# 目录

## DS

### 杰哥和数字

```
#include <iostream>
#include <cmath>
using namespace std;
bool IsValid(int num, int digit[], int
digitCount)
{
  for (int i = 0; i < digitCount; i++)
  {
    int tempNum = num;
    while (tempNum != 0)
    {
      if (digit[i] == tempNum % 10)
      {
        return true;
      }
      tempNum /= 10;
    }
  }
  return false;
}
int main(void)
{
  // Input & Initialize
  int input;
  int ansCnt = 0;
  cin >> input;
  int digitArray[10], digitCount = 0;
  int numTemp = input;
  while (numTemp != 0)
  {
    int digitCache = numTemp % 10;
    // Check if repeated
    bool isRepeated = false;
    for (int i = 0; i < digitCount; i++)
    {
      if (digitCache == digitArray[i])
      {
        isRepeated = true;
        break;
      }
    }
    if (!isRepeated)
    {
      digitArray[digitCount++] =
digitCache;
    }
    numTemp /= 10;
  }
  int maxi = sqrt(input);
  for (int i = 1; i <= maxi; i++)
  {
    if (input % i == 0)
    {
      ansCnt += i != input / i ?
            IsValid(i, digitArray,
digitCount) + IsValid(input / i,
digitArray, digitCount)
          : IsValid(i, digitArray,
digitCount);
    }
  }

  cout << ansCnt;
  return 0;
}
```

### 素数区间

```
#include <iostream>
#include <cstring>
#define MAXN 100004
using namespace std;


// Linear Sieve
int prime[MAXN];
int primeCount = 0;
bool isPrime[MAXN];
int primeTableCapacity = 0;

void LinearPrimeSieve(int n)
{
  if (primeTableCapacity >= n)
  {
    return;
  }
  int startNum = primeTableCapacity ?
prime[primeCount - 1] + 1 : 2;
  for (int i = startNum; i < MAXN; i++)
  {
    if (isPrime[i])
    {
      prime[primeCount++] = i;
    }

    for (int j = 0; j < primeCount && i *
prime[j] <= MAXN; j++)
    {
      isPrime[i * prime[j]] = 0;
      if (i % prime[j] == 0)
      {
        break;
      }
    }
  }

  primeTableCapacity = n;
  return;
}

int BinarySearch(int num, int arr[], int
left, int right)
{
  int mid = (left + right) / 2;

  // Exit function
  if (right - left == 1)
  {
    return arr[right] - arr[left] - 1;
  }
  if (arr[mid] == num)
  {
    return 0;
  }


  if (arr[mid] < num)
  {
    return BinarySearch(num, arr, mid,
right);
  }
  else
  {
    return BinarySearch(num, arr, left,
mid);
  }
}

int main(void)
{
  // Input & Initialize
  memset(isPrime, true, sizeof(isPrime));
  int questCount;
  int input;
  cin >> questCount;

  for (int i = 0; i < questCount; i++)
  {
    cin >> input;
    LinearPrimeSieve(input * 2 <= MAXN ?
input * 2 : MAXN);

    cout << BinarySearch(input, prime, 0,
primeCount - 1) << endl;
  }


  return 0;
}
```

### 沟里学姐的残忍

```
#include <iostream>
int main(void)
{
  int n, n_max;
  scanf("%d %d", &n, &n_max);
  int arr[n];
  scanf("%d", &arr[0]);
  int ans = arr[0], temp_ans = arr[0];
  int tmp = 0;
  int s_i = 0, e_i = 0;
  for (int i = 1; i < n; i++)
  {
    arr[i] = read();
    while (temp_ans < 0 || i - tmp + 1 >
n_max || i - tmp > 0 && arr[tmp] < 0)
      temp_ans -= arr[tmp++];
    temp_ans += arr[i];
    if (temp_ans > ans)
    {
      s_i = tmp;
      e_i = i;
      ans = temp_ans;
    }
  }
  printf("%d %d %d", ans, s_i + 1, e_i +
1);
  return 0;
}
```

### 迷宫

```
#include <iostream>
#include <queue>
#include <cstring>
#define EMPTY '.'
#define WALL '#'
#define START 'S'
#define END 'E'
enum DIRECTION { UP, RIGHT, DOWN, LEFT };
struct Position { int x, y; };
using namespace std;
int main(void)
{
  int row, col;
  cin >> row >> col;
  // Read the map
  Position start, end;
  char charBuf;
  char map[row + 2][col + 2];
  memset(map, WALL, sizeof(map));
  for (int i = 1; i <= row; i++)
  {
    getchar();
    for (int j = 1; j <= col; j++)
    {
      scanf("%c", &charBuf);
      map[i][j] = charBuf;
      if (charBuf == START)
      {
        start.x = i;
        start.y = j;
        map[i][j] = WALL;
      }
      if (charBuf == END)
      {
        end.x = i;
        end.y = j;
      }
    }
  }
  // BFS
  int ans[row + 2][col + 2];
  memset(ans, 0, sizeof(ans));
  queue<Position> checkPos;
  checkPos.push(start);
  while (!checkPos.empty())
  {
    Position check = checkPos.front();
    checkPos.pop();
    for (DIRECTION i = UP; i <= LEFT; i =
DIRECTION(i + 1))
    {
      Position tempPos = check;
      int befStep = ans[check.x][check.y];
      switch (i)
      {
        case UP:
          tempPos.x--;
          break;
        case RIGHT:
          tempPos.y++;
          break;
        case DOWN:
          tempPos.x++;
          break;
        case LEFT:
          tempPos.y--;
          break;
      }
      if (map[tempPos.x][tempPos.y] != WALL
&& ans[tempPos.x][tempPos.y] == 0)
      {
        ans[tempPos.x][tempPos.y] = befStep
+ 1;
        checkPos.push(tempPos);
      }
    }
    if (ans[end.x][end.y] != 0)
    {
      cout << ans[end.x][end.y];
      break;
    }
  }
  if (ans[end.x][end.y] == 0)
    cout << -1;
  return 0;
}
```

## 幸运儿

```
#include <iostream>
#include <vector>
#include <list>
using namespace std;
int main(void)
{
  int pepCount, opCount;
  scanf("%d %d", &pepCount, &opCount);
  list<int> pepArr;
  for (int i = 0; i < pepCount; i++)
  {
    int input;
    scanf("%d", &input);
    pepArr.push_back(input);
  }
  int opArr[opCount];
  for (int i = 0; i < opCount; i++)
  {
    scanf("%d", &opArr[i]);
  }

  for (int i = 0; i < opCount; i++)
  {
    int k = 0;
    for (auto j = pepArr.begin(); j !=
pepArr.end(); k++)
    {
      if ((k + 1) % opArr[i] == 0)
      {
        j = pepArr.erase(j);
      }
      else
      {
        j++;
      }
    }
  }
  printf("%lld\n", pepArr.size());
  for (auto i = pepArr.begin(); i !=
pepArr.end(); i++)
  {
    printf("%d ", *i);
  }
  return 0;
}
```

## 最长递增子段

```
#include<bits/stdc++.h>
using namespace std;
int r[100090],l[100090],a[100090],ans,n;
int main(){
  scanf("%d",&n);
  for(int i=1;i<=n;++i){
    scanf("%d",&a[i]);
  }
  r[n]=1;
  l[1]=1;
  for(int i=2;i<=n;++i){
    if(a[i-1]<a[i])
    l[i]=l[i-1]+1;
    else
    l[i]=1;
  }
  for(int i=n-1;i;--i){
    if(a[i]<a[i+1]){
      r[i]=r[i+1]+1;
    }
    else
    r[i]=1;
  }
  for(int i=1;i<=n;++i){
    ans=max(ans,l[i]+1);
    ans=max(ans,r[i]+1);
    if(a[i+1]-a[i-1]>1)
    ans=max(ans,l[i-1]+r[i+1]+1);
  }
  printf("%d\n",ans);
}
```

## station

```
#include <iostream>
#include <stack>
using namespace std;
int main(void)
{
  // Initialize & Input
  int numCount;
  stack<int> input;
  cin >> numCount;
  int inputArray[numCount];
  int buf;
  for (int i = 0; i < numCount; i++)
  {
    cin >> buf;
    input.push(buf);
    inputArray[i] = buf;
  }
  // Simulate stack permutation
  stack<int> auxStack;
  bool ans = true;
  // i control the target element
  for (int i = numCount; i > 0; i--)
  {
    if (!auxStack.empty() && auxStack.top()
== i)
    {
      auxStack.pop();
      continue;
    }
    // push elements into auxilary stack
until found target element
    while (!input.empty() && input.top() !=
i)
    {
      auxStack.push(input.top());
      input.pop();
    }
    if (input.empty())
    {
      if (auxStack.top() == i)
        auxStack.pop();
      else
      {
        ans = false;
        break;
      }
    }
    else
    {
      if (input.top() == i)
      {
        input.pop();
        continue;
      }
    }
  }
  printf("%s", ans ? "YES\n" : "NO\n");
  return 0;
}
```

## circle

```
#include <iostream>
#include <cstring>
#include <stack>
using namespace std;
bool check(int *map, int n)
{
  stack<int> s;
  for (int i = 0; i < n; i++)
  {
    if (map[i] == 0) continue;
    if (map[i] > 0 || s.empty())
    {
      s.push(map[i]);
    }
    else
    {
      if (s.top() + map[i] == 0)
      {
        s.pop();
      }
      else
      {
        return false;
      }
    }
  }
  return s.empty() ? true : false;
}
int main(void)
{
  int queryCount;
  cin >> queryCount;
  int pntBuf, lnkBuf;
  int *linkMap;
  for (int i = 0; i < queryCount; i++)
  {
    scanf("%d %d", &pntBuf, &lnkBuf);
    linkMap = new int[pntBuf * 2];
    memset(linkMap, 0, sizeof(int) * pntBuf
* 2);
```

```
    // Initialize LinkMap
    int p1, p2;
    for (int j = 1; j <= lnkBuf; j++)
    {
      scanf("%d %d", &p1, &p2);
      if (p1 > p2)
        swap(p1, p2); // always start with
a positive num
      linkMap[p1 - 1] = j;
      linkMap[p2 - 1] = -j;
    }
    bool ans;
    // Push LinkMap into checkStack one by
one, check if conflict?
    printf("%s", check(linkMap, 2 *
pntBuf) ? "YES\n" : "NO\n");
    delete linkMap;
  }
  return 0;
}
```

## 简单的排序

```
#include <iostream>
#include <cstring>
#define MAX_NUM 200010
#define SHIFT 100000
using namespace std;
int main()
{
  int n;
  scanf("%d", &n);
  int bucket[MAX_NUM];
  memset(bucket, 0, sizeof(bucket));
  int buf;
  for (int i = 0; i < n; i++)
  {
    // scanf("%d", &buf);
    buf = read();
    bucket[buf + SHIFT]++;
  }
  bool isFirst = true;
  char output[10];
  for (int i = 0; i < MAX_NUM; i++)
  {
    if (bucket[i])
    {
      sprintf(output, "%d ", i - SHIFT);
      for (int j = 0; j < bucket[i]; j++)
      {
        char * tmp = output;
        while (*tmp != '\0')
putchar(*tmp++);
      }
    }
  }
}
```

## 小孩的游戏

```
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
#define MAX_CARD 110
int main(void)
{
  // init & input
  int n, buf;
  cin >> n;
  int bucket[MAX_CARD];
  memset(bucket, 0, sizeof(bucket));
  for (int i = 0; i < n; i++)
  {
    buf = read();
    bucket[buf]++;
  }
  // Priority generator
  int pri[101];
  int cnt = 0;
  for (int i = 9; i > 0; i--)
  {
    for (int j = 9; j >= 0; j--)
    {
      if (i == j) pri[cnt++] = i;
      pri[cnt++] = i * 10 + j;
    }
  }
  pri[99] = 100;
  pri[100] = 0;
  // output
  for (int i = 0; i <= 100; i++)
  {
    if (bucket[pri[i]])
    {
      for (int j = 0; j < bucket[pri[i]];
j++)
      {
        cout << pri[i];
      }
    }
  }
  return 0;
}
```

## jiaoshuicishu

```
#include <iostream>
#include <vector>
#define MAX_NUM 100010
using namespace std;
struct node
{
  int to;
  bool status;
};
vector<node> map[MAX_NUM];
int visit[MAX_NUM];
int dfs(int nodeIndex)
{
  visit[nodeIndex] = 1;
  int ret = 0;
  // traverse subNode
  for (int i = 0; i <
map[nodeIndex].size(); i++)
  {
    int childNodeIndex =
map[nodeIndex][i].to;
    if (!visit[childNodeIndex])
    {
      // 每个节点的子节点中有多少条没水的渠道
      ret +=
max((int)map[nodeIndex][i].status,
dfs(childNodeIndex));
    }
  }
  return ret;
}
inline int read()
{
  int ret = 0, sign = 1;
  char ch = getchar();
  while (ch < '0' || ch > '9')
  {
    if (ch == '-')
      sign = -1;
    ch = getchar();
  }
  while (ch >= '0' && ch <= '9')
  {
    ret = ret * 10 + (ch - '0');
    ch = getchar();
  }
  return ret * sign;
}

int main()
{
  int n, i;
  int x, y, s;
  n = read();
  for (i = 0; i < n - 1; i++)
  {
    x = read();
    y = read();
    s = read();
    map[x].push_back({y, s == 2});
    map[y].push_back({x, s == 2});
  }
  cout << dfs(1);
  return 0;
}
```

## shoufei

```
#include <iostream>
#include <cstring>
#define TREE_SIZE 1000010
using namespace std;
int main(void)
{
  int queryCount = read();
  long long *weight = new long
long[TREE_SIZE];
  memset(weight, 0, TREE_SIZE * 8);
  int opBuf, fromBuf, toBuf, weightBuf;
  long long ans;
  for (int i = 0; i < queryCount; i++)
  {
    opBuf = read();
    fromBuf = read();
    toBuf = read();
    switch (opBuf)
    {
    case 1:
      weightBuf = read();
      while (fromBuf != toBuf)
      {
        if (fromBuf < toBuf)
        {
          weight[toBuf] += weightBuf;
          toBuf /= 2;
        }
        else
        {
          weight[fromBuf] += weightBuf;
          fromBuf /= 2;
        }
      }
      break;
    case 2:
      ans = 0;
      while (fromBuf != toBuf)
      {
        if (fromBuf < toBuf)
        {
          ans += weight[toBuf];
          toBuf /= 2;
        }
        else
        {
          ans += weight[fromBuf];
          fromBuf /= 2;
        }
      }
      cout << ans << '\n';
      break;
    }
  }
  return 0;
}
```

## 集合运算

```
#include <iostream>
#include <algorithm>
#include <set>
#include <vector>
using namespace std;
void InitSet(set<int> &s, int cnt)
{
  for (int i = 0; i < cnt; i++)
    s.insert(read());
  return;
}
int main()
{
  set<int> a, b, c;
  int cntA = read(), cntB = read(), cntC =
read();
  InitSet(a, cntA);
  InitSet(b, cntB);
  InitSet(c, cntC);

  set<int> setBuf;
  // union
  setBuf.clear();
  set_union(a.begin(), a.end(), b.begin(),
b.end(), inserter(setBuf,
setBuf.begin()));
  if (c == setBuf)
  {
    for (auto i : setBuf)
      cout << i << " ";
    return 0;
  }
  // intersect
  setBuf.clear();
  set_intersection(a.begin(), a.end(),
b.begin(), b.end(), inserter(setBuf,
setBuf.begin()));
  if (c == setBuf)
  {
    for (auto i : setBuf)
      cout << i << " ";
    return 0;
  }
  // diff
  setBuf.clear();
  set_difference(a.begin(), a.end(),
b.begin(), b.end(), inserter(setBuf,
setBuf.begin()));
  if (c == setBuf)
  {
    for (auto i : setBuf)
      cout << i << " ";
    return 0;
  }
  setBuf.clear();
  set_difference(b.begin(), b.end(),
a.begin(), a.end(), inserter(setBuf,
setBuf.begin()));
  if (c == setBuf)
  {
    for (auto i : setBuf)
      cout << i << " ";
    return 0;
  }
  cout << "What a pity!";
  return 0;
}
```

## 三角形游戏

```
#include <iostream>
#include <bitset>
#include <map>
#include <unordered_set>
using namespace std;
uint32_t HashTriplet(int a, int b, int c)
{
  // sort abc
  if (a > b) swap(a, b);
  if (b > c) swap(b, c);
  if (a > b) swap(a, b);
  // join abc into a int32
  uint32_t ret = a + (b << 8) + (c << 16);
  return ret;
}
inline int read();
int main(void)
{
  map<uint32_t, int> triSet;
    map<uint32_t, int>::iterator it;
  uint32_t hashBuf;
  // 1. input
  int n;
  cin >> n;
  int edge[3];
  for (int i = 0; i < n; i++)
  {
    edge[0] = read();
    edge[1] = read();
    edge[2] = read();
    triSet[HashTriplet(edge[0], edge[1],
edge[2])] += 1;
  }
  // 2. query
  int queryCount;
  cin >> queryCount;
  for (int i = 0; i < queryCount; i++)
  {
    edge[0] = read();
    edge[1] = read();
    edge[2] = read();
    hashBuf = HashTriplet(edge[0], edge[1],
edge[2]);
    it = triSet.find(hashBuf);
    if (it != triSet.end())
    {
      printf("%d\n", it->second);
    }
    else
    {
      printf("0\n");
    }
  }
}
```

## jihe

```
#include <iostream>
#include <set>
using namespace std;
inline int read();
int main()
{
  set<int> search;
  set<int>::iterator floor, ceil;
  int queryCount = read(), opBuf, opNum;
  search.insert(0);
  for (int i = 0; i < queryCount; i++)
  {
    opBuf = read();
    opNum = read();
    if (opBuf == 1)
      search.insert(opNum);
    if (opBuf == 2)
    {
      ceil = search.lower_bound(opNum);
      floor = --search.lower_bound(opNum);
      if (opNum == *ceil && ceil !=
search.end())
        cout << *ceil;
      else
      {
        bool isLeftNull = *floor == 0;
        bool isRightNull = ceil ==
search.end();
        if (isLeftNull && isRightNull)
          printf("Empty!");
        if (isLeftNull && !isRightNull)
          printf("%d", *ceil);
        if (!isLeftNull && isRightNull)
          printf("%d", *floor);
        if (!isLeftNull && !isRightNull)
        {
          if (abs(*floor - opNum) <
abs(*ceil - opNum))
          {
            printf("%d", *floor);
          }
          else if (abs(*floor - opNum) >
abs(*ceil - opNum))
            printf("%d", *ceil);
          else
            printf("%d %d", *floor, *ceil);
        }
      }
      putchar('\n');
    }
  }
  return 0;
}
```

## road

```
#include <iostream>
#include <stack>
#include <vector>
using namespace std;
inline int read();
struct TreeNode
{
  int data;
  TreeNode *left = nullptr;
  TreeNode *right = nullptr;
  TreeNode *parent = nullptr;
  int height = 1;
  int factor()
  {
    if (left == nullptr && right ==
nullptr)
      return 0;
    else if (left == nullptr)
      return right->height;
    else if (right == nullptr)
      return - left->height;
    else
      return (right->height -
left->height);
  }
  void update_height()
  {
    if (left == nullptr && right ==
nullptr)
      height = 1;
    else if (left == nullptr)
      height = right->height + 1;
    else if (right == nullptr)
      height = left->height + 1;
    else
      height = max(left->height,
right->height) + 1;
  }
};
// Rename TreeNode, when adjust tree need
modify a pointer of a pointer.
typedef TreeNode * TreeNodePtr;
class AVLTree
{
  TreeNode *root = nullptr;
  void p_adjust(TreeNodePtr &node)
  {
    int nodeBlncFactor = node->factor();
    if (nodeBlncFactor < -1)
    {
      if (node->left->factor() < 0)
    node = p_RotateRight(node);
   else
   {
    node->left =
p_RotateLeft(node->left);
    node = p_RotateRight(node);
   }
  }
  else if (nodeBlncFactor > 1)
  {
   if (node->right->factor() > 0)
    node = p_RotateLeft(node);
   else
   {
    node->right =
p_RotateRight(node->right);
    node = p_RotateLeft(node);
   }
  }
  else
   return;
 }
 TreeNode *p_RotateLeft(TreeNode *node)
 {
  TreeNode *succ = node->right;
  node->right = succ->left;
  succ->left = node;
  succ->parent = node->parent;
  node->parent = succ;
  node->update_height();
  succ->update_height();

  return succ;
 }
 TreeNode *p_RotateRight(TreeNode *node)
 {
  TreeNode *succ = node->left;
  node->left = succ->right;
  succ->right = node;
  succ->parent = node->parent;
  node->parent = succ;
  node->update_height();
  succ->update_height();
  return succ;
 }
public:
 void insert(const int item)
 {
  stack<TreeNodePtr *> path;
  // Step 1. Find Node & Insert
  TreeNodePtr *scan = &root;   // pointer
to scan the tree
  TreeNodePtr *prev = nullptr; // pointer
to save the node previous to pointer[scan]
  while (*scan)
  {
   path.push(scan); // record the node
visited.
   prev = scan;
   scan = item < (*scan)->data ?
&(*scan)->left : &(*scan)->right;
  }
  // Create a new node and initialize it
by item and previous node.
  TreeNode *newNode = new TreeNode;
  newNode->data = item;
  newNode->parent = prev ? *prev :
nullptr;
  if ((prev ? *prev : nullptr) !=
nullptr)
   (item < (*prev)->data ?
(*prev)->left : (*prev)->right) = newNode;
  else
   root = newNode;
  // Step 2. Check Path
  while (!path.empty())
  {
   // take one node out
   scan = path.top();
   path.pop();
   (*scan)->update_height();
   p_adjust(*scan);
  }
 }
 vector<int> find_path(const int item)
 {
  vector<int> path;
  TreeNode *scan = root;  // pointer to
scan the tree
  TreeNode *prev = nullptr; // pointer to
save the node previous to pointer[scan]
  while (scan && scan->data != item)
  {
   path.push_back(scan->data); // record
the node visited.
   prev = scan;
   scan = item < scan->data ?
scan->left : scan->right;
  }
  path.push_back(item);
  return path;
 }
 void find_path_out(const int item)
 {
  TreeNode *scan = root;  // pointer to
scan the tree
  TreeNode *prev = nullptr; // pointer to
save the node previous to pointer[scan]
  while (scan && scan->data != item)
  {
   cout << scan->data << " ";
   prev = scan;
   scan = item < scan->data ?
scan->left : scan->right;
  }
  cout << item;
  return;
 }
};
int main(void)
{
 AVLTree tree;
 int opCount = read(), opBuf;
 for (int i = 0; i < opCount; i++)
 {
  opBuf = read();
  if (opBuf == 1)
   tree.insert(read());
  if (opBuf == 2)
  {
   tree.find_path_out(read());
   putchar('\n');
  }
 }
 return 0;
}
```

## 分智慧果

```
#include <iostream>
using namespace std;
// Big heap
class BinaryHeap
{
 int *_heap = nullptr;
 int _size = 0;
 int _capacity = 0;
 int _head;
 int _Parent(int x) { return ((x - 1) >>
1); }
 int _LChild(int x) { return ((x << 1) +
1); }
 int _RChild(int x) { return ((x << 1) +
2); }
public:
 BinaryHeap(int c = 100000) : _capacity(c)
{ _heap = new int[c + 100]; }
 void init(int h) { _head = h; }
 void insert(int n)
 {
  _heap[_size++] = n;
  int scan = _size - 1;
  int buf;
  while ((buf = _Parent(scan)) >= 0)
  {
   if (_heap[scan] < _heap[buf])
    break;
   swap(_heap[scan], _heap[buf]);
   scan = buf;
  }
 }
 int solve()
 {
  int ret = 0;
  while (_head < _heap[0])
  {
   _heap[0]--; _head++; ret++;
   for (int scan = 0, buf = 1; buf <
_size; buf = _LChild(scan))
   {
    if (buf + 1 < _size && _heap[buf] <
_heap[buf + 1])
     buf++;
    if (_heap[scan] > _heap[buf])
     break;
    swap(_heap[scan], _heap[buf]);
    scan = buf;
   }
  }
  return ret;
 }
};
int main(void)
{
 int numCount = read();
 BinaryHeap heap(numCount);
 heap.init(read());
 for (int i = 1; i < numCount; i++)
  heap.insert(read());
 cout << heap.solve();
 return 0;
}
```

## 森林冰火人

```
#include<bits/stdc++.h>
using namespace std;
int v[100000], t[100000];
long long s = 0, e = 0, sum = 0,
ans[100000];
int main()
{
  int n;
  scanf("%d", &n);
  priority_queue<long long,vector<long
long>,greater<long long> > q;
  for(int i = 0; i < n; i++)scanf("%d",
&v[i]);
  for(int i = 0; i < n; i++)scanf("%d",
&t[i]);
  for(int i = 0; i < n; i++){
    q.push(v[i] + s);
    e += s, s += t[i];
    while(q.size() && q.top() <= s)
    {
      sum += q.top();
      q.pop();
    }
    ans[i] = sum + q.size() * s - e;
  }
  for(int i = n; i; i--) ans[i] -= ans[i
- 1];
  for(int i = 0; i < n; i++) printf("%lld
", ans[i]);
}
```

## 朋友圈

```
#include <iostream>
#include <cstring>
#pragma GCC optimize(2)
using namespace std;
inline int read();
class DisjointSet
{
  int *_parent;
  int _size;
  int _max = 1;

  int _findRoot_compress(int index)
  {
    // Compress all the nodes which route
will pass by,
    int root = index;
    while (_parent[root] > 0)
      root = _parent[root];
    while (_parent[index] > 0)
    {
      _parent[index] = root;
      index = _parent[index];
    }
    return root;
  }
public:
  DisjointSet(int s) : _size(s)
  {
    _parent = new int[s + 100];
    memset(_parent, -1, sizeof(int) * (s +
100));
  }
  int max() { return _max; }
  void merge(int a, int b)
  {
    int rootA = _findRoot_compress(a),
rootB = _findRoot_compress(b);
    if (rootA == rootB)
      return;
    if (_parent[rootA] < _parent[rootB])
      swap(rootA, rootB);
    _parent[rootA] += _parent[rootB];
    _parent[rootB] = rootA;
    // Update max value
    _max = -_parent[rootA] > _max ? -
_parent[rootA] : _max;

    return;
  }
};
int main(void)
{
  int friendCount = read(), opCount =
read();
  int x, y;
  DisjointSet relation(friendCount);
  for (int i = 0; i < opCount; i++)
  {
    x = read();
    y = read();
    relation.merge(x, y);
  }
  cout << relation.max();
  return 0;
}
```

