



---

# ACM/ICPC Template Manual

---

**ZheJiang GongShang University**

Happy Otaku

March 20, 2019

## Contents

<b>0</b>	<b>头文件</b>	<b>1</b>
0.1	stdc++ . . . . .	1
<b>1</b>	<b>字符串</b>	<b>4</b>
1.1	KMP . . . . .	4
<b>2</b>	<b>动态规划</b>	<b>6</b>
2.1	01Bag . . . . .	6
2.2	BagProblem . . . . .	6
2.3	FullBag . . . . .	6
2.4	MultiBag . . . . .	7
<b>3</b>	<b>数据结构</b>	<b>9</b>
3.1	BTree . . . . .	9
3.2	pbds-bbtree . . . . .	11
3.3	树状数组 . . . . .	11
3.4	二维树状数组 . . . . .	12
3.5	线段树 . . . . .	13
3.6	二维线段树 . . . . .	14
3.7	树状数组求逆序对 . . . . .	16
<b>4</b>	<b>图论</b>	<b>17</b>
4.1	Dijkstra . . . . .	17
4.2	spfa . . . . .	17
4.3	Dinic . . . . .	18
4.4	hungry . . . . .	20
4.5	MinSpanTree . . . . .	21
<b>5</b>	<b>博弈</b>	<b>22</b>
5.1	GameProblem . . . . .	22
<b>6</b>	<b>分治</b>	<b>23</b>
6.1	IntegerFastPower . . . . .	23
6.2	MatrixFastPower . . . . .	23
<b>7</b>	<b>其他</b>	<b>25</b>
7.1	BigInteger . . . . .	25
7.2	FastIO . . . . .	31
7.3	InputOutputSpeedUp . . . . .	35
7.4	gcd . . . . .	36
7.5	IntFastSqrt . . . . .	36
7.6	myItoa . . . . .	36
7.7	prime . . . . .	37
7.8	Permutation . . . . .	37
7.9	reverseInt . . . . .	37

## 0 头文件

```
1 // 巨菜的ACMer-Happy233
2
3 #include <bits/stdc++.h>
4
5 using namespace std;
6
7 //-----
8 typedef double db;
9 typedef long long ll;
10 typedef vector<int> vi;
11 typedef pair<int, int> pii;
12 typedef pair<ll, ll> pll;
13 #define fi first
14 #define se second
15 #define mp make_pair
16 #define pb push_back
17 #define pw(x) (1ll << (x))
18 #define sz(x) ((int)(x).size())
19 #define all(x) (x).begin(),(x).end()
20 #define rep(i, l, r) for(int i=l;i<(r);++i)
21 #define per(i, l, r) for(int i=(r)-1;i>=l;--i)
22 #define sf(x) scanf("%d", &(x))
23
24 const double pi = acos(-1);
```

### 0.1 stdc++

```
1 // C++ includes used for precompiling -*- C++ -*-
2
3 // Copyright (C) 2003-2017 Free Software Foundation, Inc.
4 //
5 // This file is part of the GNU ISO C++ Library. This library is free
6 // software; you can redistribute it and/or modify it under the
7 // terms of the GNU General Public License as published by the
8 // Free Software Foundation; either version 3, or (at your option)
9 // any later version.
10
11 // This library is distributed in the hope that it will be useful,
12 // but WITHOUT ANY WARRANTY; without even the implied warranty of
13 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 // GNU General Public License for more details.
15
16 // Under Section 7 of GPL version 3, you are granted additional
17 // permissions described in the GCC Runtime Library Exception, version
18 // 3.1, as published by the Free Software Foundation.
19
20 // You should have received a copy of the GNU General Public License and
21 // a copy of the GCC Runtime Library Exception along with this program;
22 // see the files COPYING3 and COPYING.RUNTIME respectively. If not, see
23 // <http://www.gnu.org/licenses/>.
24
25 /** @file stdc++.h
26  * This is an implementation file for a precompiled header.
27  */
28
29 // 17.4.1.2 Headers
```

```
30
31 // C
32 #ifndef _GLIBCXX_NO_ASSERT
33 #include <cassert>
34 #endif
35 #include <cctype>
36 #include <cerrno>
37 #include <cfloat>
38 #include <ciso646>
39 #include <climits>
40 #include <locale>
41 #include <cmath>
42 #include <csetjmp>
43 #include <csignal>
44 #include <cstdarg>
45 #include <cstddef>
46 #include <cstdio>
47 #include <cstdlib>
48 #include <cstring>
49 #include <ctime>
50
51 #if __cplusplus >= 201103L
52 #include <complex>
53 #include <cfenv>
54 #include <cinttypes>
55 #include <cstdalign>
56 #include <cstdbool>
57 #include <cstdint>
58 #include <ctgmath>
59 #include <cuchar>
60 #include <wchar>
61 #include <cwctype>
62 #endif
63
64 // C++
65 #include <algorithm>
66 #include <bitset>
67 #include <complex>
68 #include <deque>
69 #include <exception>
70 #include <fstream>
71 #include <functional>
72 #include <iomanip>
73 #include <ios>
74 #include <iosfwd>
75 #include <iostream>
76 #include <istream>
77 #include <iterator>
78 #include <limits>
79 #include <list>
80 #include <locale>
81 #include <map>
82 #include <memory>
83 #include <new>
84 #include <numeric>
85 #include <ostream>
86 #include <queue>
87 #include <set>
88 #include <sstream>
```

```
89 #include <stack>
90 #include <stdexcept>
91 #include <streambuf>
92 #include <string>
93 #include <typeinfo>
94 #include <utility>
95 #include <valarray>
96 #include <vector>
97
98 #if __cplusplus >= 201103L
99 #include <array>
100 #include <atomic>
101 #include <chrono>
102 #include <codecvt>
103 #include <condition_variable>
104 #include <forward_list>
105 #include <future>
106 #include <initializer_list>
107 #include <mutex>
108 #include <random>
109 #include <ratio>
110 #include <regex>
111 #include <scoped_allocator>
112 #include <system_error>
113 #include <thread>
114 #include <tuple>
115 #include <typeindex>
116 #include <type_traits>
117 #include <unordered_map>
118 #include <unordered_set>
119 #endif
120
121 #if __cplusplus >= 201402L
122 #include <shared_mutex>
123 #endif
```

