te.f -= a;
                eg[i ^ 1].f += 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)) {
            fill_n(cur, n, 0);
            ans += dfs(s, e, INF);
        }
        return ans;
    }

    // 清空数据
    void clear() {
        rep(i, 0, n) head[i].clear();
        eg.clear();
    }
};
```

4.4 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);
}
```

4.5 MinSpanTree

```
#define INF 1000000
#define N 100000
ll lowcost[N]; //此数组用来记录第j个节点到其余节点最少花费
ll mpp[N][N]; //用来记录第i个节点到其余n-1个节点的距离
int vis[N]; //用来记录最小生成树中的节点
ll city;

ll prim() {
    ll min, i, j, next = 0, mincost = 0;
    memset(vis, 0, sizeof(vis)); //给最小生成树数组清零
    for (i = 1; i <= city; i++) {
        lowcost[i] = mpp[1][i]; //初始化lowcost数组为第1个节点到剩下所有节点的距离
    }
    vis[1] = 1; //选择第一个点为最小生成树的起点
    for (i = 1; i < city; i++) {
        min = INF;
        for (j = 1; j <= city; j++) {
            if (!vis[j] && min > lowcost[j]) { //如果第j个点不是最小生成树中的点并且其花费小于min
                min = lowcost[j];
                next = j;//记录下此时最小的位置节点
            }
        }
        if (min == INF) return INF;
        mincost += min; //将最小生成树中所有权值相加
        vis[next] = 1; //next点加入最小生成树
        for (j = 1; j <= city; j++) {
            if (!vis[j] && lowcost[j] > mpp[next][j]) { //如果第j点不是最小生成树中的点并且此点处权值大于第next点到j点的权值
                lowcost[j] = mpp[next][j];  //更新lowcost数组
            }
        }
    }
    return mincost;
}
```

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;
            }
        }
    }
}
```

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;
}
```

```

Mat operator ^ (Mat a, int k) {
    long long n = a.n;
    Mat c(n);
    c.init();
    for (; k; k >>= 1) {
        if (k & 1)   c = c * a;
        a = a * a;
    }
    return c;
}
```

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(long long v) { *this = v; }

    bigint &operator=(long long 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 {
                *this = other - *this;
                this->sign = -this->sign;
            }
        } else {
            *this += -other;
        }
        return *this;
    }

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

    bigint &operator*=(int v) {
        if (v < 0)
            sign = -sign, v = -v;
        for (int i = 0, carry = 0; i < z.size() || carry; ++i) {
            if (i == z.size())
                z.push_back(0);
            long long cur = (long long) z[i] * v + carry;
            carry = (int) (cur / base);
            z[i] = (int) (cur % base);
        }
        trim();
        return *this;
    }

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

    friend pair<bigint, bigint> divmod(const bigint &a1, const bigint &b1) {
        int norm = base / (b1.z.back() + 1);
        bigint a = a1.abs() * norm;
        bigint b = b1.abs() * norm;
        bigint q, r;
        q.z.resize(a.z.size());

        for (int i = (int) a.z.size() - 1; i >= 0; i--) {
            r *= base;
            r += a.z[i];
            int s1 = b.z.size() < r.z.size() ? r.z[b.z.size()] : 0;
            int s2 = b.z.size() - 1 < r.z.size() ? r.z[b.z.size() - 1] : 0;
            int d = (int) (((long long) s1 * base + s2) / b.z.back());
            r -= b * d;
            while (r < 0)
                r += b, --d;
            q.z[i] = d;
        }

        q.sign = a1.sign * b1.sign;
        r.sign = a1.sign;
        q.trim();
        r.trim();
        return {q, r / norm};
```

```cpp
    }

    friend bigint sqrt(const bigint &a1) {
        bigint a = a1;
        while (a.z.empty() || a.z.size() % 2 == 1)
            a.z.push_back(0);

        int n = a.z.size();

        int firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
        int norm = base / (firstDigit + 1);
        a *= norm;
        a *= norm;
        while (a.z.empty() || a.z.size() % 2 == 1)
            a.z.push_back(0);

        bigint r = (long long) a.z[n - 1] * base + a.z[n - 2];
        firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
        int q = firstDigit;
        bigint res;