## 水杯

```
#include <iostream>
#include <cstring>
typedef long long ll;
using namespace std;
inline ll read();
class DisjointSet
{
  int *_parent;
  ll *_cupCapacity;
  ll *_volume;
  int _size;
  int _findRoot(int index)
  {
    while (_parent[index] > 0)
      index = _parent[index];
    return index;
  }
public:
  DisjointSet(int s) : _size(s)
  {
    _parent = new int[s + 100];
    _cupCapacity = new ll[s + 100];
    _volume = new ll[s + 100];
    memset(_parent, -1, sizeof(int) * (s +
100));
  }
  void initCapacity()
  {
    for (int i = 1; i <= _size; i++)
    {
      _volume[i] = 0;
      _cupCapacity[i] = read();
    }
    _volume[_size] = 0;
  }
  void water(int cup, int vol)
  {
    int i;
    for (i = cup; _volume[i] + vol >
_cupCapacity[i] && i <= _size; i =
_parent[i])
    {
      vol = _volume[i] + vol -
_cupCapacity[i];
      _volume[i] = _cupCapacity[i];
      _parent[i] = _findRoot(i + 1);
    }
    if (i != _size + 1)
      _volume[i] += vol;
  }
  ll volume(int index) { return
_volume[index]; }
};
int main(void)
{
  int cupCount = read();
  DisjointSet cupSet(cupCount);
  cupSet.initCapacity();
  int opCount = read();
  long long num1, num2;
  for (int i = 0; i < opCount; i++)
  {
    if (read() == 1)
    {
      num1 = read();
      num2 = read();
      cupSet.water(num1, num2);
    }
    else
      cout << cupSet.volume(read()) <<
'\n';
  }
  return 0;
}
```

## 寡人的难题

```
#include <iostream>
#include <vector>
#include <numeric>
using namespace std;
inline int read();
class DisjointSet
{
  vector<int> _parent, _size;
public:
  DisjointSet(int s) : _parent(s), _size(s,
1) { iota(_parent.begin(), _parent.end(),
0); }
  int find(int x) { return _parent[x] ==
x ? x : _parent[x] = find(_parent[x]); }
  bool unite(int x, int y)
  {
    x = find(x), y = find(y);
    if (x == y) return false;
    if (_size[x] < _size[y]) swap(x, y);
    _parent[y] = x;
    _size[x] += _size[y];
    return true;
  }
};
struct Edge {int from, to;};
#define WEIGHT_MAX 10010
int main(void)
{
  int n = read();
  int m = read();
  int u, v, w, max = 0;
  // 存储图 & 按权重排序边
  // 用计数排序的思想，将边的权重作为 edgeSet
的下标，在输入的过程中完成自然排序.
  vector<Edge> edgeSet[WEIGHT_MAX];
  for (int i = 0; i < m; i++)
  {
    u = read(); v = read(); w = read();
    edgeSet[w].push_back({u - 1, v - 1});
  }
  int ans = 0;
  // 并查集记录已连接的顶点.
  DisjointSet mst(n);
  // 从权重为 0 的边开始遍历，当已经加入 n - 1
条边或
  for (int i = 0, cnt = 1; cnt < n; i++)
    for (auto j : edgeSet[i])
      if (mst.unite(j.from, j.to))
      {
        ans += i;
        cnt++;
      }
  cout << ans;
  return 0;
}
```

## 旅行二

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
#include <vector>
using namespace std;
#define MAX_SIZE 50010
#define INF (1 << 29)
struct Graph
{
  vector<int> adj;
  vector<int> w;
  void add_edge(int to, int weight)
{adj.push_back(to); w.push_back(weight);}
} g[MAX_SIZE];
class AdjoinSet : public pair<long long,
int>
{
public:
  AdjoinSet() = default;
  AdjoinSet(long long a, int b) : pair<long
long, int>(a, b) {}
  friend bool operator<(const AdjoinSet
&p1, const AdjoinSet &p2){ return
p1.first > p2.first;}
};
  long long dis[MAX_SIZE];
  int vis[MAX_SIZE];
```

```
long long Dijkstra(int n)
{
  priority_queue<AdjoinSet,
vector<AdjoinSet>> queue;
  fill(dis, dis + n + 10, INF);
  dis[0] = 0;
  queue.push({dis[0], 0});

  AdjoinSet verBuf;
  while (!queue.empty())
  {
    int ind = queue.top().second;
    if (vis[ind] != 1)
    {
      vis[ind] = 1;
      for (vector<int>::iterator it =
g[ind].adj.begin(); it !=
g[ind].adj.end(); it++)
      {
        if (vis[*it] == 0 && dis[*it] >
dis[ind] + g[ind].w[it -
g[ind].adj.begin()])
        {
          dis[*it] = dis[ind] + g[ind].w[it
- g[ind].adj.begin()];
          queue.push({dis[*it], *it});
        }
      }
    }
    queue.pop();
  }
  return dis[n + 1];
}
inline int read();
int main()
{
  int vertexCount, bonusCount, edgeCount,
goalCount, u, v, w;
  cin >> vertexCount >> edgeCount >>
bonusCount >> goalCount;
  for (int i = 0; i < edgeCount; i++)
  {
    u = read();
    v = read();
    w = read();
    g[u].add_edge(v, w);
    g[v].add_edge(u, w);
  }
  for (int i = 1; i <= bonusCount; i++)
  {
    u = read();
    g[0].add_edge(u, 0);
    g[u].add_edge(0, 0);
  }
  for (int i = 1; i <= goalCount; i++)
  {
    u = read();
    g[vertexCount + 1].add_edge(u, 0);
    g[u].add_edge(vertexCount + 1, 0);
  }
  cout << Dijkstra(vertexCount);
  return 0;
}
```

## 团体程序设计天梯赛

### L1-001 Hello World

### L1-002 打印沙漏

```
#include <stdio.h>
#include <math.h>

void sandclock(char ch, int line, int
spaceCount);
void repeatprint(int times, char ch);

int main(void)
{
  int input;
  char ch;
  scanf("%d %c", &input, &ch);
  int line = sqrt((input + 1) / 2);
  //calculate the lines require
  sandclock(ch, line, 0);
  printf("%d", input - (line * line * 2 -
1)); //remain characters
  return 0;
}

void repeatprint(int times, char ch)
{
  for (int i = 0; i < times; i++)
    putchar(ch);
}

void sandclock(char ch, int line, int
spaceCount)
{
  int starCount = (line * 2) - 1;

  if (line > 1)
  {
    repeatprint(spaceCount, ' ');
    repeatprint(starCount, ch);
    putchar('\n');

    sandclock(ch, line - 1, spaceCount +
1);
  }

  repeatprint(spaceCount, ' ');
  repeatprint(starCount, ch);
  putchar('\n');
}
```

### L1-003 个位数统计

### L1-004 计算摄氏温度

### L1-005 考试座位号

```
#include <stdio.h>
typedef struct
{
  char no[17];
  int Tseat;
  int Eseat;
} INFO;
int main(void)
{
  int stuCount;
  scanf("%d", &stuCount);
  INFO stuInfo[stuCount];

  for (int i = 0; i < stuCount; i++)
  {
    scanf("%s %d %d", stuInfo[i].no,
&stuInfo[i].Tseat, &stuInfo[i].Eseat);
  }

  int queryCount;
  scanf("%d", &queryCount);
  int queryArray[queryCount];
  for (int i = 0; i < queryCount; i++)
  {
    scanf("%d", &queryArray[i]);
  }

//  int outputIndex;
  for (int i = 0; i < queryCount; i++)
  {
    for (int j = 0; j < stuCount; j++)
    {
      if (stuInfo[j].Tseat ==
queryArray[i])
      {
        //outputIndex = i;
        printf("%s %d\n", stuInfo[j].no,
stuInfo[j].Eseat);
      }
    }

  }


  return 0;
}
```

### L1-006 连续因子

```
#include <iostream>
#include <cmath>
using namespace std;
bool isPrime(const int n)
{
  int max = sqrt(n) + 1;
  for (int i = 2; i <= max; i++)
    if (n % i == 0)
      return false;
  return true;
}
int main(void)
{
  int inputNum;
  cin >> inputNum;
  int max = sqrt(inputNum) + 1;
  int ansOffset = 2;
  int ansCount = 0;
  if (isPrime(inputNum))
  {
    ansCount = 1;
    ansOffset = inputNum;
  }
  // Method: enumeration
  // Each iteration starts by inputNum,
every iteration ++ansOffset
  // try every possible num, until a
certain num isn't a factor of inputNum
  // Compare them to ans in memory, if
bigger than them. update.
  for (int i = ansOffset; i <= max; i++)
  {
    int tempNum = inputNum;
    int tempCount = 0;

    for (int j = i; j <= max; j++)
    {
      // Make sure j is a factor of tempNum
FIRST
      if (tempNum % j != 0)
        break;
      tempCount++;
      tempNum /= j;
    }
    // Update answer
    if (ansCount < tempCount)
    {
      ansCount = tempCount;
      ansOffset = i;
    }
  }
  // Print answer
  printf("%d\n", ansCount);
  for (int i = 0; i < ansCount; i++)
    printf("*%d" + !i, ansOffset + i);

  return 0;
}
```

## L1-007 念数字

## L1-008 求整数段和

```c
#include <stdio.h>
#define COLUMN 5
int main(void)
{
  int start, end;
  scanf("%d %d", &start, &end);

  int numCount = end - start + 1;
  int numSum = 0;
  for (int i = 0; i < numCount; i++)
  {
    if (i % COLUMN == 0 && i != 0)
    {
      putchar('\n');
    }
    printf("%5d", start);
    numSum += start;
    start++;
  }
  putchar('\n');

  printf("Sum = %d", numSum);

  return 0;
}
```

## L1-009 N 个数求和

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>
typedef struct
{
    int nume;
    int deno;
} FRAC;
FRAC FracAdd(FRAC A, FRAC B);
int main(void)
{

    int fracCount;
    scanf("%d", &fracCount);
    FRAC input;
    FRAC result = {0, 1};
    for (int i = 0; i < fracCount; i++)
    {
        scanf("%d/%d", &input.nume,
&input.deno);
        result = FracAdd(result, input);
    }


    int output = result.nume / result.deno;
    if (result.nume % result.deno != 0)
    {
    if (result.nume > result.deno)
      printf("%d %d/%d", output,
result.nume - output * result.deno,
result.deno);
    else
      printf("%d/%d", result.nume,
result.deno);
    }
    else
        printf("%d", output);
    return 0;
}

int GCD(int m, int n)
{
    //swap
    //make sure m > n
    int temp;
    if (m < n)
    {
        temp = m;
        m = n;
        n = temp;
    }
    while (n != 0)
    {
        temp = m % n;
        m = n;
        n = temp;
    }
    return m;
}
int LCM(int m, int n)    //using GCD()
{
    int temp = GCD(m, n);
    return m * n / temp;
}

FRAC FracAdd(FRAC A, FRAC B)
{
    FRAC tempFrac;
    tempFrac.nume = A.nume * B.deno +
B.nume * A.deno;
    tempFrac.deno = A.deno * B.deno;
    int tempVar = GCD(tempFrac.nume,
tempFrac.deno);
    tempFrac.nume /= tempVar;
    tempFrac.deno /= tempVar;

    return tempFrac;
}
```

## L1-010 比较大小

## L1-011 A-B

```c
#include <stdio.h>
#include <string.h>
#define MAXLENGTH 10001
#define DELNUM 1
int main(void)
{
  char input[MAXLENGTH];
  gets(input);
  int inputLength =strlen(input);


  char del[MAXLENGTH];
  gets(del);

  for (int i = 0; i < inputLength; i++)
  {
    if (strchr(del, input[i]) != NULL)
    {
      input[i] = DELNUM;
    }
  }

  //output
  for (int i = 0; i < inputLength; i++)
  {
    if (input[i] != DELNUM)
    {
      putchar(input[i]);
    }
  }

  return 0;
}
```

## L1-012 计算指数

## L1-013 计算阶乘和

## L1-014 简单题

## L1-015 跟奥巴马一起画方块

## L1-016 查验身份证

```c
#include <stdio.h>
#include <stdbool.h>
int main(void)
{
    int idCount;
    scanf("%d", &idCount);
    int weightArray[17] = {7, 9, 10, 5, 8,
4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2};
    char checkArray[11] = {'1', '0', 'X',
'9', '8', '7', '6', '5', '4', '3', '2'};
    char idArray[idCount][19];
  bool isValid[idCount];
  bool isAllPass = true;


    int sum = 0;
    for (int i = 0; i < idCount; i++)
    {
        scanf("%s", idArray[i]);

        for (int j = 0; j < 17; j++)
        {
            sum += (idArray[i][j] - '0') *
weightArray[j];
        }

  // printf("%d\n", sum % 11)
        if (idArray[i][17] ==
checkArray[sum % 11])
    {
      isValid[i] = true;
    }
    else
    {
      isValid[i] = false;
      isAllPass = false;
    }
    sum = 0;

    }

  if (!isAllPass)
  {
    for (int i = 0; i < idCount; i++)
      if (isValid[i] == false)
        printf("%s\n", idArray[i]);
  }
  else
    printf("All passed");


    return 0;
}
```

## L1-017 到底有多二

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#define MAXLENGTH 51

int main(void)
{
  char input[MAXLENGTH];
  scanf("%s", input);

  bool isNegative = input[0] == '-' ?
true : false;
  int numLength = strlen(input);
  bool isEven = ((input[numLength - 1] -
'0') % 2 == 0) ? true : false;

  int twoCount = 0;
  for (int i = 0; i < numLength; i++)
  {
    if (input[i] == '2')
    {
      twoCount++;
    }
  }

  numLength -= isNegative ? 1 : 0;
  double twoRatio = (double)twoCount /
(double)numLength;
  twoRatio *= isNegative ? 1.5 : 1;
  twoRatio *= isEven ? 2 : 1;

  printf("%.2lf%%", twoRatio * 100);


  return 0;
}
```

## L1-018 大笨钟

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    int hour, min;
    scanf("%d:%d", &hour, &min);
    int timeInMinutes = hour * 60 + min;
    if (timeInMinutes <= 720)
    {
        printf("Only %02d:%02d.  Too early
to Dang.", hour, min);
        return 0;
    }
    int dangCount = ceil(timeInMinutes /
60.0) - 12;
    for (int i = 0; i < dangCount; i++)
    printf("Dang");
    return 0;
}
```

## L1-019 谁先倒

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
int WhoWinGame(void);
int main(void)
{
  int aMaxDrink, bMaxDrink;
  scanf("%d %d", &aMaxDrink, &bMaxDrink);

  int guessCount;
  scanf("%d", &guessCount);
  int aDrinkCount = 0;
  int bDrinkCount = 0;
  int status;
  for (int i = 0; i < guessCount; i++)
  {
    status = WhoWinGame();
    if (status == -1)
      bDrinkCount++;
    else if (status == 1)
      aDrinkCount++;
    else
      continue;

    if (aDrinkCount > aMaxDrink)
    {
      printf("A\n%d", bDrinkCount);
      break;
    }
    if (bDrinkCount >bMaxDrink)
    {
      printf("B\n%d", aDrinkCount);
      break;

    }
  }
   return 0;
}

//-1 a
//0 draw
//1 b
int WhoWinGame(void)
{
  int aSay, aShow, bSay, bShow;
  scanf("%d %d %d %d", &aSay, &aShow,
&bSay, &bShow);

  int saySum = aSay + bSay;
  bool isAWin = aShow != saySum ? true :
false;
  bool isBWin = bShow != saySum ? true :
false;

  if ((isAWin && isBWin) || (!isAWin
&& !isBWin))
  {
    return 0;
  }
  else if (isAWin && !isBWin)
  {
    return -1;
  }
  else if (!isAWin && isBWin)
  {
    return 1;
  }
}
```

## L1-020 帅到没朋友

```
#include <stdio.h>
#include <stdbool.h>

int main(void)
{
  int dataCount;
  scanf("%d", &dataCount);

  int inputIdCount;
  int input;
  bool idList[100000] = {false};
  for (int i = 0; i < dataCount; i++)
  {
    scanf("%d", &inputIdCount);
    for (int j = 0; j < inputIdCount; j++)
    {
      scanf("%d", &input);

      if (inputIdCount != 1)
      {
        idList[input] = true;
      }
    }
  }

  int queryCount;
  int outputCount = 0;

  scanf("%d", &queryCount);
  for (int i = 0; i < queryCount; i++)
  {
    scanf("%d", &input);
    if (idList[input] == false)
    {
      if (!outputCount)
        printf("%05d", input);
      else
        printf(" %05d", input);

      outputCount++;
      idList[input] = true;
    }
  }

  if (!outputCount)
    printf("No one is handsome");

  return 0;
}
```

## L1-021 重要的话说三遍

## L1-022 奇偶分家

## L1-023 输出 GPLT

```
#include <stdio.h>
#include <string.h>
int main(void)
{
  int GPLTCount[4] = {0};
  char check[] = "GgPpLlTt";

  char tempch;
  int ttlCount = 0;
  while ((tempch = getchar()) != '\n')
  {
    if (strchr(check, tempch) != NULL)
    {
      GPLTCount[(strchr(check, tempch) -
check)/ 2]++;
      ttlCount++;
    }

  }

  char output[] = "GPLT";
  for (int i = 0; i < ttlCount; i++)
  {
    if (GPLTCount[i % 4] != 0)
    {
      putchar(output[i % 4]);
      GPLTCount[i % 4]--;
    }
    else
    {
      ttlCount++;
    }
  }


  return 0;
}
```

## L1-024 后天

## L1-025 正整数 A+B

```
#include <bits/stdc++.h>
using namespace std;
bool isLegal(string s)   //判断字符串是不是
一个在[1,1000]内的正整数
{
    for(auto it : s)
    {
        if(it < '0' || it > '9')  //判断每个
字符是不是正整数即可
        {
            return false;
        }
    }
    int temp = atoi(s.c_str());   //强制把
string 型转换成 char*型再变成 int 型
    if(temp < 1 || temp > 1000)   //超出
[1,1000]这个范围的数字非法
    {
        return false;
    }
    return true;
}
int main()
{
    string A,B;
    cin >> A;
    //不能直接用 cin >> A >> B; 第二个字符串
有空格时导致有个测试用例 WA
    getchar();  //吃回车
    getline(cin,B);   //第二个字符串中可能有
空格,用 getline()读取
    int a = atoi(A.c_str());
    int b = atoi(B.c_str());
    if(isLegal(A) && isLegal(B))
    {
        printf("%d + %d = %d\n",a,b,a+b);
    }
    else if(!isLegal(A) && isLegal(B))  //
若 A 非法、B 合法
    {
        printf("? + %d = ?\n", b);
    }
    else if(isLegal(A) && !isLegal(B))  //
若 A 合法、B 非法
    {
        printf("%d + ? = ?\n", a);
    }
    else   //若 A、B 都非法
    {
        printf("? + ? = ?\n");
    }
    return 0;
}
```

## L1-026 I Love GPLT

## L1-027 出租

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int FindDiffDig(char *dest, char input[],
int len);
void BubbleSort(char numArray[], int
numCount);
int main(void)
{
  char phoneNum[12];
  gets(phoneNum);
  char digArray[12];
  memset(digArray, '\0', 12);
  int digCount = FindDiffDig(digArray,
phoneNum, 11);

  BubbleSort(digArray, digCount);

  int output[11];
  for (int i = 0; i < 11; i++)
  {
    output[i] = strchr(digArray,
phoneNum[i]) - digArray;
  }
  //output
  printf("int[] arr = new int[]{%d",
digArray[0] - '0');
  for (int i = 1; i < digCount; i++)
  {
    printf(",%d", digArray[i] - '0');
  }
  printf("};\n");

  printf("int[] index = new int[]{%d",
output[0]);
  for (int i = 1; i < 11; i++)
  {
    printf(",%d", output[i]);
  }
  printf("};\n");

  return 0;
}
int FindDiffDig(char *dest, char input[],
int len)
{
  int digitCount = 0;
  for (int i = 0; i < len; i++)
  {
    if (strchr(dest, input[i]) == NULL)
    {
      dest[digitCount] = input[i];
      digitCount++;
    }
  }
  return digitCount;
}
void CharSwap(char *a, char *b)
{
  char temp = *a;
  *a = *b;
  *b = temp;
}
//bubble sort
void BubbleSort(char numArray[], int
numCount)
{
  for (int i = 0; i < numCount; i++)
    for (int j = 1; j < numCount; j++)
      if (numArray[j - 1] < numArray[j])
        CharSwap(&numArray[j - 1],
&numArray[j]);
}
```

## L1-028 判断素数

## L1-029 是不是太胖了

## L1-030 一帮一

```
#include <stdio.h>
#include <math.h>
#include <stdbool.h>

typedef struct
{
  int rank;
  bool gender;
  char name[9];
  bool status;
} STU;

int main(void)
{
  int stuCount;
  scanf("%d", &stuCount);

  STU stuArray[stuCount];
  for (int i = 0; i < stuCount; i++)
  {
    stuArray[i].rank = i + 1;
    stuArray[i].status = false;
    scanf("%d %s", &stuArray[i].gender,
stuArray[i].name);
  }

  int grpCount = stuCount / 2;
  for (int i = 0; i < grpCount; i++)
  {
    if (stuArray[i].status == false)
    {
      for (int j = stuCount - 1; j >= 0; j-
-)
      {
        if (stuArray[j].status == false)
        {
          if (stuArray[i].gender !=
stuArray[j].gender)
          {
            printf("%s %s\n",
stuArray[i].name, stuArray[j].name);
            stuArray[i].status = true;
            stuArray[j].status = true;
            break;
          }
        }
      }
    }
  }
  return 0;
}
```

## L1-031 到底是不是太胖了

## L1-032 Left-pad

## L1-033 出生年

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>
int CntDiffDigit(char str[], int
strLength);
int main(void)
{
  int birthYear, goalVar;
  scanf("%d %d", &birthYear, &goalVar);

  char yearChar[5];

  for (int i = 0; ; i++)
  {
    sprintf(yearChar, "%04d", birthYear +
i);
    if (CntDiffDigit(yearChar, 4) ==
goalVar)
    {
      printf("%d %04d", i, birthYear + i);
      break;

    }
  }

  return 0;
}
int CntDiffDigit(char str[], int
strLength)
{
  bool isRepeat;
  char rptChar[strLength];
  int rptCount = 0;
  for (int i = 0; i < strLength; i++)
  {
    isRepeat = false;
    for (int j = 0; j < rptCount; j++)
    {
      if (str[i] == rptChar[j])
      {
        isRepeat = true;
        break;
      }
    }
    if (!isRepeat)
    {
      rptChar[rptCount] = str[i];
      rptCount++;
    }
  }

  //printf("%d", rptCount);
  return rptCount;
}
```

## L1-034 点赞

```
#include <stdio.h>
#include <stdbool.h>
#define MAX_TAG 1000
#define MAX_BLOG 1000
typedef struct
{
  int tagID;
  int cnt;

} TAG;
int main(void)
{

  int blogCount;
  scanf("%d", &blogCount);

  TAG tagList[MAX_TAG];
  int tagCount = 0;
  bool isExist = false;
  int inputTagCount;
  int inputTag;
  for (int i = 0; i < blogCount; i++)
  {
    scanf("%d", &inputTagCount);
    for (int j = 0; j < inputTagCount; j++)
    {
      scanf("%d", &inputTag);
      //search if exist?
      isExist = false;
      for (int k = 0; k < tagCount; k++)
      {
        if (tagList[k].tagID == inputTag)
        {
          isExist = true;
          tagList[k].cnt++;
        }
      }

      if (isExist == false)
      {
        tagList[tagCount].tagID = inputTag;
        tagList[tagCount].cnt = 1;
        tagCount++;
      }
      else
      {
        continue;
      }
    }
  }

  //find max tag
  int maxCnt = tagList[0].cnt;
  int maxID = tagList[0].tagID;
  for (int i = 0; i < tagCount; i++)
  {
    if (tagList[i].cnt > maxCnt)
    {
      maxCnt = tagList[i].cnt;
      maxID = tagList[i].tagID;
    }
    else if (tagList[i].cnt == maxCnt)
    {
      if (tagList[i].tagID > maxID)
      {
        maxCnt = tagList[i].cnt;
        maxID = tagList[i].tagID;
      }
    }
  }

  //out put

  printf("%d %d", maxID, maxCnt);

  return 0;
}
```

## L1-035 情人节

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#define MAX_LENGTH 15
#define MAX_PEOPLE 100
int main(void)
{
  int peopleCount = 0;


  bool isA = false;
  bool isB = false;
  char nameA[MAX_LENGTH];
  char nameB[MAX_LENGTH];
  char input[MAX_LENGTH];
  while (scanf("%s", input) == 1 &&
strcmp(input, ".") != 0)
  {
    peopleCount++;
    if (peopleCount == 2)
    {
      isA = true;
      strcpy(nameA, input);
    }
    if (peopleCount == 14)
    {
      isB = true;
      strcpy(nameB, input);
    }
  }

  if (isA && isB)
    printf("%s and %s are inviting you to
dinner...", nameA, nameB);
  else if (isA)
    printf("%s is the only one for you...",
nameA);
  else
    printf("Momo... No one is for
you ...");
  return 0;
}
```

## L1-036 A 乘以 B

## L1-037 A 除以 B

## L1-038 新世界

## L1-039 古风排版

```
#include <iostream>
#include <string>
#include <vector>
#include <stack>
using namespace std;
int main(void)
{
  int charIntv;
  cin >> charIntv;
  getchar();
  string input;
  getline(cin, input);
  // fill string
  int fillCnt = input.length() % charIntv ?
charIntv - (input.length() % charIntv) :
0;
  int strLenthCnt = input.length() +
fillCnt;
  for (int i = 0; i < fillCnt; i++)
  {
    input.push_back(' ');
  }
  stack<char> tempStack;
  for (int i = 0; i < charIntv; i++)
  {
    for (int j = i; j < strLenthCnt; j +=
charIntv)
    {
      tempStack.push(input[j]);
    }
    while (!tempStack.empty())
    {
      putchar(tempStack.top());
      tempStack.pop();
    }
    putchar('\n');
  }

  return 0;
}
```

## L1-040 最佳情侣身高差

## L1-041 寻找 250

## L1-042 日期格式化

## L1-043 阅览室

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
int b[1005];
int main()
{
  int n;
  scanf("%d", &n);
  int x, y, z;
  char a[5];
  int sum = 0, s = 0;
  memset(b, -1, sizeof(b));
  while (n--)
  {
    while (1)
    {
      scanf("%d %s %d:%d", &x, a, &y, &z);

      if (x == 0)
      {
        memset(b, -1, sizeof(b));
        if (s != 0)
          printf("%d %0.0f\n", s,
(double)sum / s);
        else
          printf("0 0\n");
        sum = s = 0;
        break;
      }
      else
      {
        if (a[0] == 'S')
        {
          b[x] = y * 60 + z;
        }
        else if (a[0] == 'E' && b[x] != -1)
        {
          s++;
          sum += y * 60 + z - b[x];
```

```
     b[x] = -1;
    }
   }
  }
 }
 return 0;
}
```

## L1-044 稳赢

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
int WhichInput(char *input);
int main(void)
{
 int stepVar;
 scanf("%d", &stepVar);
 stepVar++;

 char *chuZhao[3] = {"JianDao", "ChuiZi",
"Bu"};
 int roundCount = 0;
 int opInput;
 bool isWin;
 char input[10];
 while ((scanf("%s", input)) == 1 &&
strcmp(input, "End") != 0)
 {
  opInput = WhichInput(input);
  isWin = (roundCount + 1) % stepVar ==
0 ? false : true;
  roundCount++;
  if (isWin)
  {
   printf("%s\n", chuZhao[(opInput +
1) % 3]);
  }
  else
  {
   printf("%s\n", chuZhao[opInput]);
  }
 }

 return 0;
}

int WhichInput(char *input)
{
 if (strcmp(input, "JianDao") == 0)
  return 0;
 else if (strcmp(input, "ChuiZi") == 0)
  return 1;
 else if (strcmp(input, "Bu") == 0)
  return 2;
 else
 {
  printf("ERROR\n");
  return -1;
```

```
 }
}
```