# 1 字符串

## 1.1 KMP

```

1  template<class elemType>
2  inline void NEXT(elemType &T, vector<int> &next) {
3      next[0] = -1;
4      for (int i = 1; i < T.size(); i++) {
5          int j = next[i - 1];
6          while (j >= 0 && T[i - 1] != T[j]) j = next[j];
7          if (j >= 0 && T[i - 1] == T[j]) next[i] = j + 1;
8          else next[i] = 0;
9      }
10 }
11
12 template<class elemType>
13 inline int COUNT_KMP(elemType &S, elemType &T) {
14     vector<int> next(T.size());
15     NEXT(T, next);
16     int index, count = 0;
17     for (index = 0; index < S.size(); ++index) {
18         int pos = 0;
19         int iter = index;
20         while (pos < T.size() && iter < S.size()) {
21             if (S[iter] == T[pos]) {
22                 ++iter;
23                 ++pos;
24             } else {
25                 if (pos == 0) ++iter;
26                 else pos = next[pos - 1] + 1;
27             }
28         }
29         if (pos == T.size() && (iter - index) == T.size()) ++count;
30     }
31     return count;
32 }
33
34 template<class elemType>
35 inline void NEXT(elemType T[], int count, vector<int> &next) {
36     next[0] = -1;
37     for (int i = 1; i < count; i++) {
38         int j = next[i - 1];
39         while (j >= 0 && T[i - 1] != T[j]) j = next[j];
40         if (j >= 0 && T[i - 1] == T[j]) next[i] = j + 1;
41         else next[i] = 0;
42     }
43 }
44
45 template<class elemType>
46 inline int COUNT_KMP(elemType S[], int c1, elemType T[], int c2) {
47     vector<int> next(c2);
48     NEXT(T, c2, next);
49     int index, count = 0;
50     for (index = 0; index < c1; ++index) {
51         int pos = 0;
52         int iter = index;
53         while (pos < c2 && iter < c1) {
54             if (S[iter] == T[pos]) {
55                 ++iter;

```

```
56         ++pos;
57     }
58     else {
59         if (pos == 0) ++iter;
60         else pos = next[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

```

1 void dp(int n, int m) {
2     // n=物品个数
3     for (int i = 0; i < n; i++) {
4         // m=背包最大容量
5         for (int j = m; j >= wei[i]; j--)
6             // wei=大小 val=价值
7             f[j] = max(f[j], f[j - wei[i]] + val[i]);
8     }
9 }

```

### 2.2 BagProblem

```

1 #define N 1000
2 // val=价值 wei=重量 num=数量
3 int val[N], wei[N], num[N], f[N];
4 // n=种类个数 m=背包最大值
5
6 // 01背包
7 void dp1(int n, int m) {
8     for (int i = 0; i < n; i++) {
9         for (int j = m; j >= wei[i]; j--)
10             f[j] = max(f[j], f[j - wei[i]] + val[i]);
11     }
12 }
13
14 // 完全背包
15 void dp2(int n, int m) {
16     //初始化看要求
17     for (int i = 0; i <= m; i++) {
18         f[i] = INF;
19     }
20     f[0] = 0;
21     //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=-∞
22     //若没要求背包装满, 只希望价格大, 初始化时应将f[0..V]=0)
23     for (int i = 0; i < n; i++)
24         for (int j = wei[i]; j <= m; j++)
25             f[j] = max(f[j], f[j - wei[i]] + val[i]);
26 }
27
28 // 多重背包
29 void dp3(int n, int m) {
30     for (int i = 0; i < n; i++)
31         for (int k = 0; k < num[i]; k++)
32             for (int j = m; j >= wei[i]; j--)
33                 f[j] = max(f[j], f[j - wei[i]] + val[i]);
34 }

```

### 2.3 FullBag

```

1 /*
2 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。
3 完全背包问题描述:
4 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]。

```



```

5  */
6
7  #include <stdio>
8  #define INF 0x3fffffff
9  #define N 10047
10 int f[N],val[N],wei[N];
11 int min(int a,int b)
12 {
13     return x<y?x:y;
14 }
15 int main()
16 {
17     int t,i,j,k,E,F,m,n;
18     scanf("%d",&t);
19     while(t-->0)
20     {
21         scanf("%d%d",&E,&F);
22         int c = F-E;
23         for(i = 0 ; i <= c ; i++)
24             f[i]=INF;
25         scanf("%d",&n);
26         for(i = 0 ; i < n ; i++)
27         {
28             scanf("%d%d",&val[i],&wei[i]); //val[i]为面额, wei[i]为重量
29         }
30         f[0]=0; //因为此处假设的是小猪储钱罐 恰好装满 的情况
31         //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-∞,
32         //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。
33         //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)
34         for(i = 0 ; i < n ; i++)
35         {
36             for(j = wei[i] ; j <= c ; j++)
37             {
38                 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
39                 //的钱,当然最后就用max了, HEHE
40             }
41         }
42         if(f[c] == INF)
43             printf("This is impossible.\n");
44         else
45             printf("The minimum amount of money in the piggy-bank is %d.\n",f[c]);
46     }
47     return 0;
48 } //此代码为HDU1114;

```

## 2.4 MultiBag

```

1  //多重背包(MultiplePack): 有N种物品和一个容量为V的背包。
2  //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]。
3  //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,
4  //且价值总和最大。
5  //HDU 2191
6
7  #include <stdio>
8  #include <cstring>
9  #define N 247
10 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

```

1  template<class T>
2
3  struct TreeNode {
4      T value;
5      TreeNode *left;
6      TreeNode *right;
7  };
8
9  template<class T>
10 TreeNode<T> *createTree(const T *pre, const T *in, const int len) {
11     TreeNode<T> *t = NULL;
12     if (len > 0) {
13         t = new TreeNode<T>;
14         t->value = pre[0];
15         int index;
16         for (index = 0; index < len; index++) {
17             if (in[index] == pre[0]) {
18                 break;
19             }
20         }
21         if (index == len) {
22             index = -1;
23         }
24         t->left = createTree(pre + 1, in, index);
25         t->right = createTree(pre + index + 1, in + index + 1, len - index - 1);
26     }
27     return t;
28 }
29
30 template<class T>
31 int preOrder(TreeNode<T> *root, queue<T> &out) {
32     if (root) {
33         int count = 1;
34         out.push(root->value);
35         count += preOrder(root->left, out);
36         count += preOrder(root->right, out);
37         return count;
38     } else {
39         return 0;
40     }
41 }
42
43 template<class T>
44 int inOrder(TreeNode<T> *root, queue<T> &out) {
45     if (root) {
46         int count = 1;
47         count += inOrder(root->left, out);
48         out.push(root->value);
49         count += inOrder(root->right, out);
50         return count;
51     } else {
52         return 0;
53     }
54 }
55

