ACM/ICPC Template Manaual

ZheJiang GongShang University

Happy Otaku

April 26, 2019

Contents

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 long long ll;
9   typedef vector<int> vi;
10  typedef pair<int, int> pii;
11  #define pw(x) (1ll << (x))
12  #define sz(x) ((int)(x).size())
13  #define all(x) (x).begin(),(x).end()
14  #define rep(i, l, r) for(int i=(l);i<(r);++i)
15  #define per(i, l, r) for(int i=(r)-1;i>=(l);--i)
16  #define sf(x) scanf("%d", &(x))
17
18  using namespace std;
19
20  const double pi = acos(-1);
```

0.1   stdc++

```
1   // C
2   #ifndef _GLIBCXX_NO_ASSERT
3   #include <cassert>
4   #endif
5   #include <cctype>
6   #include <cerrno>
7   #include <cfloat>
8   #include <ciso646>
9   #include <climits>
10  #include <clocale>
11  #include <cmath>
12  #include <csetjmp>
13  #include <csignal>
14  #include <cstdarg>
15  #include <cstddef>
16  #include <cstdio>
17  #include <cstdlib>
18  #include <cstring>
19  #include <ctime>
20
21  #if __cplusplus >= 201103L
22  #include <ccomplex>
23  #include <cfenv>
24  #include <cinttypes>
25  #include <cstdalign>
26  #include <cstdbool>
27  #include <cstdint>
28  #include <ctgmath>
29  #include <cuchar>
30  #include <cwchar>
31  #include <cwctype>
32  #endif
33
```

```
34  // C++
35  #include <algorithm>
36  #include <bitset>
37  #include <complex>
38  #include <deque>
39  #include <exception>
40  #include <fstream>
41  #include <functional>
42  #include <iomanip>
43  #include <ios>
44  #include <iosfwd>
45  #include <iostream>
46  #include <istream>
47  #include <iterator>
48  #include <limits>
49  #include <list>
50  #include <locale>
51  #include <map>
52  #include <memory>
53  #include <new>
54  #include <numeric>
55  #include <ostream>
56  #include <queue>
57  #include <set>
58  #include <sstream>
59  #include <stack>
60  #include <stdexcept>
61  #include <streambuf>
62  #include <string>
63  #include <typeinfo>
64  #include <utility>
65  #include <valarray>
66  #include <vector>
67
68  #if __cplusplus >= 201103L
69  #include <array>
70  #include <atomic>
71  #include <chrono>
72  #include <codecvt>
73  #include <condition_variable>
74  #include <forward_list>
75  #include <future>
76  #include <initializer_list>
77  #include <mutex>
78  #include <random>
79  #include <ratio>
80  #include <regex>
81  #include <scoped_allocator>
82  #include <system_error>
83  #include <thread>
84  #include <tuple>
85  #include <typeindex>
86  #include <type_traits>
87  #include <unordered_map>
88  #include <unordered_set>
89  #endif
90  #if __cplusplus >= 201402L
91  #include <shared_mutex>
92  #endif
```

1 字串符

1.1 KMP

```
 1  template<class elemType>
 2  inline void kmp_nxt(elemType &T, vector<int> &nxt) {
 3      nxt[0] = -1;
 4      for (int i = 1; i < T.size(); i++) {
 5          int j = nxt[i - 1];
 6          while (j >= 0 && T[i - 1] != T[j]) j = nxt[j];
 7          if (j >= 0 && T[i - 1] == T[j]) nxt[i] = j + 1;
 8          else nxt[i] = 0;
 9      }
10  }
11
12  template<class elemType>
13  inline int kmp_count(elemType &S, elemType &T) {
14      vector<int> nxt(T.size());
15      kmp_nxt(T, nxt);
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 = nxt[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 kmp_next(elemType T[], int count, vector<int> &nxt) {
36      nxt[0] = -1;
37      for (int i = 1; i < count; i++) {
38          int j = nxt[i - 1];
39          while (j >= 0 && T[i - 1] != T[j]) j = nxt[j];
40          if (j >= 0 && T[i - 1] == T[j]) nxt[i] = j + 1;
41          else nxt[i] = 0;
42      }
43  }
44
45  template<class elemType>
46  inline int kmp_count(elemType S[], int c1, elemType T[], int c2) {
47      vector<int> nxt(c2);
48      kmp_nxt(T, c2, nxt);
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 = nxt[pos - 1] + 1;
61              }
62          }
63          if (pos == c2 && (iter - index) == c2) ++count;
64      }
65      return count;
66  }
```

2 动态规划

2.1 01Bag