## L1-045 宇宙无敌大招呼

## L1-046 整除光棍

```
#include <stdio.h>
int main()
{
  int n,num=1,cnt=1,tag=0;
  scanf("%d",&n);
  while (num%n!=0){
    if(tag)printf("%d",num/n);
    if(num>n){
      if(tag==0){
        printf("%d",num/n);
        tag=1;
      }
      num=(num%n)*10+1;
    }else{
      num=num*10+1;
    }
    cnt++;
  }
  printf("%d %d",num/n,cnt);
  return 0;
}
```

## L1-047 装睡

## L1-048 矩阵 A 乘以 B

```
#include <stdio.h>
int main(void)
{
 int rowA, colA;
 scanf("%d %d", &rowA, &colA);
 int matrix1[rowA][colA];
 for (int i = 0; i < rowA; i++)
  for (int j = 0; j < colA; j++)
   scanf("%d", &matrix1[i][j]);
 int rowB, colB;
 scanf("%d %d", &rowB, &colB);

 if (colA != rowB)
 {
  printf("Error: %d != %d", colA, rowB);
  return 0;
 }
 int matrix2[rowB][colB];
 for (int i = 0; i < rowB; i++)
  for (int j = 0; j < colB; j++)
   scanf("%d", &matrix2[i][j]);
 int matrixResult[rowA][colB];
 for (int i = 0; i < rowA; i++)
  for (int j = 0; j < colB; j++)
  {
   matrixResult[i][j] = 0;
```

```
   for (int k = 0; k < colA; k++)
    matrixResult[i][j] += matrix1[i][k]
* matrix2[k][j];
  }

 printf("%d %d\n", rowA, colB);
 for (int i = 0; i < rowA; i++)
 {
  printf("%d", matrixResult[i][0]);
  for (int j = 1; j < colB; j++)
   printf(" %d", matrixResult[i][j]);
  putchar('\n');
 }
 return 0;
}
```

## L1-049 天梯赛座位分配

```
#include<bits/stdc++.h>
using namespace std;
int a[105];
int b[105][15][15];
int main(){
 int N;
 cin >> N;
 int maxn = -1;
 for(int i=1;i<=N;i++){
  cin >> a[i];
  maxn = max(maxn,a[i]);
 }
 int cnt = 1;
 int count = 0;
 int pre = 0;
 for(int k=1;k<=maxn;k++){
  for(int j=1;j<=10;j++){
   for(int i=1;i<=N;i++){
    if(a[i]>=cnt){
     if(i!=pre){
      b[i][cnt][j] = ++count;
     }else{
      b[i][cnt][j] = count+2;
      count += 2;
     }
     pre = i;
    }
   }
  }
  cnt++;
 }

 for(int i=1;i<=N;i++){
  printf("#%d\n",i);
  for(int j=1;j<=a[i];j++){
   for(int k=1;k<=10;k++){
    if(k==1){
     printf("%d",b[i][j][k]);
    }else{
     printf(" %d",b[i][j][k]);
    }
```

```
   }
   printf("\n");
  }
 }
 return 0;
}
```

## L1-050 倒数第 N 个字符串

```
#include<bits/stdc++.h>
using namespace std;
int a[10];
int b[10];
int main(){
 int L,N;
 cin >> L >> N;
 N -= 1;
 int cnt = 0;
 while(N){
  a[cnt++] = N%26;
  N/=26;
 }
 for(int i=L-1;i>=0;i--){
  b[i] = 25;
 }
 for(int i=0;i<=L-1;i++){
  if(a[i]>b[i]){
   b[i] += 26;
   b[i+1]--;
  }
  b[i] -= a[i];
 }
 for(int i=L-1;i>=0;i--){
  printf("%c",b[i]+'a');
 }
 return 0;
}
```

## L1-051 打折

## L1-052 2018 我们要赢

## L1-053 电子汪

## L1-054 福到了

```
#include <stdio.h>
#include <stdbool.h>
#define FILLCHAR '@'
int size;
bool IsSame(bool *inputA, bool *inputB);
bool IsSameOutput(bool input[][size]);
void ReverseOutput(bool *output, char ch);
int main(void)
{
 char replaceChar;
 scanf("%c %d", &replaceChar, &size);
 while (getchar() != '\n');
```

```
  bool map[size][size];
  char ch;

  for (int i = 0; i < size; i++)
  {
    for (int j = 0; j < size; j++)
    {
      ch =getchar();
      map[i][j] = ch == FILLCHAR ? true :
false;
    }
    while (getchar() != '\n'); //get rid of
useless input
  }
  if (IsSameOutput(map))
    printf("bu yong dao le\n");
  for (int i = size - 1; i >= 0; i--)
  {
    ReverseOutput(map[i], replaceChar);
    putchar('\n');
  }


  return 0;
}
bool IsSame(bool *inputA, bool *inputB)
{
  for (int i = 0; i < size; i++)
    if (inputA[i] != inputB[size - 1 - i])
      return false;
  return true;
}
bool IsSameOutput(bool input[][size])
{
  for (int i = 0; i < size; i++)
    if (!IsSame(input[i], input[size - 1 -
i]))
      return false;
  return true;
}
void ReverseOutput(bool *output, char ch)
{
  for (int i = size - 1; i >= 0; i--)
    if (output[i] == true)
      putchar(ch);
    else
      putchar(' ');
}
```

## L1-055 谁是赢家

```
#include<bits/stdc++.h>
using namespace std;
int hashT[5];
int main(){
  int Pa,Pb;
  cin >> Pa >> Pb;
  for(int i=0;i<3;i++){
    int t;
    cin >> t;
    hashT[t]++;
  }
  if(Pa>Pb && hashT[0]){
    cout << "The winner is a: " << Pa << "
+ " << hashT[0] << endl;
  }else if(Pa<Pb && hashT[1]){
    cout << "The winner is b: " << Pb << "
+ " << hashT[1] << endl;
  }else if(Pa>Pb && hashT[1]==3){
    cout << "The winner is b: " << Pb << "
+ " << hashT[1] << endl;
  }else if(Pa<Pb && hashT[0]==3){
    cout << "The winner is a: " << Pa << "
+ " << hashT[0] << endl;
  }
  return 0;
}
```

## L1-056 猜数字

```
#include<bits/stdc++.h>
using namespace std;
struct student{
  string s;
  int score;
};
student stu[10005];
int sum = 0;
int maxn = 10005;
int main(){
  int N;
  cin >> N;
  for(int i=1;i<=N;i++){
    cin >> stu[i].s >> stu[i].score;
    sum += stu[i].score;
  }
  int ave = sum/N/2;
  int site;
  for(int i=1;i<=N;i++){
    if(abs(ave-stu[i].score)<maxn){
      maxn = abs(ave-stu[i].score);
      site = i;
    }
  }
  cout << ave << " " << stu[site].s <<
endl;
  return 0;
}
```

## L1-057 PTA 使我精神焕发

## L1-058 6 翻了

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  string s;
  getline(cin,s);
  bool f = true;
  while(f){
    f = false;
    int cnt = 0;
    for(int i=0;i<s.size();i++){
      if(s[i] == '6'){
        cnt++;
      }else{
        if(cnt>9){
          s = s.substr(0,i-cnt) + "27" +
s.substr(i);
          f = true;
          cnt = 0;
          break;
        }else if(cnt>3){
          s = s.substr(0,i-cnt) + "9" +
s.substr(i);
          f = true;
          cnt = 0;
          break;
        }
        cnt = 0;
      }
    }
    if(cnt>9){
      s = s.substr(0,s.size()-cnt) + "27";
      f = true;
      cnt = 0;
    }else if(cnt>3){
      s = s.substr(0,s.size()-cnt) + "9";
      f = true;
      cnt = 0;
    }
    cnt = 0;
  }
  cout << s << endl;
  return 0;
}
```

## L1-059 敲笨钟

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  int N;
  cin >> N;
  getchar();
  string s;

  for(int i=1;i<=N;i++){
    getline(cin,s);
    int site = -1;
    bool f1 = false;
    bool f2 = false;
    while(1){
      site = s.find("ong",site+1);
      if(site == -1){
        break;
      }
      if(s[site+3] == ','){
        f1 = true;
      }
      if(s[site+3] == '.'){
        f2 = true;
      }
    }
    if(f1 && f2){
      int cnt = 0;
      int si;
      for(int j=s.size()-1;j>=0;j--){
        if(s[j] == ' '){
          cnt++;
        }
        if(cnt==3){
          si = j;
          break;
        }
      }
      cout << s.substr(0,si) << " qiao ben
zhong." << endl;
    }else{
      cout << "Skipped" << endl;
    }
  }
  return 0;
}
```

## L1-060 心理阴影面积

## L1-061 新胖子公式

## L1-062 幸运彩票

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  int N;
  cin >> N;
  for(int i=1;i<=N;i++){
    int t;
    cin >> t;
    int cnt = 0;
    int sum = 0;
    int sum1;
    while(cnt<6){
      sum += t%10;
      cnt++;
      t/=10;
      if(cnt==3){
        sum1 = sum;
      }
    }
    if(sum == 2*sum1){
      cout << "You are lucky!" << endl;
    }else{
      cout << "Wish you good luck." <<
endl;
```

```
    }
  }

  return 0;
}
```

## L1-063 吃鱼还是吃肉

## L1-064 估值一亿的 AI 核心代码

```
#include<bits/stdc++.h>
using namespace std;
string ans;
int ischar(char ch){
  if(ch == ' '){
    return 1;
  }else if(isdigit(ch)){
    return 2;
  }else if(isalpha(ch)){
    return 3;
  }else{
    return 4;
  }
}
int check(int l,int r){
  if((l<0 || ans[l] == ' ' ||
ischar(ans[l]) == 4) &&
  (r>=ans.size() || ans[r] == ' ' ||
ischar(ans[r]) == 4)){
    return 1;
  }
  return 0;
}
int main(){
  int N;
  cin >> N;
  getchar();
  for(int i=1;i<=N;i++){
    string s;
    getline(cin,s);
    cout << s << endl;
    cout << "AI: ";
    ans = "";
    int l,r;
    for(int i=0;i<s.size();i++){
      if(s[i] == ' '){
        continue;
      }else{
        l = i;
        break;
      }
    }
    for(int i=s.size()-1;i>=0;i--){
      if(s[i] == ' '){
        continue;
      }else{
        r = i;
        break;
      }
    }
    //replace single char
    for(int i=l;i<=r;i++){
      if(s[i] == '?'){
        ans += '!';
      }else if(isupper(s[i]) && s[i]!='I'){
        ans += tolower(s[i]);
      }else if(s[i] == ' ' && (s[i+1] == '
' || ischar(s[i+1])==4)){
        continue;
      }else{
        ans += s[i];
      }
    }
    //replace words
    for(int i=0;i<ans.size();){
      if(ans[i] == 'I' && check(i-1,i+1)){
        cout << "you";
        i++;
      }else if(ans.substr(i,2) == "me" &&
check(i-1,i+2)){
        cout << "you";
        i+=2;
      }else if(ans.substr(i,7) == "can you"
&& check(i-1,i+7)){
        cout << "I can";
        i+=7;
      }else if(ans.substr(i,9) == "could
you" && check(i-1,i+9)){
        cout << "I could";
        i+=9;
      }else{
        cout << ans[i];
        i+=1;
      }
    }
    cout << endl;
  }


  return 0;
}
```

## L1-065 耍废话上代码

## L1-066 猫是液体

## L1-067 洛希极限

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  double s;
  int id;
  double ans;
  cin >> s >> id >> ans;
  if(id==0){
    s*=2.455;
  }else{
    s*=1.26;
  }
  if(s>ans){
    printf("%.2lf T_T\n",s);
  }else{
    printf("%.2lf ^_^\n",s);
  }
  return 0;
}
```

## L1-068 调和平均

## L1-069 胎压监测

```
#include<bits/stdc++.h>
using namespace std;
int a[5];
int low;
int t;
int maxn = -1;
int main(){
  for(int i=1;i<=4;i++){
    cin >> a[i];
  }
  cin >> low >> t;
  for(int i=1;i<=4;i++){
    maxn = max(maxn,a[i]);
  }
  int cnt1 = 0;
  int cnt2 = 0;
  for(int i=1;i<=4;i++){
    if(abs(a[i]-maxn)>t){
      cnt1++;
    }
    if(a[i]<low){
      cnt2++;
    }
  }
  if(!cnt1 && !cnt2){
    cout << "Normal" << endl;
  }else if((cnt1==1 || cnt2==1) &&
cnt1+cnt2<=1){
    if(cnt1==1){
      for(int i=1;i<=4;i++){
        if(abs(a[i]-maxn)>t){
          cout << "Warning: please check #"
<< i << "!" << endl;
          break;
        }
      }
    }else if(cnt2==1){
      for(int i=1;i<=4;i++){
        if(a[i]<low){
          cout << "Warning: please check #"
<< i << "!" << endl;
          break;
        }
      }
    }
  }else{
    cout << "Warning: please check all the
tires!" << endl;
  }
  return 0;
}
```

## L1-070 吃火锅

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  string s;
  int cnt = 0;
  bool f = false;
  int ans1 = 0;
  int ans2 = 0;
  while(1){
    getline(cin,s);
    if(s.size()==1 && s[0]=='.'){
      break;
    }
    cnt++;
    int site = s.find("chi1 huo3 guo1");
    if(site!=-1){
      if(!f){
        f = true;
        ans1 = cnt;
      }
      ans2++;
    }
  }
  cout << cnt << endl;
  if(ans1 && ans2){
    cout << ans1 << " " << ans2 << endl;
  }else{
    cout << "-_-#" << endl;
  }

  return 0;
}
```

## L1-071 前世档案

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  int N,M;
  cin >> N >> M;
  for(int i=1;i<=M;i++){
    string s;
    cin >> s;
    int ans = 0;
    for(int j=0;j<N;j++){
      int t;
      if(s[j] == 'y'){
```

```
      t = 0;
    }else{
      t = 1;
    }
    ans = ans*2+t;
  }
  cout << ans+1 << endl;
 }

 return 0;
}
```

## L1-072 刮刮彩票

```
#include<bits/stdc++.h>
using namespace std;
int a[] = {0,
      0,0,0,0,0,
      10000,36,720,360,80,
      252,108,72,54,180,
      72,180,119,36,306,
      1080,144,1800,3600};
int b[5][5];
int c[5][5];
int hashT[10];
int main(){
  for(int i=1;i<=3;i++){
    for(int j=1;j<=3;j++){
      cin >> b[i][j];
      hashT[b[i][j]] = 1;
    }
  }
  int r;
  for(int i=1;i<=9;i++){
    if(!hashT[i]){
      r = i;
      break;
    }
  }
  for(int i=1;i<=3;i++){
    for(int j=1;j<=3;j++){
      if(!b[i][j]){
        c[i][j] = r;
      }else{
        c[i][j] = b[i][j];
      }
    }
  }

  for(int i=0;i<3;i++){
    int x,y;
    cin >> x >> y;
    b[x][y] = 0;
    cout << c[x][y] << endl;
  }
  int op;
  cin >> op;
  int ans = 0;
  if(op==1){
    for(int i=1;i<=3;i++){
      ans += c[1][i];
    }
  }else if(op==2){
    for(int i=1;i<=3;i++){
      ans += c[2][i];
    }
  }else if(op==3){
    for(int i=1;i<=3;i++){
      ans += c[3][i];
    }
  }else if(op==4){
    for(int i=1;i<=3;i++){
      ans += c[i][1];
    }
  }else if(op==5){
    for(int i=1;i<=3;i++){
      ans += c[i][2];
    }
  }else if(op==6){
    for(int i=1;i<=3;i++){
      ans += c[i][3];
    }
  }else if(op==7){
    for(int i=1;i<=3;i++){
      ans += c[i][i];
    }
  }else if(op==8){
    for(int i=1;i<=3;i++){
      ans += c[3-i+1][i];
    }
  }
  cout << a[ans] << endl;
  return 0;
}
```

## L1-073 人与神

## L1-074 两小时学完 C 语言

## L1-075 强迫症

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  string s;
  cin >> s;
  if(s.size()==4){
    int t = 0;
    t += (s[0]-'0');
    t*=10;
    t += (s[1]-'0');
    if(t<22){
      cout << "20" << s.substr(0,2) << "-"
<< s.substr(2) << endl;
    }else{
      cout << "19" << s.substr(0,2) << "-"
<< s.substr(2) << endl;
    }
  }else{
    cout << s.substr(0,4) << "-" <<
s.substr(4) << endl;
  }

  return 0;
}
```

## L1-076 降价提醒机器人

## L1-077 大笨钟的心情

## L1-078 吉老师的回归

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  int N,M;
  cin >> N >> M;
  getchar();
  bool f = false;
  int cnt = 0;
  for(int i=1;i<=N;i++){
    string s;
    getline(cin,s);
    if(s.find("qiandao")!=-1 ||
s.find("easy")!=-1){
      continue;
    }else{
      cnt++;
    }
    if(cnt>M){
      cout << s << endl;
      f = true;
      break;
    }
  }
  if(!f){
    cout << "Wo AK le" << endl;
  }
  return 0;
}
```

## L1-079 天梯赛的善良

```
#include<bits/stdc++.h>
using namespace std;
int minn = -1;
int maxn = 1e6+5;
int a[100005];
int main(){
  int N;
  cin >> N;
  int cnt1 = 0;
  int cnt2 = 0;
  for(int i=1;i<=N;i++){
    cin >> a[i];
  }
  for(int i=1;i<=N;i++){
    if(a[i]>minn){
      minn = a[i];
      cnt1 = 1;
    }else if(a[i] == minn){
      cnt1++;
    }
    if(a[i]<maxn){
      maxn = a[i];
      cnt2 = 1;
    }else if(a[i] == maxn){
      cnt2++;
    }
  }
  cout << maxn << " " << cnt2 << endl;
  cout << minn << " " << cnt1 << endl;
  return 0;
}
```

## L1-080 乘法口诀数列

```
#include<bits/stdc++.h>
using namespace std;
int a1,a2,n;
int a[10005];
int ans[10005];
int main(){
  cin >> a1 >> a2 >> n;
  a[1] = a1;
  a[2] = a2;
  int j = 3;
  for(int i=3;;i++){
    int t = a[i-2]*a[i-1];
    if(t<10){
      a[j++] = t;
    }else{
      int s = t/10;
      a[j++] = s;
      int g = t%10;
      a[j++] = g;
    }
    if(j>n){
      break;
    }
  }
  for(int i=1;i<=n;i++){
    if(i==1){
      printf("%d",a[i]);
    }else{
      printf(" %d",a[i]);
    }
  }
  return 0;
}
```

## L1-081 今天我要赢

## L1-082 种钻石

## L1-083 谁能进图书馆

## L1-084 拯救外星人

## L1-085 试试手气

```cpp
#include<bits/stdc++.h>
using namespace std;
int a[10];
int b[10];
int main(){
  for(int i=1;i<=6;i++){
    cin >> a[i];
  }
  int n;
  cin >> n;
  for(int i=1;i<=6;i++){
    int t = 7 - n;
    if(t>a[i]){
      b[i] = t;
    }else{
      b[i] = t-1;
    }
  }
  for(int i=1;i<=6;i++){
    if(i==1){
      cout << b[i];
    }else{
      cout << " " << b[i];
    }
  }
  return 0;
}
```

## L1-086 斯德哥尔摩火车上的题

```cpp
#include<bits/stdc++.h>
using namespace std;
string s1 = "";
string s2 = "";
string a1,a2;
int main(){
  cin >> a1 >> a2;
  for(int i=1;i<a1.size();i++) {
      if(a1[i]%2 == a1[i-1]%2) {
        s1 += max(a1[i],a1[i-1]);
      }
  }
  for(int i=1;i<a2.size();i++) {
      if(a2[i]%2 == a2[i-1]%2) {
        s2 += max(a2[i],a2[i-1]);
      }
  }
  if(s1 == s2){
    cout << s1 << endl;
  }else{
    cout << s1 << endl;
    cout << s2 << endl;
  }
  return 0;
}
```

## L1-087 机工士姆斯塔迪奥

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5+5;
bool hashR[maxn];
bool hashC[maxn];
int main(){
  int N,M,Q;
  cin >> N >> M >> Q;
  long long sum = 1LL*N*M;
  int cnt1 = 0;
  int cnt2 = 0;
  for(int i=1;i<=Q;i++){
    int T,C;
    cin >> T >> C;
    if(T==0){
      if(!hashR[C]){
        cnt1++;
        hashR[C] = true;
        sum -= M;
      }
    }else{
      if(!hashC[C]){
        cnt2++;
        hashC[C] = true;
        sum -= N;
      }
    }
  }
  if(cnt1 && cnt2){
    sum += cnt1*cnt2;
  }
  cout << sum << endl;
  return 0;
}
```

## L1-088 静静的推荐

```cpp
#include<bits/stdc++.h>
using namespace std;
struct node{
  int score;
  int PAT;
};
const int maxn = 1e5+5;
node t[maxn];
bool cmp(node t1,node t2){
  if(t1.score!=t2.score){
    return t1.score < t2.score;
  }else{
    return t1.PAT < t2.PAT;
  }
}
int main(){
  int N,K,S;
  cin >> N >> K >> S;
  for(int i=0;i<N;i++){
    scanf("%d%d",&t[i].score,&t[i].PAT);
  }
  sort(t,t+N,cmp);
  int cnt = 0;
  for(int i=0;i<N;){
    if(t[i].score>=175){
      int j1 = i;
      int j2 = i;
      int count1=0,count2=0;
      for(int k=j1;k<N;k++){
        if(t[k].score == t[k+1].score){
          continue;
        }else{
          j2 = k;
          break;
        }
      }
      for(int k=j1;k<=j2;k++){
        if(t[k].PAT>=S){
          count1++;
        }else{
          count2++;
        }
      }
      if(count2<=K){
        cnt += (count1+count2);
      }else{
        cnt += (K + count1);
      }
      i = j2+1;
    }else{
      i++;
    }
  }
  cout << cnt << endl;

  return 0;
}
```

## L2-001 紧急救援

```cpp
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;

const int N = 520;
const int INF = 0x7fffffff;
int G[N][N];   // 存图
int dis[N];    // 源点到各个点的距离（第一标尺)
int cnt[N];    // 每个城市救援队数量
int cntsum[N]; // 源点到每个点的最多队伍数量(第二标尺)
int road[N];   // 最短路径条数
bool vis[N];
int pre[N]; // 存路径
int n, m, s, d;

void printPath(int v)
{
  if (v == s)
  {
    cout << v;
    return;
  }
  printPath(pre[v]);
  cout << " " << v;
}
int main()
{
  scanf("%d%d%d%d", &n, &m, &s, &d);
  fill(G[0], G[0] + N * N, INF);
  fill(dis, dis + N, INF);
  for (int i = 0; i < n; i++)
    cin >> cnt[i];
  int a, b, c;
  for (int i = 0; i < m; i++)
  {
    scanf("%d%d%d", &a, &b, &c);
    G[a][b] = c;
    G[b][a] = c;
  }
  dis[s] = 0;
  road[s] = 1;
  cntsum[s] = cnt[s];
  for (int i = 0; i < n; i++)
  {
    int u = -1, min = INF;
    for (int j = 0; j < n; j++)
    {
      if (vis[j] == false && dis[j] < min)
      {
        min = dis[j];
        u = j;
      }
    }
    if (u == -1)
      break;
    vis[u] = true;
    for (int v = 0; v < n; v++)
    {
      if (G[u][v] != INF && vis[v] == false)
      {
        if (dis[u] + G[u][v] < dis[v])
        {
          dis[v] = dis[u] + G[u][v];
          cntsum[v] = cntsum[u] + cnt[v];
          road[v] = road[u];
```

```
         pre[v] = u;
       }
       else if (dis[u] + G[u][v] ==
dis[v])
       {
         road[v] += road[u];
         if (cntsum[v] < cntsum[u] +
cnt[v])
         {
           cntsum[v] = cntsum[u] + cnt[v];
           pre[v] = u;
         }
       }
     }
    }
  }
  cout << road[d] << " " << cntsum[d] <<
endl;
  printPath(d);
  return 0;
}
```

## L2-002 链表去重

```
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
const int maxn = 1e5;
struct Node{
    int address;
    int key;
    int next;
    int num; //记录数组下标位置
}node[maxn];
bool vis[maxn];
bool cmp(Node a,Node b){
    return a.num<b.num;
}
int main()
{
    int head,n,a;
    scanf("%d%d",&head,&n);
    int k1=0,k2=0;
    for(int i=0;i<maxn;i++){
        node[i].num=2*maxn; //开两倍空间,前
面存储没有重复的,后面是重复的
    }
    for(int i=0;i<n;i++){
        scanf("%d",&a);

scanf("%d%d",&node[a].key,&node[a].next);
        node[a].address=a;
    }
    for(int i=head;i!=-1;i=node[i].next){
        if(!vis[abs(node[i].key)]){
            vis[abs(node[i].key)]=true;
            node[i].num=k1;
            k1++;
        }else{
            node[i].num=maxn+k2;
            k2++;
        }
    }
    sort(node,node+maxn,cmp);
    int k=k1+k2;
    for(int i=0;i<k;i++){
        if(i!=k1-1&&i!=k-1){

printf("%05d %d %05d\n",node[i].address,no
de[i].key,node[i+1].address);
        }else{
            printf("%05d %d -
1\n",node[i].address,node[i].key);
        }
    }
    return 0;
}
```

## L2-003 月饼

```
#include <bits/stdc++.h>
using namespace std;
struct node{
    double w,v,x;
};
bool cmp(node a,node b){
    return a.x>b.x;
}
int main(){
    int n,i;
    double d,sum=0;
    struct node a[1010];
    scanf("%d %lf",&n,&d);
    for(i=1;i<=n;i++){
        scanf("%lf",&a[i].w);
    }
    for(i=1;i<=n;i++){
        scanf("%lf",&a[i].v);
    }
    for(i=1;i<=n;i++){
        a[i].x=(1.0*a[i].v)/a[i].w;
    }
    sort(a+1,a+n+1,cmp);
    for(i=1;i<=n;i++){
        if(d>=a[i].w){
            sum+=a[i].v;
            d-=a[i].w;
        }
        else{
            sum+=1.0*d*a[i].x;
            break;
        }
    }
    printf("%.2f\n",sum);
    return 0;
}
```

## L2-004 这是二叉搜索树吗?

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
typedef long long ll;
using namespace std;
const int maxn = 1e3 + 10;
int n,a[maxn];
vector<int> now;
void f(int l,int r,int x)
{
    if(l > r) return;
    int tl = r;
    int tr = l + 1;
    if(!x)
    {
        while(tl > l && a[tl] >= a[l]) tl-
-;

        while(tr <= r && a[tr] < a[l])
tr++;
    }
    else
    {
        while(tl > l && a[tl] < a[l]) tl--;
        while(tr <= r && a[tr] >= a[l])
tr++;
    }
    if(tr - tl != 1) return;
    f(l+1,tl,x);
    f(tr,r,x);
    now.push_back(a[l]);
}
int main()
{
    scanf("%d",&n);
    for(int i = 0;i < n;i++)
scanf("%d",&a[i]);
    f(0,n - 1,0);
    if(now.size() != n)
    {
        now.clear();
        f(0,n - 1,1);
    }
    if(now.size() != n) printf("NO");
    else
    {
        printf("YES\n%d",now[0]);
        for(int i = 1;i < n;i++)
printf(" %d",now[i]);
    }
    return 0;
}
```