```

```
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>
```

```

115 TreeNode<T> *createBSTree(T *list, int len) {
116     if (len < 1) {
117         return NULL;
118     }
119     TreeNode<T> *root = new TreeNode<char>;
120     root->value = list[0];
121     root->left = NULL;
122     root->right = NULL;
123     for (int i = 1; i < len; i++) {
124         insertIntoBSTree(root, list[i]);
125     }
126     return root;
127 }

```

### 3.2 pbds-bbtree

```

1 // 红黑树
2 tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update> t;
3 // null_type无映射(低版本g++为null_mapped_type)
4 // 类似multiset
5 tree<int, null_type, less_equal<int>, rb_tree_tag, tree_order_statistics_node_update> t;
6
7 find_by_order(size_t order);
8 // 结点更新
9 tree_order_statistics_node_update
10 insert(p);
11 erase(it);
12 // 求k在树中是第几大:
13 order_of_key(p);
14 // 找到第order小的迭代器
15 find_by_order(order);
16 // 前驱
17 lower_bound(p);
18 // 后驱
19 upper_bound(p);
20 // 合并
21 a.join(b);
22 // 分割 key小于等于v的元素属于a, 其余的属于b
23 a.split(v, b);

```

### 3.3 树状数组

```

1 const int N = 50005;
2
3 struct BITree {
4     int n;
5     ll c[N * 2];
6
7     void init(int n) {
8         memset(c, 0, sizeof(ll) * (n + 1));
9         this->n = n;
10    }
11
12    int change(int pos, ll v) {
13        for (int i = pos; i <= n; i += i & (-i))
14            c[i] += v;
15    }
16 }

```

```

15     return 0;
16 }
17
18 ll query(int x) {
19     ll ans = 0;
20     for (int i = x; i > 0; i -= i & (-i))
21         ans += c[i];
22     return ans;
23 }
24 };

```

### 3.4 二维树状数组

```

1 struct TdBITree {
2     int n, m;
3     ll c[N][N];
4     ll p[N][N];
5
6     void init(int n, int m) {
7         this->n = n;
8         this->m = m;
9         memset(c, 0, sizeof(c));
10        memset(p, 0, sizeof(p));
11    }
12
13    void init(int n, int m, ll v) {
14        this->n = n;
15        this->m = m;
16        rep(x, 1, N) {
17            rep(y, 1, N) {
18                p[x][y] = v;
19                c[x][y] = (x * y + (x - lowbit(x)) * (y - lowbit(y)) - x * (y - lowbit(
20                y)) - (x - lowbit(x)) * y) * v;
21            }
22        }
23
24        int change(int x, int y, ll v) {
25            p[x][y] += v;
26            for (int i = x; i <= n; i += lowbit(i))
27                for (int j = y; j <= m; j += lowbit(j))
28                    c[i][j] += v;
29            return 0;
30        }
31
32        ll query(int x, int y) {
33            ll ans = 0;
34            for (int i = x; i >= 1; i -= lowbit(i))
35                for (int j = y; j >= 1; j -= lowbit(j))
36                    ans += c[i][j];
37            return ans;
38        }
39
40        ll solve(int x1, int y1, int x2, int y2) {
41            return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) + query(x1 - 1, y1
42            - 1);
43        }
44    };

```

### 3.5 线段树

```

1  const int N = 50005;
2
3  struct SegTree {
4      ll c[N];
5      ll ans[N << 2];
6      ll laz[N << 2];
7
8      void init(int n) {
9          memset(c, 0, sizeof(ll) * (n + 1));
10     }
11
12     inline void up(int k) {
13         ans[k] = ans[k << 1] + laz[k << 1] + ans[k << 1 | 1] + laz[k << 1 | 1];
14     }
15
16     inline void push(int k) {
17         laz[k << 1] += laz[k];
18         laz[k << 1 | 1] += laz[k];
19         ans[k] += laz[k];
20         laz[k] = 0;
21     }
22
23     void build(int x, int y, int k) {
24         laz[k] = 0;
25         if (x == y) {
26             ans[k] = c[x];
27             return;
28         }
29         int m = (x + y) >> 1;
30         build(x, m, k << 1);
31         build(m + 1, y, k << 1 | 1);
32         up(k);
33     }
34
35     void change(int x, int y, int k, int l, int r, ll v) {
36         if (x == l && y == r) {
37             laz[k] += v;
38             return;
39         }
40         push(k);
41         int m = (x + y) >> 1;
42         if (r <= m) change(x, m, k << 1, l, r, v);
43         else if (l > m) change(m + 1, y, k << 1 | 1, l, r, v);
44         else change(x, m, k << 1, l, m, v), change(m + 1, y, k << 1 | 1, m + 1, r, v);
45         up(k);
46     }
47
48     ll query(int x, int y, int k, int l, int r) {
49         if (x == l && y == r) {
50             return ans[k] + laz[k];
51         }
52         int m = (x + y) >> 1;
53         push(k);
54         if (r <= m) return query(x, m, k << 1, l, r);
55         else if (l > m) return query(m + 1, y, k << 1 | 1, l, r);
56         else return query(x, m, k << 1, l, m) + query(m + 1, y, k << 1 | 1, m + 1, r);
57     }