        for (int j = n / 2 - 1; j >= 0; j--) {
            for (;; --q) {
                bigint r1 = (r - (res * 2 * base + q) * q) * base * base +
                            (j > 0 ? (long long) a.z[2 * j - 1] * base + a.z[2 * j - 2]
 : 0);
                if (r1 >= 0) {
                    r = r1;
                    break;
                }
            }
            res *= base;
            res += q;

            if (j > 0) {
                int d1 = res.z.size() + 2 < r.z.size() ? r.z[res.z.size() + 2] : 0;
                int d2 = res.z.size() + 1 < r.z.size() ? r.z[res.z.size() + 1] : 0;
                int d3 = res.z.size() < r.z.size() ? r.z[res.z.size()] : 0;
                q = (int) (((long long) d1 * base * base + (long long) d2 * base + d3)
/ (firstDigit * 2));
            }
        }

        res.trim();
        return res / norm;
    }

    bigint operator/(const bigint &v) const {
        return divmod(*this, v).first;
    }

    bigint operator%(const bigint &v) const {
        return divmod(*this, v).second;
    }

    bigint &operator/=(int v) {
        if (v < 0)
            sign = -sign, v = -v;
        for (int i = (int) z.size() - 1, rem = 0; i >= 0; --i) {
            long long cur = z[i] + rem * (long long) base;
            z[i] = (int) (cur / v);
            rem = (int) (cur % v);
        }
        trim();
        return *this;
    }

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

    int operator%(int v) const {
        if (v < 0)
            v = -v;
        int m = 0;
        for (int i = (int) z.size() - 1; i >= 0; --i)
            m = (int) ((z[i] + m * (long long) base) % v);
        return m * sign;
    }

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

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

    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;
    }

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

    friend bigint gcd(const bigint &a, const bigint &b) {
        return b.isZero() ? a : gcd(b, a % b);
    }

    friend bigint lcm(const bigint &a, const bigint &b) {
        return a / gcd(a, b) * b;
    }

    void read(const string &s) {
        sign = 1;
        z.clear();
        int pos = 0;
        while (pos < s.size() && (s[pos] == '-' || s[pos] == '+')) {
            if (s[pos] == '-')
                sign = -sign;
            ++pos;
        }
        for (int i = (int) s.size() - 1; i >= pos; i -= base_digits) {
            int x = 0;
            for (int j = max(pos, i - base_digits + 1); j <= i; j++)
                x = x * 10 + s[j] - '0';
            z.push_back(x);
        }
        trim();
    }

    friend istream &operator>>(istream &stream, bigint &v) {
        string s;
        stream >> s;
        v.read(s);
        return stream;
    }

    friend ostream &operator<<(ostream &stream, const bigint &v) {
        if (v.sign == -1)
            stream << '-';
        stream << (v.z.empty() ? 0 : v.z.back());
        for (int i = (int) v.z.size() - 2; i >= 0; --i)
            stream << setw(base_digits) << setfill('0') << v.z[i];
        return stream;
    }

    static vector<int> convert_base(const vector<int> &a, int old_digits, int
    new_digits) {
        vector<long long> p(max(old_digits, new_digits) + 1);
        p[0] = 1;
        for (int i = 1; i < p.size(); i++)
            p[i] = p[i - 1] * 10;
        vector<int> res;
        long long cur = 0;
        int cur_digits = 0;
        for (int v : a) {
            cur += v * p[cur_digits];
            cur_digits += old_digits;
            while (cur_digits >= new_digits) {
                res.push_back(int(cur % p[new_digits]));
                cur /= p[new_digits];
                cur_digits -= new_digits;
            }
        }
        res.push_back((int) cur);
        while (!res.empty() && res.back() == 0)
            res.pop_back();
        return res;
    }

    typedef vector<long long> 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++) {
            long long 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
```

```cpp
    //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) {
        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 (ch == '.') {
            double tmp = 1;
            ch = nc();
            for (; ch >= '0' && ch <= '9'; ch = nc())tmp /= 10.0, x += tmp * (ch - '0');
        }
        if (sign)x = -x;
    }

    inline void read(char *s) {
        char ch = nc();
        for (; blank(ch); ch = nc());
        if (IOerror)return;
        for (; !blank(ch) && !IOerror; ch = nc())*s++ = ch;
        *s = 0;
    }

    inline void read(char &c) {
        for (c = nc(); blank(c); c = nc());
        if (IOerror) {
            c = -1;
            return;
        }
    }