## L2-005 集合相似度

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <string.h>
#include <vector>
#include <set>
using namespace std;
set<int>q[55];
int main(){
    int n;scanf("%d",&n);
    for(int i = 1;i <= n;i ++){
        int k;scanf("%d",&k);
        for(int j = 1;j <= k;j ++){
            int s;scanf("%d",&s);
            q[i].insert(s);
        }
    }
    int m;scanf("%d",&m);
    while(m --){
        int a,b;
        scanf("%d%d",&a,&b);
        float ans1 = 0,ans2 = 0;
        for(auto it : q[a]){
            if(q[b].find(it) != q[b].end())
                ans1 ++;
        }
        ans2 = q[a].size() + q[b].size() -
ans1;
        printf("%.2lf%%\n",ans1 * 100 /
ans2);
    }
    return 0;
}
```

## L2-006 树的遍历

```
#include<iostream>
#include<map>
#include<queue>
using namespace std;
map<int,int>l,r;
int hou[1010];//后序遍历数组
int in[1010];// 中序遍历数组
int n;
queue<int>q;
int build(int hl,int hr,int il,int ir)
{
  if(hl>hr||il>ir) return 0;
  int root=il;
  //找到后序遍历根节点在中序遍历中的位置
  while(root<=ir&&in[root]!=hou[hr])
root++;
  int cnt=root-il;
  l[hou[hr]]=build(hl,hl+cnt-1,il,root-1);
  r[hou[hr]]=build(hl+cnt,hr-1,root+1,ir);
  return hou[hr];
}
```

```
void print(int root)//层序遍历输出
{
  q.push(root);
  cout<<root;

  int sum=1;
  if(sum!=n) cout<<" ";
  while(q.size())
  {
    root=q.front();
    q.pop();
    if(l[root])
    {
        cout<<l[root];
      q.push(l[root]);
      sum++;
      if(sum!=n) cout<<" ";
    }

    if(r[root])
    {
      cout<<r[root];
        sum++;
        q.push(r[root]);
        if(sum!=n) cout<<" ";
    }
    if(sum==n) break;

  }
}
int main()
{

  cin>>n;
  for(int i=1;i<=n;i++)
  {
    cin>>hou[i];
  }
  for(int i=1;i<=n;i++)
  {
    cin>>in[i];
  }
  int root=build(1,n,1,n);
  print(root);

  return 0;
}
```

## L2-007 家庭房产

```
#include <bits/stdc++.h>
#define IOS
std::ios::sync_with_stdio(false);std::cin.
tie(0);
using namespace std;
const int N =1e4 + 5;
int cnt = 0, k;
struct data{
    int id,fa,mom,num,area;
    int child[10];
}a[N];
struct node{
    int id,people;
    double num,area;
    bool flag = false;
}b[N];
bool vis[N];
int pre[N];
bool cmp(node a,node b){
    if(a.area != b.area)
        return a.area > b.area;
    return a.id < b.id;
}
int find(int x){
    if(x != pre[x]) pre[x] = find(pre[x]);
    return pre[x];
}
void merge(int x,int y){
    int fx = find(x);
    int fy = find(y);
    if(fx < fy) pre[fy] = fx;
    else pre[fx] = fy;
}
int main(){
  IOS;
  int n;cin >> n;
    for(int i = 0;i < N;i ++) pre[i] = i;
    for(int i = 0;i < n;i ++){
        cin >> a[i].id >> a[i].fa >>
a[i].mom >> k;
        vis[a[i].id] = 1;
        if(a[i].fa != -1){
            merge(a[i].id,a[i].fa);
            vis[a[i].fa] = 1;
        }
        if(a[i].mom != -1){
            merge(a[i].id,a[i].mom);
            vis[a[i].mom] = 1;
        }
        for(int j = 0;j < k;j ++){
            cin >> a[i].child[j];
            vis[a[i].child[j]] = 1;
            merge(a[i].child[j],a[i].id);
        }
        cin >> a[i].num >> a[i].area;
    }
    for(int i = 0;i < n;i ++){
        int id = find(a[i].id);
        b[id].id = id;
        b[id].num += a[i].num;
        b[id].area += a[i].area;
        b[id].flag = true;
    }
    for(int i = 0;i < N;i ++){
        if(vis[i])  b[find(i)].people ++;
        if(b[i].flag) cnt++;
    }
    for(int i = 0;i < N;i ++){
        if(b[i].flag){
            b[i].num =1.0 * b[i].num /
b[i].people ;
            b[i].area = 1.0 * b[i].area /
b[i].people;
        }
    }
    sort(b,b+N,cmp);
    cout << cnt << endl;
  for(int i=0; i<cnt; i++)

  printf("%04d %d %.3f %.3f\n",b[i].id,b[i]
.people,b[i].num,b[i].area);
  return 0;
}
```

## L2-008 最长对称子串

```
#include<bits/stdc++.h>
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
  string s;
  getline(cin,s);//这个是整行读入 cin 是读入
到第一个空格处
  int x=1;
  for(int i=0; i<s.length(); i++) {
    for(int j=s.length()-1; j>=i; j--) {
      int left=i,right=j;

  while(left<=right&&s[left++]==s[right--])
{
        if(left>right)
          x=max(x,j-i+1);
      }
    }
  }
  cout<<x<<endl;
  return 0;
}
```

## L2-009 抢红包

```
#include<cstdio>
#include<vector>
#include<algorithm>
using namespace std;
struct X{
    int id,sum,num;
};
bool cmp(X a,X b){
    if(a.sum!=b.sum){
        return a.sum>b.sum;
    }else if(a.num!=b.num){
        return a.num>b.num;
    }else{
        return a.id<b.id;
    }
}
int main(){
    int n,k,a,b;
    scanf("%d",&n);
    vector<X> v(n+1);
    for(int i=1;i<=n;i++){
        v[i].id = i;
        scanf("%d",&k);
        for(int j=0;j<k;j++){
            scanf("%d%d",&a,&b);
            v[a].sum+=b;
            v[a].num++;
            v[i].sum-=b;
        }
    }
    sort(v.begin()+1,v.end(),cmp);
    for(int i=1;i<=n;i++){
        double res =
(double)(v[i].sum*1.0/100);
        printf("%d %.2f\n",v[i].id,res);
    }
    return 0;
}
```

## L2-010 排座位

```
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;

int pre[110];
int enemy[110][110];
int find(int x)
{
  if (x != pre[x])
    return pre[x] = find(pre[x]);
  return pre[x];
}
void merge(int a, int b)
{
  int fx = find(a);
  int fy = find(b);
  if (fx != fy)
  {
    pre[fx] = fy;
  }
}
int main()
{
  int n, m, k, a, b, c;
  scanf("%d %d %d", &n, &m, &k);
  for (int i = 1; i <= n; i++)
  pre[i] = i;
 for (int i = 0; i < m; i++)
 {
  scanf("%d %d %d", &a, &b, &c);
  if (c == 1)
  {
   merge(a, b);
  }
  else
  {
   enemy[a][b] = 1;
   enemy[b][a] = 1;
  }
 }
 for (int i = 0; i < k; i++)
 {
  scanf("%d %d", &a, &b);
  if (find(a) == find(b) && enemy[a][b]
== 0)
  {
   printf("No problem\n");
  }
  else if (find(a) != find(b) &&
enemy[a][b] == 0)
  {
   printf("OK\n");
  }
  else if (find(a) == find(b) &&
enemy[a][b] == 1)
  {
   printf("OK but...\n");
  }
  else if (enemy[a][b] == 1)
  {
   printf("No way\n");
  }
 }
 return 0;
}
```

## L2-011 玩转二叉树

```
#include<bits/stdc++.h>
using namespace std;
int
mid[1000],in[1000],Left[1000],Right[1000];
int n;
int build(int L1,int R1,int L2,int R2)
{
  if(L1>R1)return 0;
  int root=in[L2];
  int pos=L1;
  while(mid[pos]!=root)pos++;
  int cnt=pos-L1;
  Left[root]=build(L1,pos-1,L2+1,L2+cnt);

Right[root]=build(pos+1,R1,L2+cnt+1,R2);
  return root;
}
void level()
{
  queue<int>q;
  q.push(in[1]);
  int f=0;
  while(!q.empty())
  {
    int u=q.front();q.pop();
    if(!f)printf("%d",u),f=1;
    else printf(" %d",u);
    if(Right[u])q.push(Right[u]);
    if(Left[u])q.push(Left[u]);
  }
}
int main()
{
  cin>>n;
  for(int i=1;i<=n;i++)cin>>mid[i];
  for(int i=1;i<=n;i++)cin>>in[i];
  int root=build(1,n,1,n);
  level();
  return 0;
}
```

## L2-012 关于堆的判断

```
#include <bits/stdc++.h>
#define Inf 0x3f3f3f
const int N = 1005;
using namespace std;
int n, m, cnt, no;
int ans[N];
map<int, int> p;

void create(int x)
{ // 堆的建立
 ans[++cnt] = x;
 int t = cnt;
 while (t > 1 && ans[t / 2] > ans[t])
 {
  swap(ans[t / 2], ans[t]);
  t /= 2;
 }
 ans[t] = x;
}
int main()
{
 cin >> n >> m;
 for (int i = 1; i <= n; i++)
 {
  cin >> no;
  create(no);
 }
 for (int i = 1; i <= n; i++) // 记录堆下标
  p[ans[i]] = i;
 while (m--)
 {
  string str;
  int x, y;
  cin >> x >> str;
  if (str[0] == 'a')
  { // 兄弟节点判断 x and y are siblings
   cin >> y >> str >> str;
   if (p[x] / 2 == p[y] / 2)
    cout << "T" << endl;
   else
    cout << "F" << endl;
  }
  else
  {
   cin >> str >> str;
   if (str[0] == 'c')
   { // 子节点判断:  x is a child of y
    cin >> str >> y;
    if (p[x] / 2 == p[y])
     cout << "T" << endl;
    else
     cout << "F" << endl;
   }
   else if (str[0] == 'r')
   { // 根节点判断:  x is the root
    if (p[x] == 1)
     cout << "T" << endl;
    else
     cout << "F" << endl;
   }
   else if (str[0] == 'p')
   { // 父节点判断:  x is the parent of y
    cin >> str >> y;
    if (p[x] == p[y] / 2)
     cout << "T" << endl;
    else
     cout << "F" << endl;
   }
  }
 }

 return 0;
}
```

## L2-013 红色警报

```
#include <iostream>
#include <algorithm>
#include <cstdio>
using namespace std;
const int N = 5200;
bool vis[N];
int pre[N];
struct node {
  int x,y;
}q[N];
int find(int x){
  if(pre[x] != x)
    return pre[x] = find(pre[x]);
  return pre[x];
}
void merge(int x,int y){
  int fx = find(x);
  int fy = find(y);
  if(fx != fy)
    pre[fx] = fy;
}
int n,m;
int main(){
 scanf("%d%d", &n, &m);
 for(int i = 0;i < n;i ++) pre[i] = i;
  for(int i = 0;i < m;i ++){
    cin >> q[i].x >> q[i].y;
    merge(q[i].x,q[i].y);
  }
  for(int i=0;i<n;i++)  find(i);
  int cnt1 = 0;
  for(int i = 0;i < n;i ++){
    if(find(i) == i)
      cnt1 ++;
  }
  int k;scanf("%d",&k);
  while(k --){
   for(int i = 0;i < n;i ++) pre[i] = i;
    int c;scanf("%d",&c);
    vis[c] = true;
    for(int i = 0;i < m;i ++){
      if(!vis[q[i].x]
&& !vis[q[i].y]){
        merge(q[i].x,q[i].y);
      }
    }
    for(int i=0;i<n;i++)   find(i);
    int cnt2 = 0;
    for(int i = 0;i < n;i ++){
      if(find(i) == i&&!vis[i])
        cnt2 ++;
    }
    if(cnt1 == cnt2 || cnt1 == cnt2 +
1)
      printf("City %d is lost.\n",c);
    else
      printf("Red Alert: City %d is
lost!\n",c);
    cnt1 = cnt2;
  }
  int ans = 0;
  for(int i = 0;i < n;i ++)
    if(vis[i])
      ans ++;
  if(ans == n)
    printf("Game Over.\n");
  return 0;
}
```

## L2-014 列车调度

```
#include <iostream>
#include <set>
using namespace std;
```

```
int main() {
    set<int> se;
    int n;
    cin >> n;
    for (int i = 0; i < n; ++ i) {
        int x;
        cin >> x;
        auto it = se.lower_bound(x);
        if (it != se.end()) {
            se.erase(it);
        }
        se.insert(x);
    }
    cout << se.size();
}
```

## L2-015 互评成绩

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n,k,m;
    double ch[10005][15];
    double sum[10005]={0};
    double ave[10005];
    cin>>n>>k>>m;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<k;j++)
        {
            cin>>ch[i][j];
            sum[i]+=ch[i][j];
        }
        sort(ch[i],ch[i]+k);
        sum[i]=sum[i]-ch[i][0]-ch[i][k-1];
        ave[i]=sum[i]/(double)(k-2);
    }
    sort(ave,ave+n);
    for(int t=n-m;t<n;t++)
    {
        if(t==n-m) printf("%.3lf",ave[t]);
        else printf(" %.3lf",ave[t]);
    }
    return 0;
}
```

## L2-016 愿天下有情人都是失散多年的兄妹

```
#include<iostream>
#include<algorithm>
using namespace std;
const int maxn=100010;

bool flag=1;
int vis[maxn]={0};

struct node{
  char gender;
  int father,mother;
    node(){
    father=-1;
    mother=-1;
  }
}node[maxn];
void judge(int idx,int k){
  if(idx==-1||k==6){
    return ;
  }
  vis[idx]++;
  if(vis[idx]==2)
    flag=0;
  judge(node[idx].father,k+1);
  judge(node[idx].mother,k+1);
}
int main(){
  int n,m;
  cin>>n;
  for(int i=0;i<n;i++){
    char gender;
    int id,father,mother;
    cin>>id>>gender>>father>>mother;
    node[id].gender=gender;
    node[id].father=father;
    node[id].mother=mother;
    if(father!=-1)node[father].gender='M';
    if(mother!=-1)node[mother].gender='F';
  }
  cin>>m;
  for(int i=0;i<m;i++){
    int x,y;
    cin>>x>>y;
    if(node[x].gender==node[y].gender)
     cout<<"Never Mind"<<endl;
    else{
      flag=1;
      fill(vis,vis+maxn,0);
      judge(x,1);
      judge(y,1);
      if(flag) cout<<"Yes"<<endl;
      else cout<<"No"<<endl;
    }
  }
  return 0;
}
```

## L2-017 人以群分

```
#include<bits/stdc++.h>
using namespace std;
//#define int long long
#define fo(i,a,b) for(int i=a;i<b;i++)
#define lop(i,a,b) for(int i=a;i<=b;i++)
#define MX 100007
int a[MX];
int N;
int main(){
  cin>>N;
  fo(i,0,N){
    cin>>a[i];
  }
  sort(a,a+N);
  int half=N/2;
  int s1=0,s2=0;
  fo(i,0,half) s1+=a[i];
  fo(i,half,N) s2+=a[i];
  printf("Outgoing #: %d\nIntroverted
#: %d\nDiff = %d\n",N-half,half,s2-s1);
  return 0;
}
```

## L2-018 多项式 A 除以 B

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
int b[N];
double a[N], c[N], d[N];
int main()
{
  int n, m, f1, n1 = 0, n2 = 0;
  scanf("%d", &n);
  for (int i = 0, x; i < n; i++)
  {
   scanf("%d", &x);
   scanf("%lf", &a[x]);
   if (!i)
     f1 = x;
  }
  scanf("%d", &m);
  for (int i = 0; i < m; i++)
   scanf("%d%lf", &b[i], &d[i]);
  for (int i = f1, t; i >= b[0]; i--)
  {
   t = i - b[0];
   c[t] = a[i] / d[0];
   for (int j = 0; m > j; j++)
     a[t + b[j]] = a[t + b[j]] - c[t] *
d[j];
  }
  for (int i = f1; i >= 0; i--)
  {
   if (fabs(a[i]) >= 0.05)
     n2++;
   if (fabs(c[i]) >= 0.05)
     n1++;
  }
  printf("%d", n1);
  for (int i = f1; i >= 0; i--)
   if (fabs(c[i]) >= 0.05)
     printf(" %d %.1f", i, c[i]);
  if (n1 == 0)
   printf(" 0 0.0");
  printf("\n");
  printf("%d", n2);
  for (int i = f1; i >= 0; i--)
    if (fabs(a[i]) >= 0.05)
     printf(" %d %.1f", i, a[i]);
  if (n2 == 0)
    printf(" 0 0.0");
  return 0;
}
```

## L2-019 悄悄关注

```
#include <iostream>
#include <unordered_set>
#include <map>
using namespace std;
int main() {
    int n;
    cin >> n;
    unordered_set<string> se;
    for (int i = 0; i < n; ++ i) {
        string x;
        cin >> x;
        se.insert(x);
    }
    map<string, int> ma;
    int m;
    cin >> m;
    int sum = 0;
    for (int i = 0; i < m; ++ i) {
        string x;
        int y;
        cin >> x >> y;
        ma[x] = y;
        sum += y;
    }
    double avg = sum * 1.0 / m;
    bool ok = false;
    for (auto x : ma) {
        if (x.second > avg &&
se.find(x.first) == se.end()) {
            ok = true;
            cout << x.first<< endl;
        }
    }
    if (!ok) {
        cout << "Bing Mei You";
    }
}
```

## L2-020 功夫传人

```
#include<iostream>
#include<vector>
using namespace std;
int n;
double r,result = 0;
struct people {
  int king = 1;
  vector<int> child;
}person[100000];
void search(int i, double k) {
  if (person[i].king != 1)
    result += k *= person[i].king;
  for (auto& it : person[i].child)
    search(it,k*(100 - r) / 100);
}
int main() {
  int k, t;
  double z;
  scanf("%d%lf%lf", &n, &z, &r);
  for (int i = 0; i < n; i++) {
    scanf("%d", &k);
    if (!k)
      scanf("%d", &person[i].king);
    for (int j = 0; j < k; j++) {
      scanf("%d", &t);
      person[i].child.push_back(t);
    }
  }
  search(0, z);
  printf("%d", (int)result);
  return 0;
}
```

## L2-021 点赞狂魔

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e3+10;
struct node{//存 名字 不同标签的数量 标签出现
次数平均值
  char name[10];
  int num,k;
    const bool operator < (const node &t)
const {
    if(t.num==num) return t.k>k;
    return t.num<num;
    }
}peo[N];
int main(){
  int n; scanf("%d",&n);
  for (int i = 0 ; i < n ; i++ ) {
    scanf("%s",peo[i].name);
    int k; scanf("%d",&k);
    peo[i].k = k;
    set<int> s;
    while ( k-- ){
      int x; scanf("%d",&x);
      s.insert(x);
    }
    peo[i].num = s.size();
  }
  sort(peo,peo+n);
  if(n>0)  printf("%s ", peo[0].name);
  else printf("- ");
  if(n>1) printf("%s ", peo[1].name);
  else printf("- ");
  if(n>2) printf("%s\n", peo[2].name);
  else printf("-\n");
  return 0;
}
```

## L2-022 重排链表

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>

const int INF = 0x3f3f3f3f;
const int N = 1e6+10;
struct node{//存每个点的 地址 值 下一点地址
  int add, data, next;
}Lnode[N];
int main(){
  int add_s,n; scanf("%d %d", &add_s, &n);
  int sum=1;
  for( int i = 0 ; i < n ; i++ ){
    int a,b,c;
    scanf("%d %d %d", &a, &b, &c);
    Lnode[a].add=a;
    Lnode[a].data=b;
    Lnode[a].next=c;
  }
  vector<node> Array;     //存链表上的所有点
  do{
    Array.push_back(Lnode[add_s]);
    add_s = Lnode[add_s].next;
  }while(add_s!=-1);

  int index = 0, length = Array.size() - 1;
  printf("%05d %d ", Array[length].add,
Array[length].data);
  for ( int i = 0 ; i < length ; i++ ){
    int pos; //pos 指向当前要输出的点
    if( i%2 == 0 ){
      pos = index;
      index ++;
    }
    else
      pos = length-index;
    printf("%05d\n", Array[pos].add);
    printf("%05d %d ", Array[pos].add,
Array[pos].data);
  }
  printf("-1\n");
  return 0;
}
```

## L2-023 图着色问题

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e3+10;
vector<PII> vec;    //邻接表存边
int col[N];     //存每个点的颜射
int main(){
  int v,e,k;
  scanf("%d%d%d", &v, &e, &k);
  for ( int i = 0 ; i < e ; i++ ){
    int x, y; scanf("%d%d", &x, &y);
    vec.push_back({x,y});
  }
  int q; scanf("%d", &q);
  while ( q-- ){
    set<int> s; //记录出现过的颜色
    for ( int i = 1 ; i <= v ; i++ ){
      scanf("%d", &col[i]);
      s.insert(col[i]);
    }

    int flag = ( s.size() == k );   //统计
用过的颜料
    for ( int i = 0 ; i < vec.size() ;
i++ )
      if(col[vec[i].first] ==
col[vec[i].second])
        flag = false;

    if ( flag ) cout<<"Yes\n";
    else cout<<"No\n";
  }
  return 0;
}
```

## L2-024 部落

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e4+10;
int fa[N];
void init(int n) { for ( int i = 1 ; i <=
n ; i++ ) fa[i] = i; }   //初始化
int find(int x) { return fa[x] == x ? x :
fa[x] = find( fa[x] ); } //查找 路径压缩
void merge(int a, int b) { a = find(a), b
= find(b), fa[b] = a; }  //合并
int main(){
  int n; scanf("%d", &n);
  int m=0; //记录编号最大的人 即 总人数
  init(N);
  while ( n-- ) {
    int k, x, y ; scanf("%d%d", &k, &x);
    m = max(m, x);
    for ( int i = 1 ; i < k ; i++ ) {
      scanf("%d", &y);
      merge(x, y);   //合并
      m = max(m, y);
    }
  }
  int ans = 0;//计算部落数
  for ( int i = 1 ; i <= m ; i++ )
    if ( fa[i] == i ) ans++;
  printf("%d %d\n", m, ans);

  scanf("%d", &n);
  while ( n-- ){
    int x, y; scanf("%d %d",&x, &y);
    if ( find(x) == find(y) ) cout<<"Y\n";
    else cout<<"N\n";
  }
  return 0;
}
```

## L2-025 分而治之

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e4+10;
int fa[N];
vector<int> v[N];
void init(int n) { for ( int i = 1 ; i <=
n ; i++ ) fa[i] = i; }   //初始化
int find(int x) { return fa[x] == x ? x :
fa[x] = find( fa[x] ); } //查找 路径压缩
void merge(int a, int b) { a = find(a), b
= find(b), fa[b] = a; }  //合并
int main(){
  int n, m; scanf("%d%d", &n, &m);
  while ( m-- ) {
    int x, y; scanf("%d%d",&x, &y);
    v[x].push_back(y);
  }
  scanf("%d", &m);
  while( m-- ){
    set<int> s; //用set 存被摧毁的城市 方便查
找
    int k, num; scanf("%d", &k);
    num = k;
    while ( k-- ) {
      int x; scanf("%d",&x);
      s.insert(x);
    }
    init(n);
    for ( int i = 1 ; i <= n ; i++ ){
      for ( int j = 0 ; j < v[i].size() ;
j++ ){
        if(s.count(i)==0 &&
s.count(v[i][j])==0) //路径两端都没被摧毁
          merge(i, v[i][j]);
      }
    }
    int cnt = 0;
    for ( int i = 1 ; i <= n ; i++ )
    if ( fa[i] == i && s.count(i) == 0)
cnt++;
```

```
    if ( cnt >= n-num ) printf("YES\n");
    else printf("NO\n");
  }
  return 0;
}
```

## L2-026 小字辈

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e5+10;
int fa[N],bf[N];
int fun(int x){ //递归计算辈分
  if ( bf[x] ) return bf[x];   //避免重复计
算
  if ( x == -1 ) return 0;
  return bf[x] = 1+fun(fa[x]);
}
int main(){
  int n; scanf("%d", &n);
  for ( int i = 1 ; i <= n ; i ++ ){
    int x; scanf("%d",&x);
    fa[i] = x;
  }
  for ( int i = 1 ; i <= n ; i ++ ) //计算
成员辈分
    fun(i);
  int ans = 0;
  for ( int i = 1 ; i <= n ; i ++ ) //寻找
最小辈分
    ans = max(ans, bf[i]);
  printf("%d\n", ans);
  vector<int> v;
  for ( int i = 1 ; i <= n ; i ++ ) //存答
案
    if ( bf[i] == ans ) v.push_back(i);
  for ( int i = 0 ; i < v.size() ; i ++ )
    printf("%d%c", v[i] , i == v.size()-1 ?
'\n' : ' ');
  return 0;
}
```

## L2-027 名人堂与代金券

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e4+10;
struct node{
  string name;
  int num;
  const bool operator<(const node &t) {
        if (num == t.num) return name <
t.name;
        return num > t.num;
    }
}stu[N];
int main(){
  int n, g, k;
  scanf("%d%d%d", &n, &g, &k);
  int sum = 0;
  for ( int i = 1 ; i <= n ; i ++ ){
    cin >> stu[i].name >> stu[i].num;
    if ( stu[i].num >= g) sum += 50;
    else if ( stu[i].num >= 60 ) sum += 20;

  }
  printf("%d\n",sum);
  sort(stu+1,stu+n+1);
  int cnt = 1;
  for ( int i = 1 ; i <= n ; i ++ ){
    if ( stu[i].num < stu[i-1].num ) cnt=i;
    if ( cnt > k ) break;
    cout << cnt << " " << stu[i].name << "
" << stu[i].num << endl;
  }
  return 0;
}
```