```

```
58 };
```

### 3.6 二维线段树

```
1
2 const int N = 1005;
3
4 struct SegTree {
5     inline int son(int k, int x) {
6         return (k << 2) - 2 + x;
7     }
8
9     struct node {
10         int l, r;
11
12         node() = default;
13
14         node(int a, int b) : l(a), r(b) {}
15
16         inline int mid() {
17             return (l + r) >> 1;
18         }
19
20         inline node left() {
21             return node(l, mid());
22         }
23
24         inline node right() {
25             return node(mid() + 1, r);
26         }
27
28         inline bool in(int x) {
29             return x >= l && x <= r;
30         }
31
32         inline bool more() {
33             return l < r;
34         }
35
36         bool operator==(const node &t) {
37             return l == t.l && r == t.r;
38         }
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);
97             change(x.right(), y, son(k, 1), node(x.mid() + 1, l.r), r, v);
98         }
99     }
100    if (y.more()) {
101        if (r.l <= y.mid()) {
102            change(x, y.left(), son(k, 2), l, r, v);
103        } else if (r.r > y.mid()) {
104            change(x, y.right(), son(k, 3), l, r, v);
105        } else {
106            change(x, y.left(), son(k, 2), l, node(r.l, y.mid()), v);
107            change(x, y.right(), son(k, 3), l, node(y.mid() + 1, r.r), v);
108        }
109    }
110    up(k, x.more(), y.more());
111 }
112
113 ll query(node x, node y, int k, node l, node r) {

```

```

114     if (x == l && y == r) {
115         return ans[k] + laz[k];
116     }
117     push(k);
118     ll t = 0;
119     if (x.more()) {
120         if (l.r <= x.mid()) {
121             t += query(x.left(), y, son(k, 0), l, r);
122         } else if (l.l > x.mid()) {
123             t += query(x.right(), y, son(k, 1), l, r);
124         } else {
125             t += query(x.left(), y, son(k, 0), node(l.l, x.mid()), r);
126             t += query(x.right(), y, son(k, 1), node(x.mid() + 1, l.r), r);
127         }
128     }
129     if (y.more()) {
130         if (r.l <= y.mid()) {
131             t += query(x, y.left(), son(k, 2), l, r);
132         } else if (r.r > y.mid()) {
133             t += query(x, y.right(), son(k, 3), l, r);
134         } else {
135             t += query(x, y.left(), son(k, 2), l, node(r.l, y.mid()));
136             t += query(x, y.right(), son(k, 3), l, node(y.mid() + 1, r.r));
137         }
138     }
139     return t;
140 }
141 };

```

### 3.7 树状数组求逆序对

```

1 BITree t;
2 int n;
3 pii a[N];
4
5 void solve() {
6     t.init(n);
7     for (int i = 1; i <= n; i++) {
8         int x;
9         cin >> x;
10        a[i] = make_pair(x, i);
11    }
12    sort(a + 1, a + n + 1);
13    ll ans = 0;
14    for (int i = 1; i <= n; i++) {
15        t.change(a[i].second, 1);
16        ans += (i - t.query(a[i].second));
17    }
18    cout << ans << endl;
19 }

```

## 4 图论

### 4.1 Dijkstra

```
1 #define INF 100000000
2
3 struct node {
4     int x, d;
5     node() = default;
6     node(int a, int b): x(a), d(b) {}
7     bool operator<(const node &a) const {
8         return (a.d == d ? x < a.x : d < a.d);
9     }
10 };
11
12 #define N_MAX 150
13 vector<node> eg[N_MAX];
14 int dist[N_MAX];
15 vector<int> path[N_MAX];
16
17 void dfs(int s, int n) {
18     for (int i = 1; i <= n; i++) {
19         dist[i] = INF;
20     }
21     dist[s] = 0;
22     priority_queue<node> list;
23     list.push(node(s, dist[s]));
24     path[s].push_back(s);
25     while (!list.empty()) {
26         node x = list.top();
27         list.pop();
28         for(const auto &y: eg[x.x]){
29             if (dist[y.x] > x.d + y.d) {
30                 dist[y.x] = x.d + y.d;
31                 path[y.x] = path[x.x];
32                 path[y.x].push_back(y.x);
33                 list.push(node(y.x, dist[y.x]));
34             }
35         }
36     }
37 }
```

### 4.2 spfa

```
1 vector<int> dist;
2 vector<vector<node>> eg;
3 vector<int> path;
4
5 bool spfa(int n, int start) {
6     dist.assign(n, INF);
7     dist[start] = 0;
8     deque<int> q;
9     q.push_back(start);
10    path.assign(n, -1);
11    vector<int> cnt(n, 0);
12    vector<bool> flag(n, false);
13    cnt[start] = flag[start] = true;
14    while (!q.empty()) {
```

```

15     const int now = q.front();
16     q.pop_front();
17     flag[now] = false;
18     for (auto i: eg[now]) {
19         if (dist[i.x] > dist[now] + i.d) {
20             dist[i.x] = dist[now] + i.d;
21             path[i.x] = now;
22             if (!flag[i.x]) {
23                 if (n == ++cnt[i.x]) return false;
24                 //队列非空且优于队首 (SLF)
25                 if (!q.empty() && dist[i.x] < dist[q.front()]) {
26                     q.push_front(i.x);
27                 } else {
28                     q.push_back(i.x);
29                 }
30                 flag[i.x] = true;
31             }
32         }
33     }
34 }
35 return true;
36 }

```

### 4.3 Dinic

```

1  #define N 2005
2  #define INF 0x7fffffff
3
4  struct dinic {
5
6      struct node {
7          int e;
8          ll f;
9
10         node() = default;
11
12         node(int a, ll b) : e(a), f(b) {}
13     };
14
15     // 点的范围[0, n)
16     int n;
17     vector<node> eg;
18     vector<int> head[N];
19     // 弧优化
20     int cur[N], dis[N];
21
22     dinic() = default;
23
24     // 设置N
25     void setN(int n) {
26         this->n = n;
27     }
28
29     inline void addEdge(int x, int y, ll f) {
30         //printf("%d->%d: %lld\n", x, y, f);
31         head[x].push_back(static_cast<int> &&(eg.size()));
32         eg.push_back({y, f});
33     }