```
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 <cstdio>
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--)
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
    的钱,当然最后就用max了, HEHE
39              }
40          }
41          if(f[c] == INF)
42              printf("This is impossible.\n");
43          else
44              printf("The minimum amount of money in the piggy-bank is %d.\n",f[c]);
45      }
46      return 0;
47  }
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 <cstdio>
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   // RBTree 红黑树
2   #include <ext/pb_ds/tree_policy.hpp>
3   #include <ext/pb_ds/assoc_container.hpp>
4   // 红黑树
5   __gnu_pbds::tree<int, null_type, less<int>, rb_tree_tag,
        tree_order_statistics_node_update> t;
6   // null_type无映射(低版本g++为null_mapped_type)
7   // 类似multiset
8   __gnu_pbds::tree<int, null_type, less_equal<int>, rb_tree_tag,
        tree_order_statistics_node_update> t;
9   find_by_order(size_t order);
10  // 结点更新
11  tree_order_statistics_node_update
12  insert(p);
13  erase(it);
14  // 求k在树中是第几大:
15  order_of_key(p);
16  // 找到第order小的迭代器
17  find_by_order(order);
18  // 前驱
19  lower_bound(p);
20  // 后驱
21  upper_bound(p);
22  // 合并
23  a.join(b);
24  // 分割 key小于等于v的元素属于a, 其余的属于b
25  a.split(v, b);
26
27  // 优先队列
28  #include <ext/pb_ds/priority_queue.hpp>
29  #include <ext/pb_ds/assoc_container.hpp>
30  // 操作类似于stl的优先队列
31  typedef __gnu_pbds::priority_queue<node, greater<node>, __gnu_pbds::thin_heap_tag> heap
        ;
32  heap::point_iterator; // 指向元素的指针
```

3.3   树状数组

```
1   const int N = 1000005;
2   struct BITree {
3       int n;
```

```
4      ll c[N];
5
6      void init(int n) {
7          memset(c, 0, sizeof(ll) * ++n);
8          this->n = n;
9      }
10
11      void change(int pos, ll v) {
12          for (int i = pos; i < n; i += i & (-i))
13              c[i] += v;
14      }
15
16      ll query(int x) {
17          ll ans = 0;
18          for (int i = x; i > 0; i -= i & (-i))
19              ans += c[i];
20          return ans;
21      }
22  };
```

3.4 二维树状数组

```
1   const int N = 2005;
2
3   inline int lowbit(const int &x) {
4          return x & -x;
5      }
6
7   struct TdBITree {
8       int n, m;
9       ll c[N][N];
10
11      void init(int n, int m) {
12          this->n = n;
13          this->m = m;
14          memset(c, 0, sizeof(c))
15      }
16
17      void init(int n, int m, ll v) {
18          this->n = n;
19          this->m = m;
20          rep(x, 1, N) {
21              rep(y, 1, N) {
22                  c[x][y] = (x * y + (x - lowbit(x)) * (y - lowbit(y)) - x * (y - lowbit(
    y)) - (x - lowbit(x)) * y) * v;
23              }
24          }
25      }
26
27      void change(int x, int y, ll v) {
28          for (int i = x; i <= n; i += lowbit(i))
29              for (int j = y; j <= m; j += lowbit(j))
30                  c[i][j] += v;
31      }
32
33      ll query(int x, int y) {
34          ll ans = 0;
35          for (int i = x; i >= 1; i -= lowbit(i))
```

```
36              for (int j = y; j >= 1; j -= lowbit(j))
37                  ans += c[i][j];
38          return ans;
39      }
40
41      ll solve(int x1, int y1, int x2, int y2) {
42          return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) + query(x1 - 1, y1
         - 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
6       inline int son(int k, int x) {
7           return (k << 2) - 2 + x;
8       }
9
10      struct node {
11          int l, r;
12
13          node() = default;
14
15          node(int a, int b) : l(a), r(b) {}
16
17          inline int mid() {
18              return (l + r) >> 1;
19          }
20
21          inline node left() {
22              return node(l, mid());
23          }
24
25          inline node right() {
26              return node(mid() + 1, r);
27          }
28
29          inline bool in(int x) {
30              return x >= l && x <= r;
31          }
32
33          inline bool more() {
34              return l < r;
35          }
36
37          bool operator==(const node &t) {
38              return l == t.l && r == t.r;
39          }
40      };
41
42      ll c[N << 2][N << 2];
```

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