## L2-028 秀恩爱分得快

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e3 + 3;
typedef long long ll;
int n, m, a, b, t, x, y;
string s;     // 字符串读入,用于判断"-0".用
int 无法判断-0 的情况
vector<int> v[N]; // 记录照片信息
double sum[N][N]; // 亲密度总和
int sex[N];     // 1->男 -1->女
void O(int x)
{ // 输出,因为输出"-0" 需要特判
  if (x == 0 && sex[0] == -1)
    cout << '-';
  cout << sex[abs(x)] * abs(x); // 这样子输
入普通 i,也能输出正确的性别
}
int work(int i, int a)
{ // 判断有无该情侣,有的话计算亲密度总和
  auto q = lower_bound(v[i].begin(),
v[i].end(), a);
  int tmp = 0;
  if (*q == a)
  { // 在照片中有 a 这个人
    x = sex[abs(a)];
    for (int j = 0; j < v[i].size(); j++)
    {
      y = sex[abs(v[i][j])];
      if (x * y < 0)
      { // 只有异性才计算亲密度
        sum[abs(a)][abs(v[i][j])] += 1.0 /
(v[i].size() * 1.0);
        sum[abs(v[i][j])][abs(a)] += 1.0 /
(v[i].size() * 1.0);
      }
    }
    return 1;
  }
  return 0;
}
int main()
{
  cin >> n >> m;
  for (int i = 1; i <= m; i++)
  {
    cin >> a;
    for (int j = 1; j <= a; j++)
    {
      cin >> s;
      t = stoi(s); // string 转 int
      v[i].push_back(t);
      sex[abs(t)] = s[0] == '-' ? -1 : 1;
// 判断男女
    }
    sort(v[i].begin(), v[i].end());
  } // 坑点,性别信息可能不在照片里,在给出的情
侣里
  cin >> s;
  a = stoi(s);
  sex[abs(a)] = s[0] == '-' ? -1 : 1;
  cin >> s;
  b = stoi(s);
  sex[abs(b)] = s[0] == '-' ? -1 : 1;
  for (int i = 1; i <= m; i++)
  {
    int tmp = 0;
    tmp += work(i, a);
    tmp += work(i, b);
    if (tmp == 2)
    { // 去掉重复相加的
      sum[abs(a)][abs(b)] -= 1.0 /
(v[i].size() * 1.0);
      sum[abs(b)][abs(a)] -= 1.0 /
(v[i].size() * 1.0);
    }
  }
  double Max1 = 0, Max2 = 0;
  for (int i = 0; i < n; i++)
    Max1 = max(Max1, sum[abs(a)][i]);
  for (int i = 0; i < n; i++)
    Max2 = max(Max2, sum[abs(b)][i]);
  if (Max1 == Max2 && sum[abs(a)][abs(b)]
== Max1)
    O(a), cout << " ", O(b), cout << endl;
  else
  {
    for (int i = 0; i < n; i++)
      if (sex[abs(a)] * sex[i] < 0 &&
sum[abs(a)][i] == Max1)
        O(a), cout << " ", O(i), cout <<
endl;
    for (int i = 0; i < n; i++)
      if (sex[abs(b)] * sex[i] < 0 &&
sum[abs(b)][i] == Max2)
        O(b), cout << " ", O(i), cout <<
endl;
  }
}
```

## L2-029 特立独行的幸福

```
#include<bits/stdc++.h>
using namespace std;
int is_prime(int n){
  if(n<2) return 1;
  for(int i=2;i<=sqrt(n);i++)
    if(n%i==0) return 1;
  return 2;
}
int main(){
  int left,right,appear[100001]={0};
  cin>>left>>right;
  map<int,int> result;
  for(int i=left;i<=right;i++){
    int n=i,sum=0;
    vector<int> v;
    while(n!=1){
      sum=0;
      while(n){
        sum+=(n%10)*(n%10);
        n/=10;
      }
      n=sum;

    if(find(v.begin(),v.end(),sum)!=v.end())
        break; //判断重复
      v.push_back(n);
      appear[n]=1;
    }
    if(n==1) result[i]=v.size();
  }
  map<int,int>::iterator it;
  int flag=0;
  for(it=result.begin();it!=result.end();it
++){
    if(!appear[it->first]){

  printf("%d %d\n",it->first,it->second*is_
prime(it->first));
      flag=1;
    }
  }
  if(flag==0) printf("SAD");
  return 0;
}
```

## L2-030 冰岛人

```
#include <iostream>
#include <unordered_map>
```

```cpp
using namespace std;

struct node
{
  string fa;
  int sex;
};
unordered_map<string, node> ans;

int check(string a, string b) // 判断五代以内有无公共祖先
{
  int i = 1;
  for (string A = a; !A.empty(); i++, A = ans[A].fa)
  {
    int j = 1;
    for (string B = b; !B.empty(); j++, B = ans[B].fa)
    {
      if (i >= 5 && j >= 5)
        return 1;
      if (A == B && (i < 5 || j < 5))
        return 0;
    }
  }
  return 1;
}
int main()
{
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    string a, b;
    cin >> a >> b;
    if (b[b.size() - 1] == 'm')
      ans[a].sex = 1;
    else if (b[b.size() - 1] == 'f')
      ans[a].sex = 2;
    else if (b[b.size() - 1] == 'n')
    {
      string ss = b.substr(0, b.size() - 4);
      ans[a].sex = 1;
      ans[a].fa = ss;
    }
    else if (b[b.size() - 1] == 'r')
    {
      string ss = b.substr(0, b.size() - 7);
      ans[a].sex = 2;
      ans[a].fa = ss;
    }
  }
  int k;
  cin >> k;
  while (k--)
  {
    string a, b, c, d;
    cin >> a >> b >> c >> d;
    if (!ans[a].sex || !ans[c].sex)
      cout << "NA\n";
    else if (ans[a].sex == ans[c].sex)
      cout << "Whatever\n";
    else if (check(a, c))
      cout << "Yes\n";
    else
      cout << "No\n";
  }
  return 0;
}
```

## L2-031 深入虎穴

```cpp
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e5+10;
int  fa[N],dis[N];
int dfs(int x, int id){
  if ( x == id ) return 0;
  if ( dis[x] != 0 ) return dis[x];
  return dis[x] = 1 + dfs(fa[x], id);
}
int main(){
  int n; scanf("%d", &n);
  for ( int i = 1 ; i <= n ; i ++ ) fa[i] = i;
  for ( int i = 1 ; i <= n ; i ++ ){
    int k; scanf("%d", &k);
    while( k -- ){
      int x; scanf("%d", &x);
      fa[x] = i;
    }
  }
  int id;
  for ( int i = 1 ; i <= n ; i ++ )
    if ( fa[i] == i ){
      id = i;
      break;
    }

  for ( int i = 1 ; i <= n ; i ++ ) dfs(i, id);
  int maxdis = -1, maxid = 0;
  for ( int i = 1 ; i <= n ; i ++ )
    if ( maxdis < dis[i])
      maxdis = dis[i], maxid = i;
  printf("%d\n", maxid);
  return 0;
}
```

## L2-032 彩虹瓶

```cpp
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e3+10;
int fun(vector<int> v, int m){
  stack<int> s;
  int d = 1;
  for ( int i = 0 ; i < v.size() ; i ++ ){
    s.push(v[i]);
    while( s.size() && s.top() == d ){
      d ++;
      s.pop();
    }
    if ( s.size() > m ) return false;
  }
  if (s.empty() ) return true;
  else return false;
}
int main(){
  int n, m ,k;
  scanf("%d%d%d", &n, &m, &k);
  while ( k -- ){
    vector<int> v;
    int d = 1;
    for ( int i = 0 ; i < n ; i ++ ){
      int x; scanf("%d", &x);
      v.push_back(x);
    }
    if ( fun(v, m) ) printf("YES\n");
    else printf("NO\n");
  }
  return 0;
}
```

## L2-033 简单计算器

```cpp
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e3+10;
void fun(vector<int> vi, vector<char> vc){
  stack<int> si;
  stack<char> sc;
  for ( int i = 0 ; i < vi.size() ; i ++ ) si.push(vi[i]);
  for ( int i = 0 ; i < vc.size() ; i ++ ) sc.push(vc[i]);
  int ans=1;
  while ( sc.size() ) {
    int n1, n2;
    n1 = si.top(); si.pop();
    n2 = si.top(); si.pop();
//   cout<<n1<<" "<<n2<<" "<<sc.top()<<endl;
    if ( sc.top() == '+' ) ans = n2 + n1;
    if ( sc.top() == '-' ) ans = n2 - n1;
    if ( sc.top() == '*' ) ans = n2 * n1;
    if ( sc.top() == '/' ) {
      if ( n1 == 0){
        printf("ERROR: %d/0\n", n2);
        return ;
      }
      ans = n2 / n1;
    }
    sc.pop();
    si.push(ans);
  }
  printf("%d\n", si.top());
}
int main(){
  int n; scanf("%d",&n);
  vector<int> vi;
  vector<char> vc;
  for ( int i = 1 ; i <= n ; i ++ ){
    int x; scanf("%d", &x);
    vi.push_back(x);
  }
  for ( int i = 1 ; i < n ; i ++ ){
    char c[5]; scanf("%s", c);
    vc.push_back(c[0]);
  }
  fun(vi, vc);
  return 0;
}
```

## L2-034 口罩发放

```cpp
#include<bits/stdc++.h>
#include<map>
#define N 10003
using namespace std;
struct Node {
  string name;
  string id;
  int flag;
  int hh,mm;
  int t;
  int idx;
} a[N],ans[N];

int d,p;
int t,s;
int anscnt;

map<string, int> mp;
map<string, int> vis;

bool cmp(Node x,Node y) {
  if(x.t != y.t)
  return x.t < y.t;
  else return x.idx < y.idx;
}
bool check(string s) {
  int len = s.length();
  if(len != 18) return false;
  for(int i=0; i<len; ++i) {
    if(!isdigit(s[i])) {
      return false;
    }
  }
  return true;
}
int main() {
  scanf("%d%d",&d,&p);
  for(int i=1; i<=d; i++) {
    scanf("%d%d",&t,&s);
    for(int j=1; j<=t; j++) {
      cin >> a[j].name >> a[j].id;

  scanf("%d%d:%d",&a[j].flag,&a[j].hh,&a[j]
.mm);
      a[j].t = a[j].hh*60 + a[j].mm;
      a[j].idx = j;
      if(mp.find(a[j].id) == mp.end()) {
        mp[a[j].id] = 0;
      }
      if(a[j].flag == 1 && check(a[j].id)
&& vis.find(a[j].id) == vis.end()) {
        vis[a[j].id] = 0;
        ans[anscnt++] = a[j];
      }
    }
    sort(a+1,a+t+1,cmp);
    int cnt = 0;
    for(int j=1; j<=t && cnt<s; j++) {
      if(check(a[j].id) && (!mp[a[j].id] ||
(i-mp[a[j].id])>p)) {
        cout << a[j].name << " " << a[j].id
<< endl;
        cnt++;
        mp[a[j].id] = i;
      }
    }
  }
  for(int i=0; i<anscnt; ++i) {
    cout << ans[i].name << " " << ans[i].id
<< endl;
  }
  return 0;
}
```

## L2-035 完全二叉树的层序遍历

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e3+10;
int n, tree[31];
void create(int i) {
    if (i > n) return;
    create(2 * i);
    create(2 * i + 1);
    scanf("%d", &tree[i]);
}
int main() {
    cin >> n;
    create(1);
    for ( int i = 1; i <= n; i ++)
        printf ("%d%c", tree[i], i==n ?
'\n' : ' ');
    return 0;
}
```

## L2-036 网红点打卡攻略

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e3+10;
int main(){
  int n, m;
  scanf("%d%d", &n, &m);
  int mp[n+5][n+5];
  for ( int i = 0 ; i <= n ; i ++ )
    for ( int j = 0 ; j <= n ; j ++ )
      mp[i][j] = mp[j][i] = 0;
  for ( int i = 1 ; i <= m ; i ++ ){
    int x, y, w; scanf("%d%d%d", &x, &y,
&w);
    mp[x][y] = mp[y][x] = w;
  }

  vector<PII> ans;
  int q, minw=INF; scanf("%d", &q);
  for ( int i = 1 ; i <= q ; i ++ ){
    int k; scanf("%d", &k);
    vector<int> v;
    set<int> s;
    v.push_back(0);
    for (int j = 0 ; j < k ; j ++ ){
      int x; scanf("%d", &x);
      v.push_back(x);
      s.insert(x);
    }
    v.push_back(0);

    int w = 0;
    if(k != n || s.size() != n) continue;

    bool flag = true;
    for (int j = 0 ; j < v.size()-1 ; j
++ ){
      if (mp[v[j]][v[j+1]]!=0) w +=
mp[v[j]][v[j+1]];
      else {
        flag = false;
        break;
      }
    }
    if (!flag) continue;
    minw = min(minw, w);
    ans.push_back({i, w});
  }
  printf("%d\n", ans.size());
  for ( int i = 0 ; i < ans.size() ; i ++ )
    if ( ans[i].second == minw ){
      printf("%d %d\n", ans[i].first,
ans[i].second);
      break;
    }
  return 0;
}
```

## L2-037 包装机

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e3+10;
int main(){
  int n, m, s; scanf("%d%d%d", &n, &m, &s);
  queue<int> q[n+5];
  stack<int> sta;
  for ( int i = 1 ; i <= n ; i ++ ){      //
第i 号轨道
    char str[m+5]; scanf("%s",str);
    for ( int j = 0 ; j < m ; j ++ )      //
第j 个物品
      q[i].push(str[j]-'A');
  }
  int x;
  while ( scanf("%d", &x) && x != -1){
    if ( x > 0 && q[x].size() ){
      int t = q[x].front();
      q[x].pop();
      if(sta.size() == s) {
        printf("%c", sta.top()+'A');
        sta.pop();
      }
      sta.push(t);
    }
    if ( x == 0 && sta.size() ){
      printf("%c", sta.top()+'A');
      sta.pop();
    }
  }
  return 0;
}
```

## L2-038 病毒溯源

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e4+10;
int vis[N], maxk;
vector<int> v[N], ans, t;
void dfs(int id, int k){
  if ( k > maxk ){
    maxk = k;
    ans = t;
  }
  for ( int i = 0 ; i < v[id].size() ; i
++ ){
    t.push_back(v[id][i]);
    dfs(v[id][i], k+1);
    t.pop_back();
  }
  return ;
}
int main(){
  int n;
  scanf("%d", &n);
  for ( int i = 0 ; i < n ; i ++ ){
    int k; scanf("%d", &k);
    while ( k -- ){
      int x; scanf("%d", &x);
      vis[x] = 1;
      v[i].push_back(x);
    }
    sort(v[i].begin(), v[i].end());
  }

  int id; //病毒源头
  for ( int i = 0 ; i < n ; i ++ ){
    if( vis[i] == 0 ) {
      id = i;
      break;
    }
  }

  dfs(id, 1);

  printf("%d\n%d", maxk, id);
  for (int i = 0 ; i < ans.size() ; i ++ )
    printf(" %d",ans[i]);
  return 0;
}
```

## L2-039 清点代码库

```
#include <bits/stdc++.h>
using namespace std;
#define PII pair<int, int>

const int INF = 0x3f3f3f3f;
const int N = 1e4 + 10;

struct cmp
{ // 自定义set 排序
  bool operator()(const pair<int,
vector<int>> &a, const pair<int,
vector<int>> &b) const
  {
    if (a.first != b.first)
      return a.first > b.first;
    else
      return a.second < b.second;
  }
};

int main()
{
  int n, m;
  scanf("%d%d", &n, &m);
  set<vector<int>> st;
// 存模块
  map<vector<int>, int> mp;
// 存每个模块的个数
  set<pair<int, vector<int>>, cmp> St; //
排序
  for (int i = 0; i < n; i++)
  {
    vector<int> v;
    for (int j = 0; j < m; j++)
    {
      int x;
      scanf("%d", &x);
      v.push_back(x);
    }
    mp[v]++;
    st.insert(v);
  }

  printf("%d\n", st.size());
  // 把所有模块存入 ST 排序
  set<vector<int>>::iterator it;
  for (it = st.begin(); it != st.end();
it++)
    St.insert({mp[*it], *it});
  // 输出 ST
  set<pair<int, vector<int>>>::iterator
ite;
  for (ite = St.begin(); ite != St.end();
ite++)
  {
    cout << (*ite).first;
    for (int i = 0; i <
(*ite).second.size(); i++)
      cout << ' ' << (*ite).second[i];
    cout << endl;
  }
  return 0;
}
```

## L2-040 哲哲打游戏

```
#include<bits/stdc++.h>
using namespace std;
#define PII pair<int,int>
const int INF = 0x3f3f3f3f;
const int N = 1e3+10;
vector<int> e[100010];
int que[110];
int main(){
  int n, m; scanf("%d%d", &n, &m);
  for (int i = 1 ; i <= n ; i ++) {
    int t; scanf("%d", &t);
    while (t--){
      int x; scanf("%d", &x);
      e[i].push_back(x);
    }
  }

  int now = 1;

  while (m--) {
    int x, y; scanf("%d%d", &x, &y);
    if (x == 0)
      now = e[now][y - 1];
    else if (x == 1) {
      que[y] = now;
      printf("%d\n", now);
    }
    else if (x == 2)
      now = que[y];
  }
  printf("%d\n", now);
  return 0;
}
```

## L2-041 插松枝

```
// 队列（推送器）以及栈（小盒子）模拟
#include <bits/stdc++.h>
using namespace std;

stack<int> st;
queue<int> q;
vector<int> ans[1010];

int main()
{
  int n, m, k;
  cin >> n >> m >> k;
  while (n--)
  {
    int x;
    cin >> x;
    q.push(x);
  }
  int i = 0, lst = 0;
  while (q.size() || st.size())
  {
    lst = (ans[i].size() == 0 ? 99999 :
ans[i].back());
    if (st.size() && st.top() <= lst)
    { // 先用小盒子
      ans[i].push_back(st.top());
      st.pop();
    }
    else if (q.size() && q.front() <= lst)
    { // 再用推送器
      ans[i].push_back(q.front());
      q.pop();
    }
    else if (st.size() < m && q.size())
    { // 推送器放到小盒子里
      st.push(q.front());
      q.pop();
    }
    else
    {
      i++; // 小盒子满了，下一根
    }
    if (ans[i].size() == k)
      i++;
  }
  for (int j = 0; j <= i; j++)
  {
    if (ans[j].size() == 0)
      continue;
    for (int k = 0; k < ans[j].size(); k++)
    {
      if (k)
        cout << " ";
      cout << ans[j][k];
    }
    cout << "\n";
  }
  return 0;
}
```

## L2-042 老板的作息表

```
//直接排序，然后输出两个不相邻区间的尾和头即可
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5+10;
struct node{int h1, m1, s1, h2, m2,
s2; }a[maxn];
bool cmp(node x, node y){
    if(x.h1 != y.h1)return x.h1<y.h1;
    if(x.m1 != y.m1)return x.m1<y.m1;
    if(x.s1 != y.s1)return x.s1<y.s1;
}
int main(){
    int n;  cin>>n;
    for(int i = 1; i <= n; i++){
        scanf("%d:%d:%d - %d:%d:%d",
&a[i].h1, &a[i].m1, &a[i].s1, &a[i].h2,
&a[i].m2, &a[i].s2);
    }
    a[0].h1 = 0, a[0].m1 = 0, a[0].s1 = 0;
    a[0].h2 = 0, a[0].m2 = 0, a[0].s2 = 0;
    a[n+1].h1 = 23, a[n+1].m1 = 59,
a[n+1].s1 = 59;
    sort(a,a+n+2, cmp);
    for(int i = 1; i <= n+1; i++){
        if(a[i].h1==a[i-1].h2 &&
a[i].m1==a[i-1].m2 && a[i].s1==a[i-
1].s2)continue;
        printf("%02d:%02d:%02d
- %02d:%02d:%02d\n", a[i-1].h2, a[i-1].m2,
a[i-1].s2, a[i].h1, a[i].m1, a[i].s1);
    }
    return 0;
}
```

## L2-043 龙龙送外卖

```
//题意：一棵树上不断加点，求每次加点后访问所有
点至少一次的最短距离是多少
//思路：可以贪心，外卖员最后的位置应该在距离外
卖站最远的送餐地址。所以最短路程 = 需要经过的
边数*2 - max(外卖站到送餐地址)
//每次搜索新增送餐点到外卖站未被标记的点（记忆
化）可以实现 O（n）
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5+10;
int f[maxn], rt, dep[maxn];
int vis[maxn], tmp; //tmp 每次累加就行
void dfs(int u ,int dis){//暴力跑一遍路程
    if(u==rt || vis[u]){
        tmp += dis;  return ;
    }
    vis[u] = 1;
    dfs(f[u], dis+2);
}
int calc(int u){//点 u 到 rt 的距离
    if(u==rt || dep[u])return dep[u];
    return dep[u] = calc(f[u])+1;
}
int main(){
    int n, m;  cin>>n>>m;
    for(int i = 1; i <= n; i++){
        int x;  cin>>x;  f[i] = x;
        if(f[i]==-1)rt=i;
    }
    int mx = 0; //维护当前最大距离
    while(m--){
        int x;  cin>>x;
        dfs(x, 0); //每次从这个点跑就行，根直
接当做普通点处理
        mx = max(mx, calc(x));
        cout<<tmp-mx<<"\n";
    }
    return 0;
}
```

## L2-044 大众情人

```
//用距离感建有向图，Floyd 计算全图最短路即可，
不知道哪里错了，重写一遍过了
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 1e6+10;
const int inf = 1e9+10;
struct node{int to, d;};
int e[510][510];
int sex[510];
int main(){

ios::sync_with_stdio(0),cin.tie(0),cout.ti
e(0);
    int n;  cin>>n;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            e[i][j] = inf;
    //floyed
    for(int i = 1; i <= n; i++){
        string op;  cin>>op;
        if(op=="F")sex[i] = 0; else sex[i]
= 1;//男1
        int k;  cin>>k;
        for(int j = 1; j <= k; j++){
            int to, d; char ch;
            cin>>to>>ch>>d;
            e[i][to] = d;
        }
    }
    for(int k = 1; k <= n; k++){
        for(int i = 1; i <= n; i++){
            for(int j = 1; j <= n; j++){
                if(e[i][j] >
e[i][k]+e[k][j]){
                    e[i][j] =
e[i][k]+e[k][j];
                }
            }
        }
    }
    //solve
    vector<int>girl, boy;
    map<int,int>p;
    for(int i = 1; i <= n; i++){
        int d = -1;
        for(int j = 1; j <= n; j++){
            if(sex[j]!=sex[i]){
                d = max(d, e[j][i]);
            }
        }
        if(d != -1){
            p[i] = d;
            if(sex[i])boy.push_back(i);
            else girl.push_back(i);
        }
    }
    sort(girl.begin(), girl.end(), [&p](int
x, int y){
        return p[x]==p[y]? x<y : p[x]<p[y];
    });
    sort(boy.begin(), boy.end(), [&p](int
x, int y){
        return p[x]==p[y]? x<y : p[x]<p[y];
    });
    for(int i = 0; i < girl.size(); i++){
        if(p[girl[i]] == p[girl[0]]){
            if(i!=0)cout<<" ";
            cout<<girl[i];
        }else break;
    }
    cout<<"\n";
    for(int i = 0; i < boy.size(); i++){
        if(p[boy[i]] == p[boy[0]]){
            if(i!=0)cout<<" ";
            cout<<boy[i];
        }else break;
    }
    cout<<"\n";
    return 0;
}
```

## L3-001 凑零钱

```
#include<bits/stdc++.h>
using namespace std;
int n, k;
int a[10010], b[10010];
int dfs(int i, int sum, int c) {
  int flag = 0;
  if(sum > k) return 0;
  if(sum == k) {
    for(int pp = 0; pp < c; pp++) {
      printf(pp == 0 ? "%d" : " %d",
b[pp]);
    }
    printf("\n");
    return 1;
  }
  for(int j = i + 1; j < n; j++) {
    b[c] = a[j];
    flag = dfs(j, sum + a[j], c+1);
    if(flag) return 1;
  }
  return 0;
}
int main() {
  scanf("%d%d", &n, &k);
  int flag = 0, sum = 0;
  for(int i = 0; i < n; i++) {
    scanf("%d", &a[i]);
    sum += a[i];
  }
  sort(a, a+n);
  if(sum >= k) {
    for(int i = 0; i < n; i++) {
      if(a[i] > k) break;
      b[0] = a[i];
      flag = dfs(i, a[i], 1);
      if(flag) break;
    }
  }
  if(flag == 0) printf("No Solution\n");
  return 0;
}
```

## L3-002 特殊堆栈

```
#include<bits/stdc++.h>
using namespace std;
int n, num;
stack<int> s;
vector<int> v;
string op;
int main() {
  cin >> n;
  vector<int>::iterator it;
  while(n--) {
    cin >> op;
    if(op == "Push") {
      cin >> num;
      s.push(num);
      it = lower_bound(v.begin(), v.end(),
num);
      v.insert(it, num);
    } else if (op == "Pop") {
      if(s.empty()) cout << "Invalid" <<
endl;
      else {
        cout << s.top() << endl;
        v.erase(lower_bound(v.begin(),
v.end(), s.top()));
        s.pop();
      }
    } else {
      if(s.empty()) cout << "Invalid" <<
endl;
      else {
        cout << v[(s.size()+1)/2-1] <<
endl;
      }
    }
  }
  return 0;
}
```

## L3-003 社交集群

```
#include<bits/stdc++.h>
using namespace std;
int n, num, k, sum;
vector<int> v[1010];
int tot[1010], pre[1010], ans[1010];
int find(int x) {
  if(x == pre[x]) return pre[x];
  return pre[x] = find(pre[x]);
}
void join(int x, int y) {
  int fx = find(x), fy = find(y);
  if(fx != fy) {
    pre[fx] = fy;
    tot[fy] += tot[fx];
  }
}
int main() {
  scanf("%d", &n);
  sum = 0;
  for(int i = 1; i <= n; i++) {
    pre[i] = i;
    tot[i] = 1;
  }
  for(int i = 1; i <= n; i++) {
    scanf("%d:", &num);
    for(int j = 0; j < num; j++) {
      scanf("%d", &k);
      v[k].push_back(i);
    }
  }
  for(int i = 1; i <= 1000; i++) {
    int len = v[i].size();
    for(int j = 0; j < len-1; j++) {
      join(v[i][j], v[i][j+1]);
    }
  }
  for(int i = 1; i <= n; i++) {
    if(i == find(i)) ans[sum++] = tot[i];
  }
  printf("%d\n", sum);
  sort(ans, ans+sum);
  for(int i = sum-1; i >= 0; i--) {
    printf(i == sum-1 ? "%d" : " %d",
ans[i]);
  }
  printf("\n");
  return 0;
}
```

## L3-004 肿瘤诊断

```
#include<bits/stdc++.h>
using namespace std;
long long n, m, l, t, num, vv, sum;
long long v[80000][130];
long long dir[6][2];
struct point {
  long long x, y;
};
void bfs(long long sx, long long sy) {
  queue<point> q;
  point now, next;
  now.x = sx;
  now.y = sy;
  q.push(now);
  v[sx][sy] = 0;
  while(!q.empty()) {
    now = q.front();
    q.pop();
    vv++;
    for(long long i = 0; i < 6; i++) {
      next.x = now.x + dir[i][0];
      next.y = now.y + dir[i][1];
      if(next.y < 0 && next.y >= m)
continue;//判断左右是否越界,越界就跳过
      if(next.x < 0 || next.x >= n*l)
continue;//判断前后、上下是否越界,越界就跳过
```