```

```

34
35 // 加流量
36 void addFlow(int x, int y, ll f) {
37     addEdge(x, y, f);
38     addEdge(y, x, 0);
39 }
40
41 bool bfs(int s, int e) {
42     fill_n(dis, n, -1);
43     int q[N];
44     int l, r;
45     l = r = 0;
46     dis[s] = 0;
47     q[r++] = s;
48     while (l < r) {
49         int f = q[l++];
50         for (const auto &i: head[f]) {
51             if (eg[i].f > 0 && dis[eg[i].e] == -1) {
52                 dis[eg[i].e] = dis[f] + 1;
53                 q[r++] = eg[i].e;
54             }
55         }
56     }
57     return dis[e] > 0;
58 }
59
60 ll dfs(int s, int e, ll mx) {
61     if (s == e || mx == 0) {
62         return mx;
63     }
64     int flow = 0;
65     for (int &k = cur[s]; k < head[s].size(); k++) {
66         int &i = head[s][k];
67         auto &te = eg[i];
68         ll a;
69         if (te.f > 0 && dis[te.e] == dis[s] + 1 && (a = dfs(te.e, e, min(te.f, mx)))
70     )) {
71         te.f -= a;
72         eg[i ^ 1].f += a;
73         flow += a;
74         mx -= a;
75         if (mx <= 0) break;
76     }
77     return flow;
78 }
79
80 ll max_flow(int s, int e) {
81     ll ans = 0;
82     while (bfs(s, e)) {
83         fill_n(cur, n, 0);
84         ans += dfs(s, e, INF);
85     }
86     return ans;
87 }
88
89 // 清空数据
90 void clear() {
91     rep(i, 0, n) head[i].clear();

```

```
92     eg.clear();
93 }
94 };
```

#### 4.4 hungry

```
1  #define N 105
2  #define M 10005
3  int n, m, k;
4  pii eg[M * 2];
5  int result[N * 2];
6  int head[N * 2];
7  int cnt = 0;
8
9  void addEdge(int x, int y) {
10     eg[cnt].first = y;
11     eg[cnt].second = head[x];
12     head[x] = cnt++;
13 }
14
15 bool vis[M * 2] = {false};
16
17 int dfs(int x) {
18     for (int i = head[x]; ~i; i = eg[i].second) {
19         int y = eg[i].first;
20         if (!vis[y]) {
21             vis[y] = true;
22             if (result[y] == -1 || dfs(result[y])) {
23                 result[y] = x;
24                 return 1;
25             }
26         }
27     }
28     return 0;
29 }
30
31 int MaxMatch() {
32     int ans = 0;
33     memset(result, -1, sizeof(result));
34     rep(i, 1, n + 1) {
35         memset(vis, 0, sizeof(vis));
36         ans += dfs(i);
37     }
38     return ans;
39 }
40
41 void solve() {
42     scanf("%d%d", &m, &k);
43     memset(head, -1, sizeof(head));
44     cnt = 0;
45     rep(i, 0, k) {
46         int x, y;
47         scanf("%d%d", &x, &y);
48         addEdge(x, y);
49     }
50     int ans = MaxMatch();
51     printf("%d\n", ans);
52 }
```

## 4.5 MinSpanTree

```

1  #define INF 1000000
2  #define N 100000
3  ll lowcost[N]; //此数组用来记录第j个节点到其余节点最少花费
4  ll mpp[N][N]; //用来记录第i个节点到其余n-1个节点的距离
5  int vis[N]; //用来记录最小生成树中的节点
6  ll city;
7
8  ll prim() {
9      ll min, i, j, next = 0, mincost = 0;
10     memset(vis, 0, sizeof(vis)); //给最小生成树数组清零
11     for (i = 1; i <= city; i++) {
12         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
13     }
14     vis[1] = 1; //选择第一个点为最小生成树的起点
15     for (i = 1; i < city; i++) {
16         min = INF;
17         for (j = 1; j <= city; j++) {
18             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
19                 min = lowcost[j];
20                 next = j; //记录下此时最小的位置节点
21             }
22         }
23         if (min == INF) return INF;
24         mincost += min; //将最小生成树中所有权值相加
25         vis[next] = 1; //next点加入最小生成树
26         for (j = 1; j <= city; j++) {
27             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
28                 lowcost[j] = mpp[next][j]; //更新lowcost数组
29             }
30         }
31     }
32     return mincost;
33 }

```

## 5 博弈

### 5.1 GameProblem

```

1 // 巴什博弈，是否先手必胜
2 inline bool bash_game(int n, int m) {
3     //一堆东西，n个物品，最多选m个
4     return n % (m + 1);
5 }
6
7 // 威佐夫博弈，是否先手必胜
8 // 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。
9 inline bool wythoff_game(int n, int m) {
10     if (n > m) {
11         swap(n, m);
12     }
13     int temp = floor((n2 - n1) * (1 + sqrt(5.0)) / 2.0);
14     return temp != n1;
15 }
16 // SG函数
17 #define N 1001
18 //f[]: 可以取走的石子个数
19 //sg[]:0~n的SG函数值
20 int f[N], sg[N], mex[N];
21
22 void getSG(int n) {
23     int i, j;
24     memset(sg, 0, sizeof(sg));
25     for (i = 1; i <= n; i++) {
26         memset(mex, 0, sizeof(mex));
27         for (j = 1; f[j] <= i; j++)
28             mex[sg[i - f[j]]] = 1;
29         for (j = 0; j <= n; j++) { //求mex{}中未出现的最小的非负整数
30             if (mex[j] == 0) {
31                 sg[i] = j;
32                 break;
33             }
34         }
35     }
36 }

```



## 6 分治

### 6.1 IntegerFastPower

```

1 ll fpow(ll x, ll k) {
2     ll base = x, r = 1;
3     for (; k >= 1) {
4         if (k & 1) r = r * base;
5         base = base * base;
6     }
7     return r;
8 }

```

### 6.2 MatrixFastPower

```

1 #define MAX_N 10
2 #define mod_num 9973
3
4 struct Mat {
5     long long mat[MAX_N][MAX_N];
6     long long n;
7     Mat() {
8         memset(mat, 0, sizeof(mat));
9         n = 0;
10    }
11    Mat(long long n) {
12        memset(mat, 0, sizeof(mat));
13        this->n = n;
14    }
15    void init() {
16        for (int i = 0; i < n; ++i) {
17            mat[i][i] = 1;
18        }
19    }
20    Mat(const long long ** list, long long n) {
21        this->n = n;
22        for (int i = 0; i < n; ++i) {
23            for (int j = 0; j < n; ++j) {
24                mat[i][j] = list[i][j];
25            }
26        }
27    }
28 };
29
30 Mat operator * (Mat a, Mat b) {
31     long long n = a.n;
32     Mat c(n);
33     memset(c.mat, 0, sizeof(c.mat));
34     for (int i = 0; i < n; ++i) {
35         for (int j = 0; j < n; ++j) {
36             for (int k = 0; k < n; ++k) {
37                 c.mat[i][j] += (a.mat[i][k] * b.mat[k][j]) % mod_num;
38                 c.mat[i][j] %= mod_num;
39             }
40         }
41     }
42     return c;
43 }