```
1  struct node {
2      int e, v, nxt;
3      node() = default;
4      node(int a, int b, int c = 0) : e(a), v(b), nxt(c) {}
5
6      bool operator<(const node &a) const {
7          return (a.v == v ? e < a.e : v < a.v);
8      }
9  };
10
11 const ll INF = 10000000;
12 const int N = 100000;
13 const int M = 100000;
14
15 struct Graph {
16     node eg[M];
17     int head[N];
18     int cnt;
19
20     void init(int n) {
21         memset(head, -1, sizeof(int) * ++n);
22         cnt = 0;
23     }
24     inline void addEdge(int x, int y, int v) {
25         eg[cnt] = node(y, v, head[x]);
26         head[x] = cnt++;
27     }
28 } gh;
```

4.2 Dijkstra

```
1  int dist[N];
2  int path[N];
3
4  void bfs(int s, int n) {
5      n++;
6      rep(i, 0, n) dist[i] = INF;
7      memset(path, -1, sizeof(int) * n);
8      dist[s] = 0;
9      path[s] = s;
10     priority_queue<node> q;
11     q.push(node(s, dist[s]));
12     while (!q.empty()) {
13         node f = q.top();
14         q.pop();
15         for (int i = gh.head[f.e]; ~i; i = gh.eg[i].nxt) {
16             node &t = gh.eg[i];
17             if (dist[t.e] > f.v + t.v) {
18                 dist[t.e] = f.v + t.v;
19                 path[t.e] = f.e;
20                 q.push(node(t.e, dist[t.e]));
21             }
22         }
23     }
```

```
24  }
25
26  #include <ext/pb_ds/priority_queue.hpp>
27  #include <ext/pb_ds/assoc_container.hpp>
28  typedef __gnu_pbds::priority_queue<node, greater<node>> heap;
29
30  heap::point_iterator its[N];
31  int cnt[N];
32
33  void bfs(int s, int n) {
34      n++;
35      rep(i, 0, n) dist[i] = INF;
36      memset(cnt, 0, sizeof(int) * n);
37      dist[s] = 0;
38      cnt[s] = 1;
39      heap q;
40      its[s] = q.push(node(s, dist[s]));
41      while (!q.empty()) {
42          node f = q.top();
43          q.pop();
44          for (int i = gh.head[f.e]; ~i; i = gh.eg[i].nxt) {
45              node &t = gh.eg[i];
46              its[t.e] = 0;
47              int v = f.v + t.v;
48              if (dist[t.e] > v) {
49                  dist[t.e] = v;
50                  if (its[t.e] != 0) {
51                      q.modify(its[t.e], node(t.e, dist[t.e]));
52                  } else {
53                      its[t.e] = q.push(node(t.e, dist[t.e]));
54                  }
55                  cnt[t.e] = cnt[f.e];
56              } else if (dist[t.e] == v) {
57                  (cnt[t.e] += cnt[f.e]) %= 100003;
58              }
59          }
60      }
61  }
```

4.3 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 }
```

4.4 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                  te.f -= a;
71                  eg[i ^ 1].f += a;
72                  flow += a;
73                  mx -= a;
74                  if (mx <= 0) break;
75              }
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  };
```

4.5 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 }
```

4.6 MinSpanTree

```
1  /*
```

```
2   * Prim 求 MST
3   * 耗费矩阵 cost[][], 标号从 0 开始, 0~n-1
4   * 返回最小生成树的权值, 返回 -1 表示原图不连通
5    */
6   const int INF = 0x3f3f3f3f;
7   const int N = 110;
8   bool vis[N];
9   int lowc[N]; //点是 0 n-1
10  int prim(int cost[][N], int n) {
11      int ans = 0;
12      memset(vis, false, sizeof(vis));
13      vis[0] = true;
14      for (int i = 1; i < n; i++)lowc[i] = cost[0][i];
15      for (int i = 1; i < n; i++) {
16          int minc = INF;
17          int p =-1;
18          19
19          for (int j = 0; j < n; j++)
20              if (!vis[j] && minc > lowc[j]) {
21                  minc = lowc[j];
22                  p = j;
23              }
24          if (minc == INF)return -1;//原图不连通
25          ans += minc;
26          vis[p] = true;
27          for (int j = 0; j < n; j++)
28              if (!vis[j] && lowc[j] > cost[p][j])
29                  lowc[j] = cost[p][j];
30      }
31      return ans;
32  }
```