```
        else {//如果在前后和上下的总范围内
          if(i == 2 || i == 3) {//前后
            if(next.x / n != now.x / n)
continue;//不在一个平面就跳过
          }
          if(i == 4 || i == 5) {//上下
             if(next.x / n == now.x / n)
continue;//在一个平面就跳过
          }
        }
        if(v[next.x][next.y] == 1) {
          v[next.x][next.y] = 0;
          q.push(next);
        }
      }
    }
}
int main() {
  sum = 0;
  scanf("%lld%lld%lld%lld", &n, &m, &l,
&t);
  dir[0][0] = 0;
  dir[0][1] = 1;
  dir[1][0] = 0;
  dir[1][1] = -1;
  dir[2][0] = 1;
  dir[2][1] = 0;
  dir[3][0] = -1;
  dir[3][1] = 0;
  dir[4][0] = n;
  dir[4][1] = 0;
  dir[5][0] = -n;
  dir[5][1] = 0;
  for(long long i = 0; i < n*l; i++) {
    for(long long j = 0; j < m; j++) {
      scanf("%lld", &v[i][j]);
    }
  }
  for(long long i = 0; i < n*l; i++) {
    for(long long j = 0; j < m; j++) {
      if(v[i][j]) {
        vv = 0;
        bfs(i, j);
        if(vv >= t) sum += vv;
      }
    }
  }
  printf("%lld\n", sum);
  return 0;
}
```

## L3-005 垃圾箱分布

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
using namespace std;
int n, m, k, ds, len, uu, vv, flag,
sumdis, mindis, realsumdis, ans,
maxmindis;
int way[1030][1030], to[1030],
dis[1030][1030], vis[1030];
char u[10], v[10];
void Dijstl(int s) {
  to[s] = 0;
  for(int i = 1; i <= n+m; i++) {
    int minn = inf, next = -1;
    for(int j = 1; j <= n+m; j++) {
      if(vis[j] == 0 && to[j] < minn) {
        minn = to[j];
        next = j;
      }
    }
    if(next == -1) break;
    else
      vis[next] = 1;
    for(int j = 1; j <= n+m; j++)
      if(vis[j] == 0 && to[next] +
dis[next][j] < to[j]) to[j] = to[next] +
dis[next][j];
  }
}
int main() {
  flag = 0;
  maxmindis = -1;
  scanf("%d%d%d%d", &n, &m, &k, &ds);
  for(int i = 1; i <= n+m; i++) {
    for(int j = 1; j <= n+m; j++) {
      if(i == j) way[i][j] = 0;
      else
        way[i][j] = inf;
    }
  }
  for(int i = 0; i < k; i++) {
    scanf("%s%s%d", u, v, &len);
    if(u[0] == 'G') {
      uu = 0;
      for(int j = 1; j < strlen(u); j++)
        uu = uu * 10 + (int)(u[j] - '0');
      uu += n;
    } else {
      uu = atoi(u);
    }
    if(v[0] == 'G') {
      vv = 0;
      for(int j = 1; j < strlen(v); j++)
        vv = vv * 10 + (int)(v[j] - '0');
      vv += n;
    } else {
      vv = atoi(v);
    }
    way[uu][vv] = len;
    way[vv][uu] = len;
  }
  for(int i = n+1; i <= n+m; i++) {
    memset(vis, 0, sizeof(vis));
    memset(to, inf, sizeof(to));
    for(int ii = 1; ii <= n+m; ii++) {
      for(int jj = 1; jj <= n+m; jj++) {
        dis[ii][jj] = way[ii][jj];
      }
    }
    Dijstl(i);
    sumdis = 0;
    mindis = inf;
    int flag2 = 0;
    for(int j = 1; j <= n; j++) {
      if(to[j] > ds || to[j] == inf) {
        flag2 = 1;
        break;
      }
      sumdis += to[j];
      if(i != j) mindis = min(mindis,
to[j]);
    }
    if(flag2 == 0) {
      flag = 1;
      if(mindis > maxmindis) {
        ans = i;
        maxmindis = mindis;
        realsumdis = sumdis;
      } else if(mindis == maxmindis) {
        if(sumdis < realsumdis) {
          ans = i;
          maxmindis = mindis;
          realsumdis = sumdis;
        }
      }
    }
  }
  if(flag) printf("G%d\n%d.0 %.1lf\n",
ans-n, maxmindis, realsumdis*1.0/n);
  else
    printf("No Solution\n");
  return 0;
}
```

## L3-006 迎风一刀斩

```
//超级注释版
#include<bits/stdc++.h>
using namespace std;
//1.分别存储两个图形的斜边(2 个点), 顶点数,
vector<int> v[2], n;
//2.特判情况:四边形直角腰, 矩形个数
vector<int>len; int flag;
//1.找到斜边
void deal(int id, vector<int>& x,
vector<int>& y){
  int sz = x.size(); set<int>st;
  for(int i = 0; i < sz; i++){
    //相邻点横纵坐标都不等: 这两点构成斜边。
    if(x[i]!=x[(i+1)%sz] &&
y[i]!=y[(i+1)%sz]){
      st.insert(i);  st.insert((i+1)%sz);
      //如果是四边形: 存储直角腰的长度
      if(sz==4)len.push_back(abs(x[(i+2)%4]-
x[(i+3)%4])+abs(y[(i+2)%4]-y[(i+3)%4]));
    }
  }
  if(st.size()==0){//没有斜边, 所以是矩形
    //存下两条直角边
    v[id].push_back(abs(x[2]-x[0]));
    v[id].push_back(abs(y[2]-y[0]));
    flag++; //矩形个数+1
  }else{
    //存储斜边(2 个端点)
    for(int i : st){
      v[id].push_back(x[i]);
      v[id].push_back(y[i]);
    }
  }
}
//2.情况判断
void solve(){
  //最多也就三边形+五边形, 超过 8 个点就错。
  if(n[0]<=5 && n[1]<=5 && n[0]+n[1]<=8){
    if(flag==2){//两个矩形
      //只要矩形 A(x,y)两条直接边有一条能和矩形
B 合上就行
      int
x=v[0][0],y=v[0][1],c=v[1][0],d=v[1][1];

  if(x==c||x==d||y==c||y==d){cout<<"YES\n";
return;}
    }
    if(flag==0){//没有矩形
      //如果没有斜边, 不成立
      if(v[0].size()==4 && v[1].size()==4){
        //特判直角腰

  if(n[0]==4&&n[1]==4&&len[0]!=len[1]){cout
<<"NO\n";return;}
        //存下两条直角边(斜边分别做垂直的直角三
角形)
        int x=abs(v[0][2]-
v[0][0]),y=abs(v[0][3]-v[0][1]); if(x>y)
swap(x,y);
        int c=abs(v[1][2]-
v[1][0]),d=abs(v[1][3]-v[1][1]); if(c>d)
swap(c,d);
        //当且仅当直角边都相等, 斜边相等

  if(x==c&&y==d){cout<<"YES\n";return;}
      }
    }
    //一个矩形的情况不存在
  }
  cout<<"NO\n";
}
int main(){
  int T;  cin>>T;
  while(T--){
    //1. 变量全部初始化
    flag = 0; n.clear();  len.clear();
    v[0].clear();  v[1].clear();
    //2. 输入两个多边形
    for(int i = 0; i < 2; i++){
      int k;  cin>>k;  n.push_back(k);
      vector<int>x(k), y(k);
      for(int j = 0; j < k;
j++)cin>>x[j]>>y[j];
      //2.1 找到斜边
      deal(i,x,y);
    }
    //3. 结论判断
    solve();
  }
  return 0;
}
```

## L3-007 天梯地图

```
//AC
#include<bits/stdc++.h>
using namespace std;
const int maxn = 550;
int n, m, s, t;
int e[2][maxn][maxn], dist[2][maxn],
vis[2][maxn], pre[2][maxn], w[2][maxn];
void Dijkstra(int rk){
  memset(dist[rk],0x3f,sizeof(dist[rk]));
  memset(vis[rk],0,sizeof(vis[rk]));
  memset(pre[rk],-1,sizeof(pre[rk]));
  dist[rk][s] = 0;
  for(int i = 0; i < n; i++){
    int u = -1, _min = 1e9;
    for(int j = 0; j < n; j++){
      if(!vis[rk][j] && dist[rk][j]<_min){
        _min = dist[rk][j];  u = j;
      }
    }
    if(u==-1)return ;
    vis[rk][u] = 1;
    for(int j = 0; j < n; j++){
      if(!vis[rk][j] &&
dist[rk][j]>dist[rk][u]+e[rk][u][j]){
        dist[rk][j] =
dist[rk][u]+e[rk][u][j];
        pre[rk][j] = u;
        //距离
        if(rk==0){
          w[rk][j] =
w[rk][u]+1;//+1?!!:WA4!!

        }
        //时间
        if(rk==1){
          w[rk][j] =
w[rk][u]+e[!rk][u][j];//WA2!
        }
      }else if(!vis[rk][j] && dist[rk][j]
== dist[rk][u]+e[rk][u][j]){
        //距离
        if(rk==0){
          if(w[rk][j] > w[rk][u]+1){
            w[rk][j] = w[rk][u]+1;
            pre[rk][j] = u;
          }
        }
        //时间
        if(rk==1){
          if(w[rk][j] >
w[rk][u]+e[!rk][u][j]){
            w[rk][j] =
w[rk][u]+e[!rk][u][j];
            pre[rk][j] = u;
          }
        }
      }
    }
  }
}
void Print(int rk, int x){
  if(x==-1){
    return ;
  }else{
    Print(rk, pre[rk][x]);
    printf(" %d =>", x);
  }
}
int main(){
  memset(e,0x3f,sizeof(e));
  cin>>n>>m;
  for(int i = 1; i <= m; i++){
    int a, b, on, le, ti;
    cin>>a>>b>>on>>le>>ti;
    if(on==1){
      e[0][a][b] = le;
      e[1][a][b] = ti;
    }else{
      e[0][a][b] = le;
      e[1][a][b] = ti;
      e[0][b][a] = le;
      e[1][b][a] = ti;
    }
  }
  cin>>s>>t;
  Dijkstra(0);
  Dijkstra(1);
  int ok = 1, i = pre[0][t], j = pre[1][t];
  while(i!=-1 && j!=-1){
    if(pre[0][i] != pre[0][j]){ok=0;break;}
    i = pre[0][i];
    j = pre[1][j];
  }
  if(ok){
    printf("Time = %d;",dist[1][t]);
    printf(" Distance = %d:",dist[0][t]);
    Print(1,pre[1][t]);
    printf(" %d\n",t);
    return 0;
  }
  printf("Time = %d:",dist[1][t]);
  Print(1,pre[1][t]);
  printf(" %d\n",t);
  printf("Distance = %d:",dist[0][t]);
  Print(0,pre[0][t]);
  printf(" %d",t);
  return 0;
}
```

## L3-008 喊山

```
#include<bits/stdc++.h>
using namespace std;
int n, m, k, u1, u2, num;
int vis[10010];
vector<int> v[10010];
struct shan {
  int id, sum;
//  friend bool operator <(shan a, shan b)
{//21 分
//    return a.id > b.id;
//  }
};
void bfs(int s) {
//  priority_queue<shan> q;//21 分
  queue<shan> q;
  shan now, next;
  now.id = s;
  now.sum = 0;
  vis[s] = 1;
  q.push(now);
  int ans = 0, step = 0;
  while(!q.empty()) {
//    now = q.top();//21 分
    now = q.front();
    q.pop();
    if(now.sum > step) {
      ans = now.id;
      step++;
    }
    for(int i = 0; i < v[now.id].size();
i++) {
      next.id = v[now.id][i];
      next.sum = now.sum + 1;
      if(vis[next.id] == 0) {
        q.push(next);
        vis[next.id] = 1;
      }
    }
  }
  printf("%d\n", ans);
}
int main() {
  scanf("%d%d%d", &n, &m, &k);
  for(int i = 0; i < m; i++) {
    scanf("%d%d", &u1, &u2);
    v[u1].push_back(u2);
    v[u2].push_back(u1);
  }
  while(k--) {
    memset(vis, 0, sizeof(vis));
    scanf("%d", &num);
    bfs(num);
  }
  return 0;
}
```

## L3-009 长城

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 5e5+10;
LL x[maxn], y[maxn];
int stk[maxn], top;
set<int>se;
bool check(int a, int b, int c){//向量 ab 在
ac 下面(kab<kac), b 是凹点
  return (x[c]-x[a])*(y[b]-y[a])<=(x[b]-
x[a])*(y[c]-y[a]);
}
int main(){
  ios::sync_with_stdio(false);
  int n;  cin>>n;
  for(int i = 0; i < n; i++){
    cin>>x[i]>>y[i];
    if(top>=1){
      while(top>=2 && check(i,stk[top-
1],stk[top-2]))top--;//b 是凹点不要它了
      if(stk[top-1])se.insert(stk[top-
1]);//找到凹点了入栈
    }
    stk[top++] = i;
  }
  cout<<se.size()<<endl;
  return 0;
}
```

## L3-010 是否完全二叉搜索树

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1010;
int Tree[maxn];
void update(int root, int val){
  if(!Tree[root])
    Tree[root] = val;
  else if(val > Tree[root])
    update(root*2, val);
  else
    update(root*2+1,val);
}
int main(){
  int n;  cin>>n;
  for(int i = 1; i <= n; i++){
    int x;  cin>>x;  update(1,x);
  }
  int ok = 0, cnt = 0;
  for(int i = 1; i < maxn; i++){
    if(Tree[i]){
      if(ok)cout<<" ";
      else ok = 1;
      cout<<Tree[i];
      cnt = i;
    }
  }
  if(cnt > n)cout<<"\nNO\n";
  else cout<<"\nYES\n";
  return 0;
}
```

## L3-011 直捣黄龙

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 250;
map<string,int>ma;
map<int,string>mb;
int tot = 1;
int getid(string s){
  if(ma.count(s))return ma[s];
  else{
    mb[tot] = s;
    ma[s] = tot;
    tot++;
    return ma[s];
  }
}
int n, k, s, t;
int e[maxn][maxn], w[maxn];
int dist[maxn], vis[maxn], pre[maxn],
cnt[maxn], weight[maxn], cc[maxn];
void Dijkstra(int u){
  memset(dist, 0x3f,sizeof(dist));
  memset(pre,-1,sizeof(pre));
  dist[u] = 0; cnt[u] = 0; weight[u]=w[u];
cc[u] = 1;
  for(int i = 1; i <= n; i++){
    int v = -1, minn = 1e9;
    for(int j = 1; j <= n; j++){
      if(!vis[j] && dist[j]<minn){
        minn = dist[j];
        v = j;
      }
    }
    vis[v] = 1;
    for(int j = 1; j <= n; j++){
      if(!vis[j] &&
dist[j]>dist[v]+e[v][j]){
        dist[j] = dist[v]+e[v][j];
        cc[j] = cc[v];
        cnt[j] = cnt[v]+1;
        weight[j] = weight[v]+w[j];
        pre[j] = v;
      }else if(!vis[j] &&
dist[j]==dist[v]+e[v][j]){
        cc[j] += cc[v];//+=
        if(cnt[j]<cnt[v]+1){
          cnt[j] = cnt[v]+1;
          weight[j] = weight[v]+w[j];
          pre[j] = v;
        }else if(cnt[j]==cnt[v]+1){
          if(weight[j]<weight[v]+w[j]){
            weight[j] = weight[v]+w[j];
            pre[j] = v;
          }
        }
      }
    }
  }
}
int main(){
  cin>>n>>k;
  string a,b;  cin>>a>>b;
  s = getid(a);  t = getid(b);
  memset(e,0x3f,sizeof(e));
  for(int i = 1; i < n; i++){
    string a; int b;  cin>>a>>b;
    w[getid(a)] = b;
  }
  for(int i = 1; i <= k; i++){
    string a, b;  cin>>a>>b;
    int aa = getid(a), bb = getid(b);
    int cc;  cin>>cc;
    e[aa][bb] = e[bb][aa] = cc;
  }
  Dijkstra(s);
  vector<string>vec;
  int x = t;
  while(x!=-1){
    vec.push_back(mb[x]);
    x = pre[x];
  }
  reverse(vec.begin(),vec.end());
  for(int i = 0; i < vec.size(); i++){
    if(i==vec.size()-1)cout<<vec[i]<<endl;
    else cout<<vec[i]<<"->";
  }
  cout<<cc[t]<<" "<<dist[t]<<"
"<<weight[t]<<"\n";
  return 0;
}
```

## L3-012 水果忍者

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 10010;
struct seg{ double x, y1, y2; }s[maxn];
bool cmp(seg a, seg b){return a.x<b.x;}
double
maxk,mink,ansmaxk,ansmink,ansx,ansy;
int main(){
  int n;  cin>>n;
  for(int i = 1; i <= n; i++)
    cin>>s[i].x>>s[i].y1>>s[i].y2;//y1 在
上,y2 在下
  sort(s+1,s+n+1,cmp);
  for(int i = 1; i <= n; i++){
    ansmaxk = 1e9;
    ansmink = -1e9;
    int j;
    for(j = 1; j <= n; j++){
      if(j==i)continue;
      if(i<j){// j 在 i 右侧
        maxk = (s[i].y2-s[j].y1)/(s[i].x-
s[j].x);
        mink = (s[i].y2-s[j].y2)/(s[i].x-
s[j].x);
      }else{ //j 在 i 左侧
        maxk = (s[i].y2-s[j].y2)/(s[i].x-
s[j].x);
        mink = (s[i].y2-s[j].y1)/(s[i].x-
s[j].x);
      }
      if(ansmaxk<mink ||
ansmink>maxk)break;
      if(maxk<ansmaxk){
        ansmaxk = maxk;
        ansx = s[j].x;
        ansy = s[j].y1;
      }
      ansmink = max(ansmink, mink);
    }
    if(j==n+1){//存在解
      //线段 i 上取了最低点, 则另一条线段要取最
高点

  printf("%.0lf %.0lf %.0lf %.0lf\n",s[i].x
,s[i].y2,ansx,ansy);
      return 0;
    }
  }
  return 0;
}
```

## L3-013 非常弹的球

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  double w, p; cin>>w>>p;
  double E = 1000, g = 9.8;
  double s = 1, sum = 0;
  while(s>1e-8){//精度
    s = 2*E/(w/100*g);
    sum += s;
    E *= (100-p)/100;
  }
  printf("%0.3lf",sum);
  return 0;
}
```

## L3-014 周游世界

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 10010;
int minCnt, minTrans;
vector<int>path, tempath;
int line[maxn][maxn]; //维护两点之间的线路归
属
int count(vector<int>a){//给出路径, 计算换乘
  int cnt = -1, preLine = 0;
  for(int i = 1; i < a.size(); i++){
    if(line[a[i-1]][a[i]] != preLine)cnt++;
    preLine = line[a[i-1]][a[i]];
  }
  return cnt;
}
vector<int>G[maxn]; int vis[maxn];
void dfs(int u, int end, int cnt){
  if(u==end){
    if(cnt<minCnt || (cnt==minCnt &&
count(tempath)<minTrans)){
      minCnt = cnt;
      minTrans = count(tempath);
      path = tempath;
    }
    return ;
  }
  for(int i = 0; i < G[u].size(); i++){
    int v = G[u][i];
    if(!vis[v]){
      vis[v] = 1;
      tempath.push_back(v);
      dfs(v,end,cnt+1);
      vis[v] = 0;
      tempath.pop_back();
    }
  }
}
int main(){
  int n, m, k, pre, tmp, a, b;
  cin>>n;
  for(int i = 1; i <= n; i++){
    cin>>m>>pre;
    for(int j = 2; j <= m; j++){
      cin>>tmp;
      G[pre].push_back(tmp);
      G[tmp].push_back(pre);
      line[pre][tmp] = i;
      line[tmp][pre] = i;
      pre = tmp;
    }
  }
  cin>>k;
  for(int i = 1; i <= k; i++){
    cin>>a>>b;
    //memset(vis,0,sizeof(vis));
    minCnt = 1e9, minTrans = 1e9;
    tempath.clear();  tempath.push_back(a);
    vis[a] = 1;
    dfs(a,b,0);
    vis[a] = 0;//memset
    if(minCnt == 1e9) {
      printf("Sorry, no line is 
available.\n");
      continue;
       }
    cout<<minCnt<<"\n";
    int preLine = 0, preTrans = a;//上次的线
路，以及转乘点
    for(int j = 1; j < path.size(); j++){
      if(line[path[j-
1]][path[j]]!=preLine){
        if(preLine!=0)printf("Go by the 
line of company #%d from %04d to %04d.\n", 
preLine, preTrans, path[j-1]);
        preLine = line[path[j-1]][path[j]];
        preTrans = path[j-1];
      }
    }
    printf("Go by the line of company #%d 
from %04d to %04d.\n", preLine, preTrans, 
b);
  }
  return 0;
}
```

## L3-015 球队“食物链”

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 50;
int n, T0, ok;
int e[maxn][maxn], vis[maxn];
vector<int>q;
void dfs(int u, int k){
  for(int i = 1; i <= n; i++){
    int flag = 0;//剩余队伍不存在战胜第一只队
伍
    for(int j = 1; j <= n; j++){
      if(!vis[j] && e[j][T0]){
        flag = 1; break;
      }
    }
    if(flag && !ok && !vis[i] && e[u][i]){
      vis[i] = 1;  q.push_back(i);
      if(k==n-1 && e[i][T0]){
        ok = 1;
        for(int j = 0; j < q.size(); j++){
          if(j!=0)cout<<" ";
          cout<<q[j];
        }
      }else{
        dfs(i,k+1);
      }
      vis[i] = 0;  q.pop_back();
    }
  }
}
int main(){
  cin>>n;  cin.get();
  for(int i = 1; i <= n; i++){
    for(int j = 1; j <= n; j++){
      char ch;  cin>>ch;
      if(ch=='W')e[i][j] = 1;
      if(ch=='L')e[j][i] = 1;
    }
    cin.get();
  }
  for(int i = 1; i <= n; i++){//另i为队首
    vis[i] = 1;  q.push_back(i);
    T0 = i;  dfs(i,1);
    if(ok)return 0;
    vis[i] = 0;  q.pop_back();
  }
  cout<<"No Solution";
  return 0;
}
```

## L3-016 二叉搜索树的结构

```
#include<bits/stdc++.h>
using namespace std;
struct node{int l=-1, r=-1, fa=-1, h;};
map<int,node>Tree;
void insert(int u, int h, int v){
  if(u==-1)return ;
  int uu = (v<u ? Tree[u].l : Tree[u].r);
  if(uu!=-1){
    insert(uu,h+1,v);
  }else{
    if(v<u)Tree[u].l = v;
    else Tree[u].r = v;
    Tree[v].fa = u;
    Tree[v].h = h;
  }
}
bool judge(int u, int a, int b, string 
lk){
  if(lk=="root")return u==a;;
  if(Tree.find(a)==Tree.end() || 
Tree.find(b)==Tree.end())return false;
  if(lk=="siblings")return 
Tree[a].fa==Tree[b].fa;
  if(lk=="parent")return Tree[a].l==b || 
Tree[a].r==b;
  if(lk=="left")return Tree[b].l == a;
  if(lk=="right")return Tree[b].r == a;
  if(lk=="level")return 
Tree[a].h==Tree[b].h;
}
int main(){
  int n, rt, t;
  cin>>n>>rt; //rt 是根
  for(int i = 2; i <= n; i++){
    cin>>t;
    insert(rt,1,t);
  }
  int m, a=0, b=0;  cin>>m;
  for(int i = 1; i <= m; i++){
    string s, lk;  cin>>a>>s;
    if(s=="and"){
      cin>>b>>s>>s;
      if(s=="siblings")lk = s;
      else cin>>s>>s>>lk;
    }else{
      cin>>s>>lk;
      if(lk=="parent")cin>>s>>b;
      else if(lk!="root")cin>>s>>s>>b;
    }
    if(judge(rt,a,b,lk))cout<<"Yes\n";
    else cout<<"No\n";
  }
  return 0;
}
```

## L3-017 森森快递

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 1e5+10;
struct seg{int x, y;}sg[maxn];
bool cmp(seg a, seg b){return 
a.y!=b.y?a.y<b.y:a.x<b.x;}
LL rmq[maxn<<2], tag[maxn<<2], c[maxn];
#define lch p<<1
#define rch p<<1|1
void pushdown(int p){
  if(tag[p]){
    tag[lch] += tag[p], tag[rch]+=tag[p];
    rmq[lch] += tag[p], rmq[rch]+=tag[p];
    tag[p] = 0;
  }
}
void pushup(int p){
  rmq[p] = min(rmq[lch], rmq[rch]);
}
void build(int p, int l, int r){
  tag[p] = 0;
  if(l==r){
    rmq[p] = c[l];
    return ;
  }else{
    int m = l+r>>1;
    build(lch,l,m);
    build(rch,m+1,r);
    pushup(p);
  }
}
void update(int p, int l, int r, int L, 
int R, int v){
  if(l>R || r<L)return ;
  if(L<=l && r<=R){
    rmq[p] += v;  tag[p] += v;
    return ;
  }
  pushdown(p);
  int mid = l+r>>1;
  update(lch,l,mid,L,R,v);
  update(rch,mid+1,r,L,R,v);
  pushup(p);
}
LL query(int p, int l, int r, int L, int 
R){
  if(l>R || r<L)return (1ll<<60);
  if(L<=l && r<=R)return rmq[p];
  pushdown(p);
  LL mid = l+r>>1, ans = 1ll<<60;
  ans = min(ans, query(lch,l,mid,L,R));
  ans = min(ans, query(rch,mid+1,r,L,R));
  return ans;
}
int main(){
  int n, q;
  cin>>n>>q;
  for(int i = 1; i < n; i++)
    cin>>c[i];
  build(1,1,n-1); //1-(n-1)号城市分别对应 i
与 i+1 的边
  for(int i = 1; i <= q; i++){
    cin>>sg[i].x>>sg[i].y;

  if(sg[i].x>sg[i].y)swap(sg[i].x,sg[i].y);
  }
  sort(sg+1,sg+q+1,cmp);
  LL ans = 0;
  for(int i = 1; i <= q; i++){
    //cout<<sg[i].x+1<<" "<<sg[i].y<<" ";
    LL res = query(1,1,n-
1,sg[i].x+1,sg[i].y);//因为编号从 0 开始，所
以 x+1。
    //cout<<res<<"\n";
    ans += res;
    if(res)update(1,1,n-
1,sg[i].x+1,sg[i].y,-res);
  }
  cout<<ans<<endl;
  return 0;
}
```

## L3-018 森森美图

```
#include<bits/stdc++.h>
using namespace std;
const int inf = 1e9+10;
const int maxn = 110;
struct point{ int x, y;  double dis;};
bool operator!=(point a, point b){return
a.x!=b.x||a.y!=b.y;}
bool operator==(point a, point b){return
a.x==b.x&&a.y==b.y;}
int n, m;
double sc[maxn][maxn];//分数
point s, t;
void input(){
  cin>>n>>m;
  for(int i = 1; i <= n; i++)
    for(int j = 1; j <= m; j++)
      cin>>sc[i][j];
  cin>>s.y>>s.x>>t.y>>t.x;
  s.x++;s.y++;t.x++;t.y++;
  s.dis = sc[s.x][s.y];
}
int flag;//1 上半部分，0 下半部分
double f[maxn][maxn]; //到i,j 为止的最小值
int dir[][2]= {{0,1},{1,0},{-1,0},{0,-
1},{-1,-1},{1,-1},{-1,1},{1,1}}; //上下左右
+前后左右
int cross(point a,point b,point p){//三角形
行列式公式,判断三点是否在一个直线上
  return (b.x-a.x)*(p.y-a.y)-(p.x-
a.x)*(b.y-a.y);
}
bool check(point p){//检查p 是否合法（越界）
  if(p.x<1||p.x>n||p.y<1||p.y>m)return
false;//越界
  if(flag && p!=s&&p!=t &&
cross(s,t,p)<=0)return false;//1:上半部分但
点在下面(起点终点不算)
  if(!flag && p!=s&&p!=t &&
cross(s,t,p)>=0)return false;//2.下半部分但
点在上面
  if(p.dis>=f[p.x][p.y])return false;//不是
最小值
  return true;
}
void bfs(){
  //init
  queue<point>q;
  for(int i = 1; i <= n; i++)
    for(int j = 1; j <= m; j++)
      f[i][j] = inf;
  //search
  if(check(s)){
    f[s.x][s.y] = s.dis;
    q.push(s);
  }
  while(q.size()){
    point now = q.front(); q.pop();
    point next;
    for(int i = 0; i < 8; i++){
      next.x=now.x+dir[i][0];
      next.y=now.y+dir[i][1];
      if(i<4)next.dis =
f[now.x][now.y]+sc[next.x][next.y];
      else
next.dis=f[now.x][now.y]+sc[next.x][next.y
]+(sc[next.x][next.y]+sc[now.x][now.y])*(s
qrt(2)-1);
      if(check(next)){
        f[next.x][next.y] = next.dis;
        q.push(next);
      }
    }
  }
}
int main(){
  input();
  double ans = 0;
  flag = 1; bfs(); ans += f[t.x][t.y];//搜
上面
  flag = 0; bfs(); ans += f[t.x][t.y];//搜
下面
  ans -= sc[s.x][s.y]+sc[t.x][t.y];//重复
  printf("%.2f\n",ans);
  return 0;
}
```