```

```
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

```

1
2 // base and base_digits must be consistent
3 constexpr int base = 1000000000;
4 constexpr int base_digits = 9;
5
6 struct bigint {
7     // value == 0 is represented by empty z
8     vector<int> z; // digits
9
10    // sign == 1 <==> value >= 0
11    // sign == -1 <==> value < 0
12    int sign;
13
14    bigint() : sign(1) {}
15
16    bigint(long long v) { *this = v; }
17
18    bigint &operator=(long long v) {
19        sign = v < 0 ? -1 : 1;
20        v *= sign;
21        z.clear();
22        for (; v > 0; v = v / base)
23            z.push_back((int) (v % base));
24        return *this;
25    }
26
27    bigint(const string &s) { read(s); }
28
29    bigint &operator+=(const bigint &other) {
30        if (sign == other.sign) {
31            for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
32                if (i == z.size())
33                    z.push_back(0);
34                z[i] += carry + (i < other.z.size() ? other.z[i] : 0);
35                carry = z[i] >= base;
36                if (carry)
37                    z[i] -= base;
38            }
39        } else if (other != 0 /* prevent infinite loop */) {
40            *this -= -other;
41        }
42        return *this;
43    }
44
45    friend bigint operator+(bigint a, const bigint &b) {
46        return a += b;
47    }
48
49    bigint &operator-=(const bigint &other) {
50        if (sign == other.sign) {
51            if (sign == 1 && *this >= other || sign == -1 && *this <= other) {
52                for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
53                    z[i] -= carry + (i < other.z.size() ? other.z[i] : 0);
54                    carry = z[i] < 0;
55                    if (carry)

```

```

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

```

```

115     }
116
117     friend bigint sqrt(const bigint &a1) {
118         bigint a = a1;
119         while (a.z.empty() || a.z.size() % 2 == 1)
120             a.z.push_back(0);
121
122         int n = a.z.size();
123
124         int firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
125         int norm = base / (firstDigit + 1);
126         a *= norm;
127         a *= norm;
128         while (a.z.empty() || a.z.size() % 2 == 1)
129             a.z.push_back(0);
130
131         bigint r = (long long) a.z[n - 1] * base + a.z[n - 2];
132         firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
133         int q = firstDigit;
134         bigint res;
135
136         for (int j = n / 2 - 1; j >= 0; j--) {
137             for (; --q) {
138                 bigint r1 = (r - (res * 2 * base + q) * q) * base * base +
139                     (j > 0 ? (long long) a.z[2 * j - 1] * base + a.z[2 * j - 2]
140 : 0);
141                 if (r1 >= 0) {
142                     r = r1;
143                     break;
144                 }
145                 res *= base;
146                 res += q;
147
148                 if (j > 0) {
149                     int d1 = res.z.size() + 2 < r.z.size() ? r.z[res.z.size() + 2] : 0;
150                     int d2 = res.z.size() + 1 < r.z.size() ? r.z[res.z.size() + 1] : 0;
151                     int d3 = res.z.size() < r.z.size() ? r.z[res.z.size()] : 0;
152                     q = (int) (((long long) d1 * base * base + (long long) d2 * base + d3)
153 / (firstDigit * 2));
154                 }
155             }
156             res.trim();
157             return res / norm;
158         }
159
160         bigint operator/(const bigint &v) const {
161             return divmod(*this, v).first;
162         }
163
164         bigint operator%(const bigint &v) const {
165             return divmod(*this, v).second;
166         }
167
168         bigint &operator/=(int v) {
169             if (v < 0)
170                 sign = -sign, v = -v;
171             for (int i = (int) z.size() - 1, rem = 0; i >= 0; --i) {

```

```
172         long long cur = z[i] + rem * (long long) base;
173         z[i] = (int) (cur / v);
174         rem = (int) (cur % v);
175     }
176     trim();
177     return *this;
178 }
179
180 bigint operator/(int v) const {
181     return bigint(*this) /= v;
182 }
183
184 int operator%(int v) const {
185     if (v < 0)
186         v = -v;
187     int m = 0;
188     for (int i = (int) z.size() - 1; i >= 0; --i)
189         m = (int) ((z[i] + m * (long long) base) % v);
190     return m * sign;
191 }
192
193 bigint &operator*=(const bigint &v) {
194     *this = *this * v;
195     return *this;
196 }
197
198 bigint &operator/=(const bigint &v) {
199     *this = *this / v;
200     return *this;
201 }
202
203 bool operator<(const bigint &v) const {
204     if (sign != v.sign)
205         return sign < v.sign;
206     if (z.size() != v.z.size())
207         return z.size() * sign < v.z.size() * v.sign;
208     for (int i = (int) z.size() - 1; i >= 0; i--)
209         if (z[i] != v.z[i])
210             return z[i] * sign < v.z[i] * sign;
211     return false;
212 }
213
214 bool operator>(const bigint &v) const {
215     return v < *this;
216 }
217
218 bool operator<=(const bigint &v) const {
219     return !(v < *this);
220 }
221
222 bool operator>=(const bigint &v) const {
223     return !(*this < v);
224 }
225
226 bool operator==(const bigint &v) const {
227     return !(*this < v) && !(v < *this);
228 }
229
230 bool operator!=(const bigint &v) const {
```