5 博弈

5.1 GameProblem

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

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

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

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

6 分治

6.1 IntegerFastPower

```
1  ll fpow(ll x, ll k) {
2      ll base = x, r = 1;
3      for (; k; 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

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

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

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

    bigint() : sign(1) {}

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

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

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

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

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

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

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

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

```
173             int m = 0;
174             for (int i = (int) z.size() - 1; i >= 0; --i)
175                 m = (int) ((z[i] + m * (ll) base) % v);
176             return m * sign;
177         }
178
179         bigint &operator*=(const bigint &v) {
180             return *this = *this * v;;
181         }
182
183         bigint &operator/=(const bigint &v) {
184             return *this = *this / v;
185         }
186
187         bool operator<(const bigint &v) const {
188             if (sign != v.sign)
189                 return sign < v.sign;
190             if (z.size() != v.z.size())
191                 return z.size() * sign < v.z.size() * v.sign;
192             for (int i = (int) z.size() - 1; i >= 0; i--)
193                 if (z[i] != v.z[i])
194                     return z[i] * sign < v.z[i] * sign;
195             return false;
196         }
197
198         bool operator>(const bigint &v) const { return v < *this; }
199
200         bool operator<=(const bigint &v) const { return !(v < *this); }
201
202         bool operator>=(const bigint &v) const { return !(*this < v); }
203
204         bool operator==(const bigint &v) const { return !(*this < v) && !(v < *this); }
205
206         bool operator!=(const bigint &v) const { return *this < v || v < *this; }
207
208         void trim() {
209             while (!z.empty() && z.back() == 0) z.pop_back();
210             if (z.empty()) sign = 1;
211         }
212
213         bool isZero() const {
214             return z.empty();
215         }
216
217         friend bigint operator-(bigint v) {
218             if (!v.z.empty()) v.sign = -v.sign;
219             return v;
220         }
221
222         bigint abs() const {
223             return sign == 1 ? *this : -*this;
224         }
225
226         ll longValue() const {
227             ll res = 0;
228             for (int i = (int) z.size() - 1; i >= 0; i--)
229                 res = res * base + z[i];
230             return res * sign;
231         }
```

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

```
290                 }
291             res.push_back((int) cur);
292             while (!res.empty() && res.back() == 0) res.pop_back();
293             return res;
294         }
295
296         typedef vector<ll> vll;
297
298         static vll karatsubaMultiply(const vll &a, const vll &b) {
299             int n = a.size();
300             vll res(n + n);
301             if (n <= 32) {
302                 for (int i = 0; i < n; i++)
303                     for (int j = 0; j < n; j++)
304                         res[i + j] += a[i] * b[j];
305                 return res;
306             }
307
308             int k = n >> 1;
309             vll a1(a.begin(), a.begin() + k);
310             vll a2(a.begin() + k, a.end());
311             vll b1(b.begin(), b.begin() + k);
312             vll b2(b.begin() + k, b.end());
313
314             vll a1b1 = karatsubaMultiply(a1, b1);
315             vll a2b2 = karatsubaMultiply(a2, b2);
316
317             for (int i = 0; i < k; i++) a2[i] += a1[i];
318             for (int i = 0; i < k; i++) b2[i] += b1[i];
319
320             vll r = karatsubaMultiply(a2, b2);
321             for (int i = 0; i < a1b1.size(); i++) r[i] -= a1b1[i];
322             for (int i = 0; i < a2b2.size(); i++) r[i] -= a2b2[i];
323
324             for (int i = 0; i < r.size(); i++) res[i + k] += r[i];
325             for (int i = 0; i < a1b1.size(); i++) res[i] += a1b1[i];
326             for (int i = 0; i < a2b2.size(); i++) res[i + n] += a2b2[i];
327             return res;
328         }
329
330         bigint operator*(const bigint &v) const {
331             vector<int> a6 = convert_base(this->z, base_digits, 6);
332             vector<int> b6 = convert_base(v.z, base_digits, 6);
333             vll a(a6.begin(), a6.end());
334             vll b(b6.begin(), b6.end());
335             while (a.size() < b.size()) a.push_back(0);
336             while (b.size() < a.size()) b.push_back(0);
337             while (a.size() & (a.size() - 1)) a.push_back(0), b.push_back(0);
338             vll c = karatsubaMultiply(a, b);
339             bigint res;
340             res.sign = sign * v.sign;
341             for (int i = 0, carry = 0; i < c.size(); i++) {
342                 ll cur = c[i] + carry;
343                 res.z.push_back((int) (cur % 1000000));
344                 carry = (int) (cur / 1000000);
345             }
346             res.z = convert_base(res.z, 6, base_digits);
347             res.trim();
348             return res;
```