## L3-019 代码排版

```
#include<bits/stdc++.h>
using namespace std;
//判断语句块类型
int judge(string dat, int i){
  //WA3:当前位置是if 并且不是在字符串内
  if(dat.find("if", i)==i && (dat[i+2]=='
'||dat[i+2]=='('))return 2;
  if(dat.find("for",i)==i && (dat[i+3]==' '
||dat[i+3]=='('))return 3;
   if(dat.find("while",i)==i &&
(dat[i+5]==' '||dat[i+5]=='('))return 5;
   if(dat.find("else",i)==i && dat[i+4]=='
')return 4;
  return 0;//普通语句
}
//输出前删除多余空格，并输出当前对应的空格
void erase_space(string dat,int
&i){while(dat[i]==' ')i++;}
void print_space(int sp){for(int
i=0;i<sp;i++)putchar(' ');}
int main(){
  string dat;  getline(cin,dat);

  //处理int main()  找i 和)输出
  int l = dat.find('i',0), r =
dat.find(')',0);
  cout<<dat.substr(l,r-l+1)<<"\n{\n";

  //处理其他，按照行分类
  int tmp, space = 2;//语句类型,空格数
  int flag, debt=0;//单句标记，层数（补全缺少
的）
  for(int i = dat.find('{')+1,j=0,k; i <
dat.size(); ){
    erase_space(dat,i);//删除每行前的空格
    if(dat[i]=='{' || dat[i]=='}'){
      if(dat[i]=='{'){
        print_space(space);
        printf("{\n");
        space += 2;
        i++;
        continue;
      }else{
        space -= 2;
        print_space(space);
        printf("}\n");
        i++;
        if(space==0)break;//main 的}输完就结
束了

        // 【重复】单句特判
        erase_space(dat,i);
        while(debt && judge(dat,i)!=4){
          space -= 2;
          print_space(space);
          printf("}\n");
          debt--;
        }
      }
    }else if((tmp=judge(dat,i))){
      print_space(space);
      //处理for,while,if,+()或者else
      if(tmp==4){
        printf("else");
        k = i+3;
      }else{
        cout<<dat.substr(i,tmp)<<" ";
        i += tmp;
        erase_space(dat, i);
        //考虑if()中也有()条件的情况
        k = i; int t = 0;
        while(1){
          if(dat[k]=='(')t++;
          if(dat[k]==')')t--;
          if(!t)break;
          k++;
        }
        cout<<dat.substr(i,k-i+1);
      }
      //预处理{}的内容,考虑单句特判
      int m = k+1;
      erase_space(dat,m);
      if(dat[m] != '{'){//单句标记
        printf(" {\n");
        flag = 1;
        debt++;
        i = m;
      }else{
        printf(" {\n");
        flag = 0;
        i = m+1;
      }
      space += 2;
    }else{//普通语句
      int ed = dat.find(';', i);
      print_space(space);
      cout<<dat.substr(i,ed-i+1)<<"\n";
      i = ed+1;

      //这是单句内的语句
      if(flag && debt){
        space -= 2;
        print_space(space);
        printf("}\n");
        debt--;

        // 【重复】单句特判
        erase_space(dat,i);
        while(debt && judge(dat,i)!=4){
          space -= 2;
          print_space(space);
          printf("}\n");
          debt--;
        }
      }
    }
  }
  return 0;
}
```

## L3-020 至多删三个字符

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 1e6+10;
LL f[maxn][5];
int main(){
  string s;  cin>>s; s = "0"+s;//从1 开始
  f[0][0] = 1;
  for(int i = 1; i < s.size(); i++){
    for(int j = 0; j <= 3; j++){//删0-3 个
      f[i][j] = f[i-1][j]+f[i-1][j-1];//第i
个删还是不删
      for(int k=i-1; k>=1 && (i-k)<=j; k-
-){//去重
        if(s[k]==s[i]){//如果当前字符一样，那
么前面的重复统计了
          f[i][j] -= f[k-1][j-(i-k)];
          break;
        }
      }
    }
  }
  LL ans = 0;
  for(int i = 0; i <= 3; i++)
    ans += f[s.size()-1][i];
  cout<<ans<<endl;
  return 0;
}
```

## L3-021 神坛

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 5e5+10;
struct point{LL x, y;}pp[maxn], tmp[maxn];
bool cmp(point a, point b){return
a.x*b.y>a.y*b.x;}
int main(){
  int n;  cin>>n;
  for(int i = 0; i < n; i++)
    cin>>pp[i].x>>pp[i].y;
  double ans = 1e18;
  for(int i = 0; i < n; i++){
    int cc = 0;
    for(int j = 0; j < n; j++){
      if(i==j)continue;
      tmp[cc].x = pp[j].x-pp[i].x;
      tmp[cc].y = pp[j].y-pp[i].y;
      cc++;
    }
    sort(tmp,tmp+cc,cmp);
    for(int j = 0; j < cc; j++)

  ans=min(ans,abs(0.5*(tmp[j].x*tmp[(j+1)%c
c].y-tmp[(j+1)%cc].x*tmp[j].y)));
  }
  printf("%.3f\n",ans);
  return 0;
}
```

## L3-022 地铁一日游

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 210;
const int inf = 1e9+10;
int G[maxn][maxn];
vector<int>st[maxn]; int ed[maxn],
vis[maxn];
void dfs(int u){
  for(int i = 0; i < st[u].size(); i++){
    int v = st[u][i];
    if(!vis[v]){
      vis[v] = 1;
      dfs(v);
    }
  }
}
int main(){
  //input
  int n, m, k;  cin>>n>>m>>k;
  for(int i = 1; i <= n; i++)
    for(int j = 1; j <= n; j++)
      G[i][j] = inf;
  for(int i = 1; i <= m; i++){
    int a, b, dis;
    cin>>a; ed[a] = 1;
    while(cin>>dis>>b){
      G[a][b] = min(G[a][b], dis);
      G[b][a] = min(G[b][a], dis);
      a = b;
      if(getchar()=='\n')break;
    }
    ed[a] = 1;
  }
  //solve
  for(int k = 1; k <= n; k++)//Floyd
    for(int i = 1; i <= n; i++)
      for(int j = 1; j <= n; j++)
        if(i!=j)G[i][j] =
min(G[i][j],G[i][k]+G[k][j]);
  for(int i = 1; i <= n; i++){//从点i 出发
    map<int,int>cost;//各种费用能到的最远距离
    for(int j = 1; j <= n; j++){//遍历到每个
点的费用去更新距离
      if(G[i][j]==inf)continue;
      cost[2+G[i][j]/k] =
max(cost[2+G[i][j]/k],G[i][j]);
    }
    for(int j = 1; j <= n; j++){//更新点i 能
到达的最远点或者端点
      if(G[i][j]==cost[2+G[i][j]/k] ||
i!=j&&ed[j]&&G[i][j]!=inf){
        st[i].push_back(j);
      }
    }
  }
  int q;  cin>>q;
  for(int i = 1; i <= q; i++){
    int x;  cin>>x;
    memset(vis,0,sizeof(vis));
    vis[x] = 1;
    dfs(x);
    for(int j = 1; j <= n; j++)
      if(vis[j])st[x].push_back(j);
    sort(st[x].begin(), st[x].end());
    st[x].erase(unique(st[x].begin(),
st[x].end()), st[x].end());
    for(int j = 0; j < st[x].size(); j++){
      if(j!=0)cout<<" ";
      cout<<st[x][j];
    }
    cout<<"\n";
  }
  return 0;
}
```

## L3-023 计算图

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 5e4 + 10;

struct node
{
  int op, left, right; // 运算符和数值
  double val;         // 当前节点的值
  int post;       // 后继节点的
} a[maxn];
map<int, map<int, map<int, double>>> f; //
记忆化数组

// 第一个参数为结点，第二个参数决定是否求导，
第三个参数是对谁求导
double calc(int nd, int key, int p)
{
  if (f[nd][key][p])
    return f[nd][key][p];
  int id = a[nd].op;
  if (id == 0)
    return f[nd][key][p] = (key == 0 ?
a[nd].val : (nd == p ? 1 : 0));
  if (id == 1)
    return f[nd][key][p] = calc(a[nd].left,
key, p) + calc(a[nd].right, key, p);
  if (id == 2)
    return f[nd][key][p] = calc(a[nd].left,
key, p) - calc(a[nd].right, key, p);
  if (id == 3)
    return f[nd][key][p] = (key ?
calc(a[nd].left, key, p) *
calc(a[nd].right, 0, p) + calc(a[nd].left,
0, p) * calc(a[nd].right, key, p) :
calc(a[nd].left, key, p) *
calc(a[nd].right, key, p));
  if (id == 4)
    return f[nd][key][p] = (key ?
exp(calc(a[nd].left, 0, p)) *
calc(a[nd].left, key, p) :
exp(calc(a[nd].left, key, p)));
  if (id == 5)
    return f[nd][key][p] = (key ? 1 /
(calc(a[nd].left, 0, p)) *
(calc(a[nd].left, key, p)) :
log(calc(a[nd].left, key, p)));
  if (id == 6)
    return f[nd][key][p] = (key ?
cos(calc(a[nd].left, 0, p)) *
calc(a[nd].left, key, p) :
sin(calc(a[nd].left, key, p)));
}

int main()
{
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    cin >> a[i].op;
    if (a[i].op == 0)
    {
      cin >> a[i].val;
    }
    else if (a[i].op <= 3)
    {
      cin >> a[i].left >> a[i].right;
      a[a[i].left].post = 1;
      a[a[i].right].post = 1;
    }
    else
    {
      cin >> a[i].left;
      a[a[i].left].post = 1;
    }
  }
  int ed = 0, ok = 0;
  while (a[ed].post)
    ed++;
  printf("%0.3lf\n", calc(ed, 0, -1));
  for (int i = 0; i < n; i++)
  {
    if (a[i].op == 0)
    {
      if (ok)
        cout << " ";
      printf("%0.3lf", calc(ed, 1, i));
      ok = 1;
    }
  }
  return 0;
}
```

## L3-024 Oriol 和 David(无满分解)

## L3-025 那就别担心了

```
#include<bits/stdc++.h>
using namespace std;
//#define int long long
vector<int>edge[510];
int s, t;
int dp[510];
int dfs(int u) {
    if (dp[u]) {
        return dp[u];
    }
    if (u == t) {
        return 1;
    }
    for (auto v : edge[u]) {
        dp[u] += dfs(v);
    }
    return dp[u];
}
```

```
int flag;
int vis[510];
void bfs1() {
    queue<int>Q;
    Q.push(s);
    while (!Q.empty()) {
        int now = Q.front(); Q.pop();
        if (vis[now]) continue;
        vis[now] = 1;
        if (!dp[now]) {
            flag = 1; break;
        }
        if (now == t) continue;
        for (auto v : edge[now]) {
            Q.push(v);
        }
    }
}
signed main() {
    int n, m;
    cin >> n >> m;
    while (m--) {
        int u, v;
        cin >> u >> v;
        edge[u].push_back(v);
    }
    cin >> s >> t;
    dfs(s);
    cout << dp[s] -dp[t] << " ";
    dp[t] = 1;
    bfs1();
    if (!flag) cout << "Yes\n";
    else cout << "No\n";
}
```

## L3-026 传送门

```
/*
+ 一开始初始化成n条链，传送门对应链上的结点，
将所有需要新增或者删除的传送门的y值离散化，存
入链上。
+ 对于每个操作，实际上要做的是“分别查询两个结
点各自所在链上的左右端点”和“将两个结点的后继结
点交换”，用splay 可以做到logq
*/
#include<bits/stdc++.h>
using namespace std;
typedef long  long LL;
const int maxn = 5e5+10, inf =1e9+10;
int n, m;
struct node{int x1,x2,y1,y2;}qr[maxn];
vector<int>rt[maxn];
#define l(u) ch[u][0]
#define r(u) ch[u][1]
int fa[maxn], ch[maxn][2], tot, X[maxn];
int sf(int u){return u== r(fa[u]);}
bool isrt(int u){return u!=l(fa[u])&&u!=
r(fa[u]);}
void rot(int u){
  int v=fa[u],f= sf (u);
  if(!isrt(v))ch[fa[v]][sf(v)]= u;
  ch[v][f]=ch[u][f^1],fa[ch[v][f]]= v;
  fa[u]=fa[v],ch[u][f^1]=v,fa[v]= u;
}
int newnode(){int u=++tot;
fa[u]=l(u)=r(u)= 0 ; return u;}
void splay(int u){ for(;!isrt(u);
rot(u))if(!isrt(fa[u])&&sf(fa[u])==sf(u))r
ot(fa[u]);}
int fdl(int u){splay(u); for(;l(u);
u=l(u)); splay(u); return u;}
int fdr(int u){splay(u); for(;r(u);
u=r(u)); splay(u); return u;}
int main(){
  ios::sync_with_stdio(false);
  cin>>n>>m;
  for(int i=1; i <= m; i++){
    char ch;  cin>>ch;
    cin>>qr[i].x1>>qr[i].x2>>qr[i].y1;
  }
  for(int i=1; i <= n; i++){
    rt[i].push_back(0);
rt[i].push_back(inf);
  }
  for(int i=1; i <= m; i++){
    rt[qr[i].x1].push_back(qr[i].y1);
    rt[qr[i].x2].push_back(qr[i].y1);
  }
  for(int i=1; i <= n; i++){
    sort(rt[i].begin(),rt[i].end());

  rt[i].resize(unique(rt[i].begin(),rt[i].e
nd()) - rt[i].begin());
  }
  for(int i=1; i <= m; i++){
    int y = qr[i].y1;
    qr[i].y1=
lower_bound(rt[qr[i].x1].begin(),rt[qr[i].
x1].end(),y)- rt[qr[i].x1].begin();

  qr[i].y2=lower_bound(rt[qr[i].x2].begin()
,rt[qr[i].x2].end(),y)-
rt[qr[i].x2].begin( );
  }
  for(int i=1; i <= n; i++){
    for(int j=0; j<rt[i].size(); j++){
      rt[i][j]=newnode(); X[rt[i][j]]= i;
    }
    for(int j=0; j<rt[i].size()-1; j++){

  r(rt[i][j])=rt[i][j+1],fa[rt[i][j+1]]=
rt[i][j];
    }
  }
  LL ans=(LL)n*(n+1)*(2*n+1)/6;
  for(int i=1; i <= m; i++){
    int
x1=qr[i].x1,x2=qr[i].x2,y1=qr[i].y1,y2=qr[
i].y2;
    int u=rt[x1][y1],v= rt[x2][y2];
    int
lu=X[fdl(u)],ru=X[fdr(u)],lv=X[fdl(v)],rv=
X[fdr(v)];
    ans-=(LL)lu*ru+(LL)lv*rv;
    ans+=(LL)lu*rv+(LL) lv*ru;
    splay(u),splay(v);
    int u2=r(u),v2=r(v);
    r(u)=v2,r(v)=u2,fa[v2]=u,fa[u2]=v;
    cout<<ans<<"\n";
  }
  return 0;
}
```

## L3-027 可怜的复杂度(无满分解)

## L3-029 还原文件

```
#include <bits/stdc++.h>
using namespace std;

#define endl '\n'
#define inf 0x3f3f3f3f
#define mod7 1000000007
#define mod9 998244353
#define m_p(a, b) make_pair(a, b)
#define mem(a, b) memset((a), (b),
sizeof(a))
#define io                         \
  ios::sync_with_stdio(false); \
  cin.tie(0);                    \
  cout.tie(0)
#define debug(a) cout << "Debuging...|" <<
#a << ": " << a << "\n";
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> pii;

#define MAX 300000 + 50
ull n, m, k, x;
ull tr[MAX];
vector<ull> v[1005];
ull hx[1005];

ull base = 13331;
ull hhash[MAX], mul[MAX];
void init()
{
  mul[0] = 1;
  hhash[0] = 0;
  for (int i = 1; i <= n; i++)
    mul[i] = mul[i - 1] * base;
  for (int i = 1; i <= n; i++)
    hhash[i] = hhash[i - 1] * base + tr[i];
}
ull getHash(int l, int r)
{
  return hhash[r] - hhash[l - 1] * mul[r -
l + 1];
}
void haxi(int id)
{
  for (int i = 0; i < v[id].size(); ++i)
  {
    hx[id] = hx[id] * base + v[id][i];
  }
}
ull ans[1005];
bool vis[1005];
void work()
{
  cin >> n;
  for (int i = 1; i <= n; ++i)
    cin >> tr[i];
  init();
  cin >> m;
  for (int i = 1; i <= m; ++i)
  {
    cin >> k;
    while (k--)
    {
      cin >> x;
      v[i].push_back(x);
    }
    haxi(i);
  }
  int l = 1;
  for (int i = 1; i <= m; ++i)
  {
    for (int j = m; j >= 1; --j)
    {
      if (vis[j])
        continue;
      int r = l + (int)v[j].size() - 1;
      if (getHash(l, r) == hx[j])
      {
        l = r;
        vis[j] = 1;
        ans[i] = j;
        break;
      }
    }
  }
  for (int i = 1; i <= m; ++i)
  {
    if (i != 1)
      cout << " ";
    cout << ans[i];
  }
}

int main()
{
```

```cpp
  work();
  return 0;
}
```

## L3-030 可怜的简单题

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int N = 21544400 + 7;
#define minus(x, y) (1ll * x - y < 0 ? 1ll
* x - y + mod : 1ll * x - y)
#define plus(x, y) (1ll * x + y >= mod ?
1ll * x + y - mod : 1ll * x + y)
ll mod;
int mu[N];
ll smu[N];
bool vis[N];
unordered_map <ll, ll> sum_mu;
int primes[N], cnt;
ll n, m;
ll mul(ll a, ll b) {
    if (mod <= 1000000000) return a * b %
mod;
    else if (mod <= 1000000000000ll) return
(((a * (b >> 20) % mod) << 20) + (a * (b &
((1 << 20) - 1)))) % mod;
    else {
        ll d = (ll)floor(a * (long double)b
/ mod + 0.5);
        ll res = (a * b - d * mod) % mod;
        if (res < 0) res += mod;
        return res;
    }
}
ll qpow(ll a, ll b)
{
  if(mod == 1) return 0;
  ll res = 1;
  a = a % mod;
  while(b) {
    if(b & 1) res = mul(res, a);
    a = mul(a, a);
    b >>= 1;
  }
  return res;
}
ll inv(ll x)
{
  return qpow(x, mod - 2);
}
void init(int n)
{
  mu[1] = 1, vis[1] = 1;
  for(int i = 2; i <= n; ++ i) {
    if(vis[i] == 0) {
      primes[ ++ cnt] = i;
      mu[i] = -1;
    }
      for(int j = 1; j <= cnt && i *
primes[j] <= n; ++ j) {
      vis[i * primes[j]] = 1;
      if(i % primes[j] == 0) {
        mu[i * primes[j]] = 0;
        break;
      }
      mu[i * primes[j]] -= mu[i];
    }
  }
  for(int i = 1; i <= n; ++ i)
    smu[i] = plus(smu[i - 1], mu[i]);
  return ;
}
inline ll g_sum(ll x)
{
  return x;
}
inline ll get_sum_mu(ll x)
{
  if(x <= N - 7) return smu[x];
  if(sum_mu.find(x) != sum_mu.end()) return
sum_mu[x];
  ll ans = 1;
  for(ll l = 2, r; l <= x; l = r + 1) {
    r = x / (x / l);
    ll tmp = mul((r - l + 1), get_sum_mu(x
/ l));
    ans = minus(ans, tmp);
  }
  ans = (ans % mod + mod) % mod;
  sum_mu[x] = ans / g_sum(1ll);
  return ans / g_sum(1ll);
}
void solve(ll m)
{
  ll ans = 1;
  for(ll l = 1, r; l <= m; l = r + 1) {
    r = m / (m / l);
    ll tmp = mul((get_sum_mu(r) -
get_sum_mu(l - 1)), mul(m / l, inv(m - m /
l)));
    ans = minus(ans, tmp);
  }
  ans = (ans + mod) % mod;
  printf("%lld\n", ans);
  return ;
}
signed main()
{
  scanf("%lld%lld", &m, &mod);
  if(m == 1) return printf("1\n"), 0;
  init(N - 7);
  solve(m);
  return 0;
}
```

## L3-031 千手观音

```cpp
// 题意：依次给出大小递增的字符串，用.分隔，求
字符串大小规则
// 思路：将读入的字符串用.分隔后和上一个逐位比
较（当前仅当长度相等时才行），然后建图跑拓扑排
序即可。
// 1，对于相对顺序无法确定，队列换成优先队列，
按字典序小的排列
// 2. 对于vector 存图会超时，用链式前向星存图
// 3. 大点指向小点建图会WA，图的结构不同
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e6 + 10;

// 离散化
unordered_map<string, int> mp;
string st[maxn];
int tot = 0;

// 建图
int head[maxn], in[maxn];
struct Edge
{
  int to, next;
} edge[maxn];
int m = 0;
void add_edge(int u, int v)
{
  edge[m].to = v;
  edge[m].next = head[u];
  head[u] = m++;
}
struct node
{
  int id;
  string s;
  friend bool operator<(node a, node b)
{ return a.s > b.s; }
};

int main()
{
  ios::sync_with_stdio(0), cin.tie(0),
cout.tie(0);
  memset(head, -1, sizeof head);
  // input
  int n;
  cin >> n;
  vector<string> pre;
  for (int i = 1; i <= n; i++)
  {
    string s;
    cin >> s;
    vector<string> now;
    int cnt = 0;
    for (int j = 0; j < s.size(); j++)
    {
      if (s[j] == '.')
      {
        string t = s.substr(j - cnt, cnt);
        if (!mp[t])
          mp[t] = ++tot, st[tot] = t;
        now.push_back(t);
        cnt = 0;
      }
      else if (j == s.size() - 1)
      {
        string t = s.substr(j - cnt, cnt +
1);
        if (!mp[t])
          mp[t] = ++tot, st[tot] = t;
        now.push_back(t);
      }
      else
      {
        cnt++;
      }
    }
    // creat graph
    if (i == 1)
      pre = now;
    else
    {
      if (pre.size() == now.size())
      {
        for (int i = 0; i < pre.size();
i++)
        {
          if (pre[i] != now[i])
          {
            add_edge(mp[pre[i]],
mp[now[i]]); // 小的向大的连一条边
            in[mp[now[i]]]++;
            break;
          }
          else
            continue;
        }
      }
      pre = now;
    }
  }
  // topu sort
  priority_queue<node> q;
  for (int i = 1; i <= tot; i++)
  {
    if (in[i] == 0)
      q.push({i, st[i]});
  }
  vector<string> res;
  while (q.size())
  {
    node t = q.top();
    q.pop();
    res.push_back(t.s);
    for (int j = head[t.id]; j != -1; j =
edge[j].next)
    {
      in[edge[j].to]--;
      if (in[edge[j].to] == 0)
        q.push({edge[j].to,
st[edge[j].to]});
    }
    if (res.size() == tot)
      break;
  }
  for (int i = 0; i < res.size(); i++)
  {
    if (i != 0)
      cout << ".";
    cout << res[i];
  }
  return 0;
}
```

## L3-032 关于深度优先搜索和逆序对的题应该不会很难吧这件事

```
#include<iostream>
#include<vector>
using namespace std;

typedef long long LL;
const int N = 300010, P = 1e9+7;

vector<int> g[N];
int sz[N],tr[N];
int n,root;
int sum=1,s1,s2;

void add(int x,int y)
{
  for(int i=x;i<N;i+=(i&-i))
    tr[i]+=y;
}

int query(int x)
{
  int res=0;
  for(int i=x;i;i-=(i&-i))
    res+=tr[i];
  return res;
}

void dfs(int u,int fa)
{
  add(u,1);
  s1=(s1+query(n)-query(u))%P;

  sz[u]=1;
  int cnt=0;
  for(auto &j:g[u])
  {
    if(j==fa) continue;
      dfs(j,u);
      sz[u]+=sz[j];
      cnt++;
  }

  for(int i=1;i<=cnt;i++)
    sum=(LL)sum*i%P;

  s2=(s2+n-query(n)-sz[u]+1)%P;
  add(u,-1);
}

int main()
{
  scanf("%d%d",&n,&root);
  for(int i=0;i<n-1;i++)
  {
    int a,b;scanf("%d%d",&a,&b);
        g[a].push_back(b);
        g[b].push_back(a);
  }

  dfs(root,-1);

  int
ans=((LL)s1*sum+(LL)s2*sum%P*(P+1)/4)%P;
  printf("%d\n",ans);
  return 0;
}
```

## L3-033 科书般的亵渎

## 数据结构与算法题目集

### 7-1 最大子列和问题

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
  // dp[i]表示到i 为止最大的子数组和
  // dp[i+1] = max(dp[i] + a[i], a[i])
  int dp[100005] = {0};
  int a[100005];
  int k;
  cin >> k;
  for (int i = 0; i < k; i++)
  {
    cin >> a[i];
  }
  int sum = -1;
  dp[0] = a[0];
  for (int i = 1; i < k; i++)
  {
    dp[i] = max(dp[i - 1] + a[i], a[i]);
  }
  for (int i = 0; i < k; i++)
  {
    sum = max(sum, dp[i]);
  }
  if (sum < 0)
    sum = 0;
  printf("%d", sum);
  return 0;
}
```

### 7-2 一元多项式的乘法与加法运算

```
#include <iostream>
using namespace std;
int main()
{
  int n, a[1005] = {0}, coff, exp,
muti[2001] = {0}, add[1005] = {0};
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    cin >> coff >> exp;
    a[exp] = coff;
    add[exp] = coff;
  }
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    cin >> coff >> exp;
    for (int j = 0; j < 1005; j++)
    {
      muti[exp + j] += a[j] * coff;
    }
    add[exp] += coff;
  }
  int cnt = 0;
  for (int j = 2000; j >= 0; j--)
  {
    if (muti[j] != 0)
    {
      if (cnt)
      {
        cout << " ";
      }
      cout << muti[j] << " " << j;
      cnt++;
    }
  }
  if (!cnt)
  {
    cout << "0 0";
  }
  cnt = 0;
  cout << endl;
  for (int j = 1005; j >= 0; j--)
  {
    if (add[j] != 0)
    {
      if (cnt)
      {
        cout << " ";
      }
      cout << add[j] << " " << j;
      cnt++;
    }
  }
  if (!cnt)
  {
    cout << "0 0";
  }
  return 0;
}
```