    //fwrite->write
    struct Ostream_fwrite {
        char *buf, *p1, *pend;

        Ostream_fwrite() {
            buf = new char[OUT_SIZE];
            p1 = buf;
            pend = buf + OUT_SIZE;
        }

        void out(char ch) {
            if (p1 == pend) {
                fwrite(buf, 1, OUT_SIZE, stdout);
                p1 = buf;
            }
            *p1++ = ch;
        }

        void print(int x) {
            static char s[15], *s1;
            s1 = s;
            if (!x)*s1++ = '0';
            if (x < 0)out('-'), x = -x;
            while (x)*s1++ = x % 10 + '0', x /= 10;
            while (s1-- != s)out(*s1);
        }

        void println(int x) {
            static char s[15], *s1;
            s1 = s;
            if (!x)*s1++ = '0';
            if (x < 0)out('-'), x = -x;
            while (x)*s1++ = x % 10 + '0', x /= 10;
            while (s1-- != s)out(*s1);
            out('\n');
        }

        void print(ll x) {
            static char s[25], *s1;
            s1 = s;
            if (!x)*s1++ = '0';
            if (x < 0)out('-'), x = -x;
            while (x)*s1++ = x % 10 + '0', x /= 10;
            while (s1-- != s)out(*s1);
        }

        void println(ll x) {
            static char s[25], *s1;
            s1 = s;
            if (!x)*s1++ = '0';
            if (x < 0)out('-'), x = -x;
            while (x)*s1++ = x % 10 + '0', x /= 10;
            while (s1-- != s)out(*s1);
            out('\n');
        }

        void print(double x, int y) {
            static ll mul[] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000,
                               1000000000, 10000000000LL, 100000000000LL, 1000000000000LL, 10000000000000LL,
                               100000000000000LL, 1000000000000000LL, 10000000000000000LL, 100000000000000000LL};
            if (x < -1e-12)out('-'), x = -x;
            x *= mul[y];
            ll x1 = (ll) floor(x);
            if (x - floor(x) >= 0.5)++x1;
            ll x2 = x1 / mul[y], x3 = x1 - x2 * mul[y];
            print(x2);
            if (y > 0) {
                out('.');
                for (size_t i = 1; i < y && x3 * mul[i] < mul[y]; out('0'), ++i);
                print(x3);
            }
        }

        void println(double x, int y) {
            print(x, y);
            out('\n');
        }

        void print(char *s) { while (*s)out(*s++); }

        void println(char *s) {
            while (*s)out(*s++);
            out('\n');
        }

        void flush() {
            if (p1 != buf) {
                fwrite(buf, 1, p1 - buf, stdout);
                p1 = buf;
            }
        }

        ~Ostream_fwrite() { flush(); }
    } Ostream;

    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;
}
inline void out (int x) {
    if (x < 0) {
        putchar('-');
        x = -x;
    }
    if (x > 9) out (x / 10);
    putchar (x % 10 + '0');
}
inline void out(int x) {
    if (x < 0) {
        putchar('-');
        x = -x;
    }
    char list[100];
    int now = 0;
    while (x > 9) {
        list[++now] = (x % 10 + '0');
        x /= 10;
    }
    putchar(x + '0');
    while (now) {
        putchar(list[now--]);
    }
}
```

7.4 gcd

```
int gcd(int x, int y) {
    int t;
    while (y){
        t = x % y;
        x = y;
        y = t;
    }
    return  x;
}
```

7.5 IntFastSqrt

```
int sqrtI(int t){
    int min = 0;
    int max = t;
    while (min != max && min + 1 != max) {
        int mid = (min + max) / 2;
        int a = mid * mid;
        if (a == t) {
            return mid;
        }
        else if (a > t) {
            max = mid;
        }
        else if (a < t) {
            min = mid;
        }
    }
    return min;
}
```

7.6 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;
}
```

7.7 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; j < prime_count && i*prime_table[j] < prime_max; j++){
            prime_list[i*prime_table[j]] = 1;
            if (i % prime_table[j] == 0) break;
        }
    }
}
```

7.8 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]);
    }
}
```

7.9 reverseInt

```
template<typename T>

T reverseInt(T a)
{
    T b = 0;
    while (a != 0) {
        b *= 10;
        b += (a % 10);
        a /= 10;
    }
    return b;
}
```