```
349        }
350    };
```

7.2   FastIO

```
1    /*
2     * FastIO
3     * 代码模板 !
4     * 如有雷同 !
5     * 纯属巧合 !
6     */
7    namespace FastIO {
8    #define BUF_SIZE 10000000
9    #define OUT_SIZE 10000000
10   #define ll long long
11        //fread->read
12        bool IOerror = 0;
13
14        inline char nc() {
15            static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
16            if (p1 == pend) {
17                p1 = buf;
18                pend = buf + fread(buf, 1, BUF_SIZE, stdin);
19                if (pend == p1) {
20                    IOerror = 1;
21                    return -1;
22                }
23                //{printf("IO error!\n");system("pause");for (;;);exit(0);}
24            }
25            return *p1++;
26        }
27
28        inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch ==
         '\t'; }
29
30        inline void read(int &x) {
31            bool sign = 0;
32            char ch = nc();
33            x = 0;
34            for (; blank(ch); ch = nc());
35            if (IOerror)return;
36            if (ch == '-')sign = 1, ch = nc();
37            for (; ch >= '0' && ch <= '9'; ch = nc())x = x * 10 + ch - '0';
38            if (sign)x = -x;
39        }
40
41        inline void read(ll &x) {
42            bool sign = 0;
43            char ch = nc();
44            x = 0;
45            for (; blank(ch); ch = nc());
46            if (IOerror)return;
47            if (ch == '-')sign = 1, ch = nc();
48            for (; ch >= '0' && ch <= '9'; ch = nc())x = x * 10 + ch - '0';
49            if (sign)x = -x;
50        }
51
52        inline void read(double &x) {
```

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

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

```
167      inline void println(int x) { Ostream.println(x); }
168      inline void print(char x) { Ostream.out(x); }
169      inline void println(char x) {
170          Ostream.out(x);
171          Ostream.out('\n');
172      }
173      inline void print(ll x) { Ostream.print(x); }
174      inline void println(ll x) { Ostream.println(x); }
175      inline void print(double x, int y) { Ostream.print(x, y); }
176      inline void println(double x, int y) { Ostream.println(x, y); }
177      inline void print(char *s) { Ostream.print(s); }
178      inline void println(char *s) { Ostream.println(s); }
179      inline void println() { Ostream.out('\n'); }
180      inline void flush() { Ostream.flush(); }
181 };
182 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 //适用于正负整数
14 template <class T>
15 inline void outU (T x) {
16     if (x < 0)  putchar('-'), x = -x;
17     if (x > 9) out (x / 10);
18     putchar (x % 10 + '0');
19 }
```

7.4 gcd

```
1  ll gcd(ll x, ll y) { // 循环版
2      ll t;
3      while (y){
4          t = x % y;
5          x = y;
6          y = t;
7      }
8      return  x;
9  }
10
11 ll gcd(ll a, ll b) { // 递归版
12     return b == 0 ? a : gcd(b, a % b);
13 }
14
15 // 扩展欧几里得
16 ll exgcd(ll a, ll b, ll &x, ll &y) {
```

```
17          if (b == 0) {
18              x = 1, y = 0;
19              return a;
20          }
21          ll q = exgcd(b, a % b, y, x);
22          y -= a / b * x;
23          return q;
24      }
```

7.5   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 if 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   }
```

7.6   Permutation

```
1    // 错排问题
2    // D(n) = n! [(-1)^2/2! + ⋯ + (-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    }
```

7.7   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, e = prime_max / i;
```

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