```

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

```

```

290     stream >> s;
291     v.read(s);
292     return stream;
293 }
294
295 friend ostream &operator<<(ostream &stream, const bigint &v) {
296     if (v.sign == -1)
297         stream << '-';
298     stream << (v.z.empty() ? 0 : v.z.back());
299     for (int i = (int) v.z.size() - 2; i >= 0; --i)
300         stream << setw(base_digits) << setfill('0') << v.z[i];
301     return stream;
302 }
303
304 static vector<int> convert_base(const vector<int> &a, int old_digits, int
new_digits) {
305     vector<long long> p(max(old_digits, new_digits) + 1);
306     p[0] = 1;
307     for (int i = 1; i < p.size(); i++)
308         p[i] = p[i - 1] * 10;
309     vector<int> res;
310     long long cur = 0;
311     int cur_digits = 0;
312     for (int v : a) {
313         cur += v * p[cur_digits];
314         cur_digits += old_digits;
315         while (cur_digits >= new_digits) {
316             res.push_back(int(cur % p[new_digits]));
317             cur /= p[new_digits];
318             cur_digits -= new_digits;
319         }
320     }
321     res.push_back((int) cur);
322     while (!res.empty() && res.back() == 0)
323         res.pop_back();
324     return res;
325 }
326
327 typedef vector<long long> vll;
328
329 static vll karatsubaMultiply(const vll &a, const vll &b) {
330     int n = a.size();
331     vll res(n + n);
332     if (n <= 32) {
333         for (int i = 0; i < n; i++)
334             for (int j = 0; j < n; j++)
335                 res[i + j] += a[i] * b[j];
336         return res;
337     }
338
339     int k = n >> 1;
340     vll a1(a.begin(), a.begin() + k);
341     vll a2(a.begin() + k, a.end());
342     vll b1(b.begin(), b.begin() + k);
343     vll b2(b.begin() + k, b.end());
344
345     vll a1b1 = karatsubaMultiply(a1, b1);
346     vll a2b2 = karatsubaMultiply(a2, b2);
347

```



```

348     for (int i = 0; i < k; i++)
349         a2[i] += a1[i];
350     for (int i = 0; i < k; i++)
351         b2[i] += b1[i];
352
353     vll r = karatsubaMultiply(a2, b2);
354     for (int i = 0; i < a1b1.size(); i++)
355         r[i] -= a1b1[i];
356     for (int i = 0; i < a2b2.size(); i++)
357         r[i] -= a2b2[i];
358
359     for (int i = 0; i < r.size(); i++)
360         res[i + k] += r[i];
361     for (int i = 0; i < a1b1.size(); i++)
362         res[i] += a1b1[i];
363     for (int i = 0; i < a2b2.size(); i++)
364         res[i + n] += a2b2[i];
365     return res;
366 }
367
368 bigint operator*(const bigint &v) const {
369     vector<int> a6 = convert_base(this->z, base_digits, 6);
370     vector<int> b6 = convert_base(v.z, base_digits, 6);
371     vll a(a6.begin(), a6.end());
372     vll b(b6.begin(), b6.end());
373     while (a.size() < b.size())
374         a.push_back(0);
375     while (b.size() < a.size())
376         b.push_back(0);
377     while (a.size() & (a.size() - 1))
378         a.push_back(0), b.push_back(0);
379     vll c = karatsubaMultiply(a, b);
380     bigint res;
381     res.sign = sign * v.sign;
382     for (int i = 0, carry = 0; i < c.size(); i++) {
383         long long cur = c[i] + carry;
384         res.z.push_back((int) (cur % 1000000));
385         carry = (int) (cur / 1000000);
386     }
387     res.z = convert_base(res.z, 6, base_digits);
388     res.trim();
389     return res;
390 }
391 };

```

## 7.2 FastIO

```

1
2 /*
3  * FastIO
4  * 代码模板 !
5  * 如有雷同 !
6  * 纯属巧合 !
7  */
8 namespace FastIO {
9     #define BUF_SIZE 10000000
10    #define OUT_SIZE 10000000
11    #define ll long long

```

```

12 //fread->read
13 bool IOError = 0;
14
15 inline char nc() {
16     static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
17     if (p1 == pend) {
18         p1 = buf;
19         pend = buf + fread(buf, 1, BUF_SIZE, stdin);
20         if (pend == p1) {
21             IOError = 1;
22             return -1;
23         }
24         //printf("IO error!\n");system("pause");for (;;);exit(0);}
25     }
26     return *p1++;
27 }
28
29 inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch ==
'\t'; }
30
31 inline void read(int &x) {
32     bool sign = 0;
33     char ch = nc();
34     x = 0;
35     for (; blank(ch); ch = nc());
36     if (IOError) return;
37     if (ch == '-') sign = 1, ch = nc();
38     for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
39     if (sign) x = -x;
40 }
41
42 inline void read(ll &x) {
43     bool sign = 0;
44     char ch = nc();
45     x = 0;
46     for (; blank(ch); ch = nc());
47     if (IOError) return;
48     if (ch == '-') sign = 1, ch = nc();
49     for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
50     if (sign) x = -x;
51 }
52
53 inline void read(double &x) {
54     bool sign = 0;
55     char ch = nc();
56     x = 0;
57     for (; blank(ch); ch = nc());
58     if (IOError) return;
59     if (ch == '-') sign = 1, ch = nc();
60     for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
61     if (ch == '.') {
62         double tmp = 1;
63         ch = nc();
64         for (; ch >= '0' && ch <= '9'; ch = nc()) tmp /= 10.0, x += tmp * (ch - '0')
;
65     }
66     if (sign) x = -x;
67 }
68

```

```

69     inline void read(char *s) {
70         char ch = nc();
71         for (; blank(ch); ch = nc());
72         if (IOerror) return;
73         for (; !blank(ch) && !IOerror; ch = nc()) *s++ = ch;
74         *s = 0;
75     }
76
77     inline void read(char &c) {
78         for (c = nc(); blank(c); c = nc());
79         if (IOerror) {
80             c = -1;
81             return;
82         }
83     }
84
85     //fwrite->write
86     struct Ostream_fwrite {
87         char *buf, *p1, *pend;
88
89         Ostream_fwrite() {
90             buf = new char[OUT_SIZE];
91             p1 = buf;
92             pend = buf + OUT_SIZE;
93         }
94
95         void out(char ch) {
96             if (p1 == pend) {
97                 fwrite(buf, 1, OUT_SIZE, stdout);
98                 p1 = buf;
99             }
100             *p1++ = ch;
101         }
102
103         void print(int x) {
104             static char s[15], *s1;
105             s1 = s;
106             if (!x) *s1++ = '0';
107             if (x < 0) out('-'), x = -x;
108             while (x) *s1++ = x % 10 + '0', x /= 10;
109             while (s1-- != s) out(*s1);
110         }
111
112         void println(int x) {
113             static char s[15], *s1;
114             s1 = s;
115             if (!x) *s1++ = '0';
116             if (x < 0) out('-'), x = -x;
117             while (x) *s1++ = x % 10 + '0', x /= 10;
118             while (s1-- != s) out(*s1);
119             out('\n');
120         }
121
122         void print(ll x) {
123             static char s[25], *s1;
124             s1 = s;
125             if (!x) *s1++ = '0';
126             if (x < 0) out('-'), x = -x;
127             while (x) *s1++ = x % 10 + '0', x /= 10;

```

```

128     while (s1-- != s)out(*s1);
129 }
130
131 void println(ll x) {
132     static char s[25], *s1;
133     s1 = s;
134     if (!x)*s1++ = '0';
135     if (x < 0)out('-'), x = -x;
136     while (x)*s1++ = x % 10 + '0', x /= 10;
137     while (s1-- != s)out(*s1);
138     out('\n');
139 }
140
141 void print(double x, int y) {
142     static ll mul[] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000,
100000000,
143                        1000000000, 10000000000LL, 100000000000LL, 1000000000000LL,
144                        10000000000000LL, 100000000000000LL, 1000000000000000LL,
145                        10000000000000000LL};
146     if (x < -1e-12)out('-'), x = -x;
147     x *= mul[y];
148     ll x1 = (ll) floor(x);
149     if (x - floor(x) >= 0.5)++x1;
150     ll x2 = x1 / mul[y], x3 = x1 - x2 * mul[y];
151     print(x2);
152     if (y > 0) {
153         out('.');
154         for (size_t i = 1; i < y && x3 * mul[i] < mul[y]; out('0'), ++i);
155         print(x3);
156     }
157 }
158
159 void println(double x, int y) {
160     print(x, y);
161     out('\n');
162 }
163
164 void print(char *s) { while (*s)out(*s++); }
165
166 void println(char *s) {
167     while (*s)out(*s++);
168     out('\n');
169 }
170
171 void flush() {
172     if (p1 != buf) {
173         fwrite(buf, 1, p1 - buf, stdout);
174         p1 = buf;
175     }
176 }
177
178 ~Ostream_fwrite() { flush(); }
179 } Ostream;
180
181 inline void print(int x) { Ostream.print(x); }
182
183 inline void println(int x) { Ostream.println(x); }

```

```

184     inline void print(char x) { Ostream.out(x); }
185
186     inline void println(char x) {
187         Ostream.out(x);
188         Ostream.out('\n');
189     }
190
191     inline void print(ll x) { Ostream.print(x); }
192
193     inline void println(ll x) { Ostream.println(x); }
194
195     inline void print(double x, int y) { Ostream.print(x, y); }
196
197     inline void println(double x, int y) { Ostream.println(x, y); }
198
199     inline void print(char *s) { Ostream.print(s); }
200
201     inline void println(char *s) { Ostream.println(s); }
202
203     inline void println() { Ostream.out('\n'); }
204
205     inline void flush() { Ostream.flush(); }
206 };
207 using namespace FastIO;

```

### 7.3 InputOutputSpeedUp

```

1  //适用于正负整数
2  template <class T>
3  inline bool scan_d (T &ret) {
4      char c; int sgn;
5      if( c = getchar(), c == EOF)    return 0; //EOF
6      while (c != '-' && (c < '0' || c > '9')) c = getchar();
7      sgn = (c == '-') ? -1 : 1;
8      ret = (c == '-') ? 0 : (c - '0');
9      while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
10     ret *= sgn;
11     return 1;
12 }
13 inline void out (int x) {
14     if (x < 0) {
15         putchar('-');
16         x = -x;
17     }
18     if (x > 9) out (x / 10);
19     putchar (x % 10 + '0');
20 }
21 inline void out(int x) {
22     if (x < 0) {
23         putchar('-');
24         x = -x;
25     }
26     char list[100];
27     int now = 0;
28     while (x > 9) {
29         list[++now] = (x % 10 + '0');
30         x /= 10;
31     }

```

```
32     putchar(x + '0');
33     while (now) {
34         putchar(list[now--]);
35     }
36 }
```

## 7.4 gcd

```
1 int gcd(int x, int y) {
2     int t;
3     while (y){
4         t = x % y;
5         x = y;
6         y = t;
7     }
8     return x;
9 }
```

## 7.5 IntFastSqrt

```
1 int sqrtI(int t){
2     int min = 0;
3     int max = t;
4     while (min != max && min + 1 != max) {
5         int mid = (min + max) / 2;
6         int a = mid * mid;
7         if (a == t) {
8             return mid;
9         }
10        else if (a > t) {
11            max = mid;
12        }
13        else if (a < t) {
14            min = mid;
15        }
16    }
17    return min;
18 }
```

## 7.6 myItoa

```
1 char * myItoa(int value, char* result, int base = 10);
2
3 char * myItoa(int value, char* result, int base) {
4     // check that the base is valid
5
6     if (base < 2 || base > 16) { *result = 0; return result; }
7     char* out = result;
8     int quotient = abs(value);
9     do {
10         const int tmp = quotient / base;
11         *out = "0123456789abcdef"[quotient - (tmp*base)];
12         ++out;
13         quotient = tmp;
14     } while (quotient);
15     // Apply negative sign
```

```
16     if (value < 0) *out++ = '-';
17     std::reverse(result, out);
18     *out = 0;
19     return result;
20 }
```

## 7.7 prime

```
1  #define prime_max 1000000
2
3  int prime_count = 0;
4  bool prime_list[prime_max] = { false }; //元素值为0代表是素数
5  int prime_table[prime_max] = { 0 };
6
7  void initPrime(){
8      for (int i = 2; i < prime_max; i++){
9          if (!prime_list[i])
10             prime_table[prime_count++] = i;
11             for (int j = 0; j < prime_count && i*prime_table[j] < prime_max; j++){
12                 prime_list[i*prime_table[j]] = 1;
13                 if (i % prime_table[j] == 0) break;
14             }
15     }
16 }
```

## 7.8 Permutation

```
1  // 错排问题
2  //  $D(n) = n! [(-1)^2/2! + \dots + (-1)^{(n-1)}/(n-1)! + (-1)^n/n!]$ .
3  long long table[1000] = {0, 0, 1};
4  void init() {
5      for (int i = 3; i <= 20; i++) {
6          table[i] = (i - 1) * (table[i - 1] + table[i - 2]);
7      }
8  }
```

## 7.9 reverseInt

```
1  template<typename T>
2
3  T reverseInt(T a)
4  {
5      T b = 0;
6      while (a != 0) {
7          b *= 10;
8          b += (a % 10);
9          a /= 10;
10     }
11     return b;
12 }
```