### 7-3 树的同构

```
#include <iostream>
using namespace std;
struct Node
{
  char ch;
  int left = -1;
  int right = -1;
};
bool find(char ch1, char ch2, char ch3,
Node tree[], int size)
{
  for (int i = 0; i < size; i++)
  {
    if (ch1 == tree[i].ch)
    {
      if ((ch2 == tree[tree[i].left].ch &&
ch3 == tree[tree[i].right].ch)
        || (ch2 == tree[tree[i].right].ch &&
ch3 == tree[tree[i].left].ch))
        return true;
    }
  }
  return false;
}
int main(void)
{
#ifdef LOCAL_COMPILE
  freopen("in.txt", "r", stdin);
  freopen("out.txt", "w", stdout);
#endif
  // Build Tree
  Node treeA[11];
  Node treeB[11];
  treeA[10] = {'#'};
  treeB[10] = {'#'};
  // Init treeA
  int treeSize, treeSizeB;
  cin >> treeSize;
  getchar();
  char chBuf, leftBuf, rightBuf;
  for (int i = 0; i < treeSize; i++)
  {
    scanf("%c %c %c", &chBuf, &leftBuf,
&rightBuf);
    getchar();
    treeA[i].ch = chBuf;
    treeA[i].left = leftBuf == '-' ? 10 :
leftBuf - '0';
    treeA[i].right = rightBuf == '-' ? 10 :
rightBuf - '0';
  }
  // Init treeB
  cin >> treeSizeB;
  getchar();
  if (treeSize != treeSizeB)
  {
    printf("No");
    return 0;
  }
  for (int i = 0; i < treeSize; i++)
  {
    scanf("%c %c %c", &chBuf, &leftBuf,
&rightBuf);
    getchar();
    treeB[i].ch = chBuf;
    treeB[i].left = leftBuf == '-' ? 10 :
leftBuf - '0';
    treeB[i].right = rightBuf == '-' ? 10 :
rightBuf - '0';
  }
  // Traverse every node, judge if all same
  bool isIsomorphism = true;
  char ch1, ch2, ch3;
  for (int i = 0; i < treeSize; i++)
  {
    ch1 = treeA[i].ch;
    ch2 = treeA[treeA[i].left].ch;
    ch3 = treeA[treeA[i].right].ch;
    if (!find(ch1, ch2, ch3, treeB,
treeSize))
    {
      isIsomorphism = false;
      break;
    }
  }
  if (isIsomorphism)
  {
    printf("Yes");
  }
  else
  {
    printf("No");
  }
  return 0;
}
```

## 7-4 是否同一棵二叉搜索树

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1024 + 7;
int n, m;
int a[maxn], b[maxn];
void build1()
{
  memset(a, -1, sizeof a);
  for (int i = 0; i < n; ++i)
  {
    int id = 1, x;
    scanf("%d", &x);
    while (1)
    {
      if (a[id] == -1)
      {
        a[id] = x;
        break;
      }
      else if (x < a[id])
      {
        id *= 2;
      }
      else
        id = 2 * id + 1;
    }
  }
}
void build2()
{
  memset(b, -1, sizeof b);
  for (int i = 0; i < n; ++i)
  {
    int id = 1, x;
    scanf("%d", &x);
    while (1)
    {
      if (b[id] == -1)
      {
        b[id] = x;
        break;
      }
      else if (x < b[id])
      {
        id *= 2;
      }
      else
        id = 2 * id + 1;
    }
  }
}
int check()
{
  for (int i = 1; i < maxn; ++i)
  {
    if (a[i] != b[i])
      return 0;
  }
  return 1;
}
int main()
{
  while (cin >> n >> m)
  {
    if (n == 0)
      break;
    build1();
    for (int i = 0; i < m; ++i)
    {
      build2();
      if (check())
        cout << "Yes" << endl;
      else
        cout << "No" << endl;
    }
  }
  return 0;
}
```

## 7-5 堆中的路径

```
#include <iostream>
using namespace std;
#define maxn 1005
#define minn -10001
int heap[maxn], size;
void BuildHeap();
void insert(int);
int main(){
    int n, m, t;
    cin >> n >> m;
    BuildHeap();
    for(int i = 0; i < n; i++){
        cin >> t;
        insert(t);
    }
    for(int i = 0; i < m; i++){
        cin >> t;
        cout << heap[t];
        while(t > 1){
            cout << " " << heap[t/2];
            t/=2;
        }
        cout <<endl;
    }
    return 0;
}
void BuildHeap(){
    size = 0;
    heap[0] = minn;//0 位置不存数据，设置岗哨
}
void insert(int x){
    //插入结点形成小顶堆
    int i;
    for(i = ++size; heap[i/2] > x; i/=2){
        //小顶堆，如果父节点大于插入结点则二者
交换
        heap[i] = heap[i/2];
    }
    heap[i] = x;
}
```

## 7-6 列出连通集

```
#include <iostream>
#include <queue>
#include <string.h>
using namespace std;
// 输出有顺序，使用邻接矩阵存储方便遍历
int edge[15][15] = {0};
bool vis[15];
int n, e;
void dfs(int index)
{
  cout << index << " ";
  vis[index] = true;
  for (int i = 0; i < n; i++)
  {
    if (!vis[i] && edge[index][i] == 1)
    {
      dfs(i);
    }
  }
  return;
}
void bfs(int index)
{
  queue<int> q;
  while (!q.empty())
  {
    q.pop();
  }
  q.push(index);
  vis[index] = true;
  while (!q.empty())
  {
    int top = q.front();
    cout << top << " ";
    q.pop();
    for (int i = 0; i < n; i++)
    {
      if (!vis[i] && edge[top][i] == 1)
      {
        q.push(i);
        vis[i] = true;
      }
    }
  }
  return;
}
int main()
{
  cin >> n >> e;
  // memeset 在这里面<string.h>
  memset(vis, false, sizeof(vis));
  for (int i = 0; i < e; i++)
  {
    int from, to;
  cin >> from >> to;
  edge[from][to] = edge[to][from] = 1;
 }
 for (int i = 0; i < n; i++)
 {
  if (!vis[i])
  {
   cout << "{ ";
   dfs(i);
   cout << "}" << endl;
  }
 }
 for (int i = 0; i < n; i++)
 {
  vis[i] = false;
 }
 for (int i = 0; i < n; i++)
 {
  if (!vis[i])
  {
   cout << "{ ";
   bfs(i);
   cout << "}" << endl;
  }
 }
 return 0;
}
```

## 7-7 六度空间

```
#include <iostream>
#include <stdio.h>
#include <queue>
#include <string.h>
using namespace std;
int n, m;   //社交网络图的结点数N（1<N≤1000
表示人数）、边数M（≤33×N，表示社交关系数）
int a[1005][1005]={0};  //全局变量声明时值为
0，这里初始化以防万一
int vis[1005]={0};
int bfs(int index){
  int cnt = 0;
  memset(vis, 0, sizeof(vis));
  queue<int> q;
  q.push(index);
  cnt++;
  vis[index] = 1;
  while (q.size()) {
    index = q.front();
    for (int i = 1; i <= n; i++) {
      if (a[index][i]&&!vis[i]) {
        vis[i] = vis[index] + 1;
        if (vis[i] < 8) {
          q.push(i);
          cnt++;
        }
      }
    }
    q.pop();
  }
  return cnt;
}
int main(){
  cin >> n >> m;
  for(int i = 0; i < m; i++){
    int from, to;
    cin >> from >> to;
    a[from][to] = a[to][from] = 1;
  }
  for(int i = 1; i <= n; i++){
    printf("%d: %.2f%%\n", i,
100.0*bfs(i)/n);
  }
  return 0;
}
```

## 7-8 哈利·波特的考试

```
#include <stdio.h>
#include <stdlib.h>
#define MAXVEX 105
#define INFINITY 65535
void CreateGraph();
void Floyd();
void FindAnimal();
int FindMax(int i);
int G[MAXVEX][MAXVEX], Nv, Ne;
int D[MAXVEX][MAXVEX]; // 存储最短路径矩阵
int main()
{
 CreateGraph();
 FindAnimal();
 return 0;
}
void CreateGraph()
{
 // 用邻接矩阵表示图
 int i, j;
 int v1, v2, w;
 scanf("%d %d", &Nv, &Ne);
 for (i = 1; i <= Nv; i++)
 {
  for (j = 1; j <= Nv; j++)
  {
   if (i == j)
   {
    G[i][j] = 0;
   }
   else
    G[i][j] = INFINITY; // 初始化
  }
 }
 for (i = 0; i < Ne; i++) // 注意这里是读入
边
 {
  scanf("%d %d %d", &v1, &v2, &w);
  G[v1][v2] = w;           // 读入权值
  G[v2][v1] = G[v1][v2]; // 无向图对称
 }
}
void FindAnimal()
{
 int max, min;
 int animal;
 int i;
 Floyd();
 min = INFINITY;
 for (i = 1; i <= Nv; i++)
 {
  // 比较每行最大距离, 寻找其中最小值
  max = FindMax(i);
  if (max == INFINITY)
  {
   // 判断图是否连同通
   printf("0\n");
   return;
  }
  if (min > max)
  {
   min = max;
   animal = i;
  }
 }
 printf("%d %d\n", animal, min);
}
int FindMax(int i)
{
 int max;
 int j;
 max = 0;
 for (j = 1; j <= Nv; j++)
 {
  if (i != j && D[i][j] > max)
  {
   max = D[i][j];
  }
 }
 return max;
}
void Floyd()
{
 int i, j, k;
 for (i = 1; i <= Nv; i++)
 {
  for (j = 1; j <= Nv; j++)
  {
   D[i][j] = G[i][j];
  }
 }
 // 注意动物是从下标1开始编号
 for (k = 1; k <= Nv; k++)
 {
  for (i = 1; i <= Nv; i++)
  {
   for (j = 1; j <= Nv; j++)
   {
    if (D[i][k] + D[k][j] < D[i][j])
    {
     D[i][j] = D[i][k] + D[k][j];
    }
   }
  }
 }
}
```

## 7-9 旅游规划

```
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <climits>
using namespace std;
#define N 505
int MAX = INT_MAX;
int graph[N][N], cost[N][N];
int dist[N], vis[N], mincost[N];
void dijkstra(int, int, int);
int main()
{
 int n, m, s, d;
 cin >> n >> m >> s >> d;
 // 初始化
 for (int i = 0; i < n; i++)
 {
  for (int j = 0; j < n; j++)
  {
   graph[i][j] = graph[j][i] = MAX;
   cost[i][j] = cost[j][i] = MAX;
  }
 }
 memset(vis, 0, sizeof(vis));
 for (int i = 0; i < m; i++)
 {
  int from, to, len, c;
  cin >> from >> to >> len >> c;
  graph[from][to] = graph[to][from] =
len;
  cost[from][to] = cost[to][from] = c;
 }
 // 以s作为起点,初始化dist数组
 for (int i = 0; i < n; i++)
 {
  dist[i] = graph[s][i];
  mincost[i] = cost[s][i];
 }
 dijkstra(s, n, d);
 cout << dist[d] << " " << mincost[d];
 return 0;
}
void dijkstra(int s, int n, int d)
{
 vis[s] = 1;
 dist[s] = 0;
 for (int i = 0; i < n - 1; i++)
```

```
  {
    int minn = MAX, minindex = -1;
    for (int j = 0; j < n; j++)
    {
      if (vis[j] == 0 && dist[j] < minn)
      {
        minn = dist[j];
        minindex = j;
      }
    }
    vis[minindex] = 1;
    for (int j = 0; j < n; j++)
    {
      if (vis[j] == 0 && graph[minindex][j]
< MAX && dist[minindex] +
graph[minindex][j] < dist[j])
      {
        dist[j] = dist[minindex] +
graph[minindex][j];
        mincost[j] = mincost[minindex] +
cost[minindex][j];
      }
      else if (vis[j] == 0 &&
graph[minindex][j] < MAX && dist[minindex]
+ graph[minindex][j] == dist[j] &&
mincost[j] > mincost[minindex] +
cost[minindex][j])
        mincost[j] = mincost[minindex] +
cost[minindex][j];
    }
  }
}
```

## 7-10 公路村村通

```
#include <iostream>
#include <string.h>
#include <climits>
using namespace std;
#define N 1005
const int INF = INT_MAX;
int g[N][N];    //邻接矩阵
bool visited[N];    //顶点是否已经进入S集合
int lowcost[N];     //从集合S到未被选中集合
的最小权值
int n, m;
int Prim();
int main(){
    cin >> n >> m;
    //初始化
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            g[i][j] = g[j][i] = INF;
        }
    }
    for(int i = 0; i < m; i++){
        int from, to, cost;
        cin >> from >> to >> cost;
        g[from][to] = g[to][from] = cost;
    }
    cout << Prim() << endl;
    return 0;
}
int Prim(){
  for(int i = 1; i <= n; i++){
    lowcost[i] = INF;
    visited[i] = false;
  }
  //不要忘记初始化
  for(int i = 2; i <= n; i++){
        if(g[1][i]!=INF) lowcost[i] =
g[1][i];
  }
    visited[1] = true;
    lowcost[1] = 0;
   /* for(int i = 1; i <= n; i++){
      cout << lowcost[i] <<" ";
    }*/
    int res = 0;
    for(int k = 1; k <= n-1; k++){//循环n-1
次
        int v = -1, minn = INF;
        for(int i = 1; i <= n; i++){
            if(!visited[i] && lowcost[i] <
minn){
                v = i;
                minn = lowcost[i];
            }
        }
        if(v == -1) return -1;
        visited[v] = true;
        res += lowcost[v];
      // cout << "v:" << v <<" " <<
lowcost[v] << endl;
        for(int i = 2; i <= n; i++){
            lowcost[i] = min(lowcost[i],
g[v][i]);
        }
    }
    return res;
}
```

## 7-11 关键活动

```
#include <stdio.h>
#include <stdlib.h>
#define MAXVER 105
#define INFINITY 65535
int G[MAXVER][MAXVER]; // 图
int early[MAXVER];     // 最早发生时间
int late[MAXVER];      // 最迟发生时间
int in[MAXVER];        // 入度
int out[MAXVER];       // 出度
int nv, ne;        // 顶点数目 , 边数目
void CreatGraph();
int EarlyTime();
void LateTime(int Scost);
int FindMax(int a, int b);
int FindMin(int a, int b);
int main()
{
  int flag;
  int i, j;
  scanf("%d %d", &nv, &ne);
  CreatGraph();
  flag = EarlyTime();
  if (flag == -1)
  {
    printf("0\n");
  }
  else
  {
    printf("%d\n", flag);
    LateTime(flag);
    for (i = 1; i <= nv; i++)
    {
      if (early[i] != late[i])
        continue;
      for (j = nv; j >= 1; j--)
      {
        if (G[i][j] >= 0 && early[j] ==
late[j] && late[j] - G[i][j] == early[i])
        {
          // i,j 均在关键路径上且相邻
          printf("%d->%d\n", i, j);
        }
      }
    }
  }
  return 0;
}
void CreatGraph()
{
  int i, j;
  int s, d, cost;
  for (i = 1; i <= nv; i++)
  {
    for (j = 1; j <= nv; j++)
    {
      G[i][j] = -1;
    }
    early[i] = 0;
    late[i] = INFINITY;
    in[i] = 0;
    out[i] = 0;
  }
  for (i = 0; i < ne; i++)
  {
    scanf("%d %d %d", &s, &d, &cost);
    G[s][d] = cost; // 有向边
    in[d]++;
    out[s]++;
  }
}
int EarlyTime()
{
  int queue[nv];
  int first = -1, rear = -1;
  int count = 0;
  int i;
  int temp, ret = 0;
  for (i = 1; i <= nv; i++)
  {
    if (in[i] == 0)
    {
      // 如果入度为0 则入队
      queue[++rear] = i;
    }
  }
  while (first < rear) // 判断队是否为空
  {
    temp = queue[++first]; // 出队
    count++;
    for (i = 1; i <= nv; i++)
    {
      if (G[temp][i] >= 0)
      {
        in[i]--;
        early[i] = FindMax(early[i],
early[temp] + G[temp][i]);
        if (in[i] == 0)
        {
          queue[++rear] = i;
        }
      }
    }
  }
  if (count != nv)
  {
    ret = -1;
  }
  else
  {
    ret = early[1];
    for (i = 2; i <= nv; i++)
    {
      if (early[i] > ret)
      {
        // 找出最大的early[i]
        ret = early[i];
      }
    }
  }
  return ret;
}
void LateTime(int Scost)
{
  int i;
  int queue[MAXVER];
  int first = -1, rear = -1;
  int temp;
  for (i = 1; i <= nv; i++)
  {
    if (out[i] == 0)
    {
      queue[++rear] = i;
      late[i] = Scost;
    }
  }
  while (first < rear)
  {
    temp = queue[++first];
    for (i = nv; i >= 1; i--)
    {
      if (G[i][temp] >= 0)
      {
        late[i] = FindMin(late[i],
late[temp] - G[i][temp]);
        out[i]--;
        if (out[i] == 0)
        {
          queue[++rear] = i;
        }
      }
    }
  }
}
int FindMax(int a, int b)
{
  return a > b ? a : b;
}
int FindMin(int a, int b)
{
  return a > b ? b : a;
}
```

## 7-12 排序

```
#include <iostream>
#include <stdio.h>
using namespace std;
void print(int *a, int n);
void insert_sort(int *a, int n);
void bin_insertsort(int *a, int n);
void shell_sort(int *a, int n);
void bubble_sort(int *a, int n);
void quick_sort(int *a, int left, int
right);
int Partition(int *a, int left, int
right);
/*堆排序*/
void sift(int *a, int low, int high);
void heap_sort(int *a, int n);
int main(){
    int n, a[100005];
    scanf("%d", &n);
    for(int i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    heap_sort(a, n);
    print(a, n);
    return 0;
}
void heap_sort(int *a, int n){
    for(int i=(n-1)/2; i>=0; i--){
        sift(a, i, n-1);
    }
    for(int i=n-1; i>=1;){
        swap(a[i], a[0]);
        i--;
        sift(a, 0, i);
    }
}
void sift(int *a, int low, int high){
    int i = low, j = 2*i+1;
    int temp = a[i];
    while(j <= high){
        if(j < high && a[j]<a[j+1]){
            j++;
        }
        if(temp < a[j]){
            a[i] = a[j];
            i = j;
            j = 2*i+1;
        }else{
            break;
        }
    }
    a[i] = temp;
}

void quick_sort(int *a, int left, int
right) {
    if (left < right) {
        int pivot_index = Partition(a,
left, right); // 返回一次划分后基准的位置
        quick_sort(a, left, pivot_index -
1); // 对划分后的左边排序
        quick_sort(a, pivot_index + 1,
right); // 对划分后的右边排序
    }
    return;
}
int Partition(int *a, int left, int right)
{
    int temp = a[left]; // 最左边元素作为基
准
    while (left < right) {
        while (left < right && a[right] >=
temp) {
            right--;
        }
        a[left] = a[right];
        while (left < right && a[left] <=
temp) {
            left++;
        }
        a[right] = a[left];
    }
    a[left] = temp; // 基准归位
    return left; // 返回基准位置
}
void bubble_sort(int *a, int n){
    int flag = 0;
    for(int i = 0; i < n-1; i++){//进行n-1
趟排序
        for(int j = n-1; j > i; j--){
            if (a[j] < a[j - 1]) {
                int temp = a[j];
                a[j] = a[j - 1];
                a[j - 1] = temp;
                flag = 1;
            }
        }
        if(!flag) break;
    }
}
void shell_sort(int *a, int n){
    for(int d=n/2; d>=1; d/=2){
        for(int i = d; i < n; i++){ // 0, 0
+ dk, 0 + 2*dk...
            int temp = a[i];
            if(a[i-d] > temp){
                int j = i-d;
                for(; j>=0 && a[j]>temp; j-
=d){
                    a[j+d] = a[j];
                }
                a[j+d] = temp;
            }
        }
    }
}
void bin_insertsort(int *a, int n){
    for(int i = 1; i < n; i++){
        int left = 0, right = i-1, temp =
a[i], mid;
        while(left <= right){
            mid = (right + left)/2;
            if(temp < a[mid]){
                right = mid-1;
            }else{
                left = mid+1;
            }
        }
        //此处a[left] > temp,将a[left~i-1]
后移一位
        for(int j = i-1; j>=left; j--){
            a[j+1] = a[j];
        }
        a[left] = temp;
    }
}
//插入排序
void insert_sort(int *a, int n){
    //初始0~0作为有序区，之后有序区为0~i，逐
渐扩大
    for(int i = 1; i < n; i++){
        if(a[i] < a[i-1]){
            int temp = a[i];
            int j = i - 1;
            for(; j>=0 && temp < a[j]; j-
-){
                a[j+1] = a[j];
            }
            a[j+1] = temp;
        }
    }
}
void print(int *a, int n){
    int flag = 0;
    for(int i = 0; i < n; i++){
        if(flag){
            printf(" ");
        }else{
            flag = 1;
        }
        printf("%d", a[i]);
    }
    //printf("\n");
}
```

## 7-13 统计工龄

```
#include <iostream>
#include <map>
using namespace std;
int main(){
    map<int, int> m;
    int n;
    cin >> n;
    for(int i = 0; i < n; i++){
        int t;
        cin >> t;
        m[t]++;
    }
    for(map<int,int>::iterator it =
m.begin(); it!=m.end(); ++it){
        cout << it->first <<":" <<
it->second << endl;
    }
    return 0;
}
```

## 7-14 电话聊天狂人

```
#include <iostream>
#include <map>
using namespace std;
int main(){
    map<string, int> m;
    int n, count = 0;
    cin >> n;
    for(int i = 0; i < n; i++){
        string a, b;
        cin >> a >> b;
        m[a]++;
        m[b]++;
    }
    int maxx = -1;
    string ans;
    for(map<string,int>::iterator it =
m.begin(); it!=m.end(); ++it){
        if(it->second > maxx){
            count = 1;
            maxx = it->second;
            ans = it->first;
        }else if(it->second == maxx){
            count++;
            if(ans.compare(it->first) > 0)
 //当前遇到的电话比 ans 小（字典序），就返回小
于 0 的
                ans = it->first;
        }
    }
    cout << ans << " " << maxx;
    if(count!=1) cout <<" " <<count;
    return 0;
}
```

## 7-15 QQ 帐户的申请与登陆

```
#include <iostream>
#include <map>
using namespace std;
int main()
{
  map<string, string> m;
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    char op;
    string a, b;
    cin >> op >> a >> b;
    if (op == 'L')
    { // 老帐户登陆
      if (m.count(a) <= 0)
      { // 老帐户 QQ 号码不存在
        cout << "ERROR: Not Exist" << endl;
      }
      else
      { // 老帐户登陆成功
        if (b.compare(m[a]) == 0)
        {
          cout << "Login: OK" << endl;
        }
        else
        { // 老帐户密码错误
          cout << "ERROR: Wrong PW" << endl;
        }
      }
    }
    else if (op == 'N')
    { // 申请账户
      if (m.count(a) > 0)
      {
        // 若新申请的号码已经存在，则输出
“ERROR: Exist”;
        cout << "ERROR: Exist" << endl;
      }
      else
      {
        // 新申请帐户成功
        cout << "New: OK" << endl;
        m[a] = b;
      }
    }
  }
  return 0;
}
```

## 7-16 一元多项式求导

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int coff, exp;
    int flag = 0;
    while(scanf("%d %d",&coff,&exp)!=EOF){
        if(exp){
            if( flag )
                printf(" ");  //如果不是第一
个数就先输出一个空格
            else
                flag = 1;
            printf("%d %d",coff*exp,exp-1);
        }
        else break;
    }
    if(!flag){
        printf("0 0");
    }
    return 0;
}
```

## 7-17 汉诺塔的非递归实现

```
#include<iostream>
#include <stack>
char s[4] = { 'q','a','b','c' };
std::stack<int> a[4];
bool move(int before, int after) {
  if (a[before].empty())
    return false;
  if (!a[after].empty())
    if ((a[after].top() - a[before].top())
< 0)
      return false;
  a[after].push(a[before].top());
  a[before].pop();
  printf("%c -> %c\n", s[before],
s[after]);//faster than cout
  return true;
}
int main() {
  int  N, count = 0;
  std::cin >> N;
  for (int i = 0; i < N; i++)
    a[1].push(N - i);
  if (N % 2 == 1) {
    s[2] = 'c'; s[3] = 'b';
  }
  while (++count) {
    move((count - 1) % 3 + 1, (count) % 3 +
1);
    if (!move((count - 1) % 3 + 1, (count +
1) % 3 + 1)&&!move((count + 1) % 3 + 1,
(count - 1) % 3 + 1))
        break;
  }
}
```

## 7-18 银行业务队列简单模拟

```
#include <iostream>
#include <queue>
using namespace std;
int main(void)
{
  int clientCount;
  cin >> clientCount;
  queue<int> line1;
  queue<int> line2;
  int buf;
  for (int i = 0; i < clientCount; i++)
  {
    cin >> buf;
    if (buf & 1)
    {
      line1.push(buf);
    }
    else
    {
      line2.push(buf);
    }
  }
  int output[1010];
  int outputCount = 0;
  while (line1.size() >= 2
&& !line2.empty())
  {
    output[outputCount++] = line1.front();
    line1.pop();
    output[outputCount++] = line1.front();
    line1.pop();
    output[outputCount++] = line2.front();
    line2.pop();
  }
  while (!line1.empty())
  {
    output[outputCount++] = line1.front();
    line1.pop();
  }
  while (!line2.empty())
  {
    output[outputCount++] = line2.front();
    line2.pop();
  }
  for (int i = 0; i < outputCount; i++)
  {
    printf(" %d" + !i, output[i]);
  }


  return 0;
}
```

## 7-19 求链式线性表的倒数第 K 项

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
  vector<int> a;
  int n, x;
  scanf("%d", &n);
  while(1){
        scanf("%d", &x);
        if(x < 0) break;
        a.push_back(x);
    }
  int count = a.size();
  if(n>0 && n<=count) cout<<a[count-
n]<<endl;
  else cout<<"NULL";
  return 0;
}
```

## 7-20 表达式转换

```
/*
中缀表达式转后缀表达式的方法：
1.遇到操作数：直接输出（添加到后缀表达式中）
2.栈为空时，遇到运算符，直接入栈
3.遇到左括号：将其入栈
4.遇到右括号：执行出栈操作，并将出栈的元素输
出，直到弹出栈的是左括号，左括号不输出。
5.遇到其他运算符：加减乘除：弹出所有优先级大于
或者等于该运算符的栈顶元素，然后将该运算符入栈
6.最终将栈中的元素依次出栈，输出。
*/
#include<bits/stdc++.h>
using namespace std;
int main(){
  stack<char>s;
  map<char,int>m;//设置符号间的优先级

  m['+'] = 1; m['-'] = 1;
  m['*'] = 2; m['/'] = 2;
  m['('] = 3; m[')'] = 3;
  int flag = 0;
  string str;
  cin >> str;

  for( int i = 0; i < str.size(); i++ )
  {
    //判断代码当中是否有数字 或则小数点 或则是
这个数带符号（负号）
    if( ( ((i == 0) || str[i - 1] == '(')
&& (str[i] == '+' || str[i] == '-'))// 如果
是负数则 eg (-10)  其中 i = 0 考虑到 第一个数
如果是 eg +10
        || ( str[i] >='0' && str[i] <= '9')
        || ( str[i] == '.' )
       )
    {
      if(flag != 0 )
        cout << ' ';

      if( str[i] != '+')
        cout << str[i];//此处是如果输出的是负
号 则输出 如果是正号则不输出
      //这是要输出的数字eg 5.5
      while( (str[i+1] == '.') || (str[i +
1] >= '0' && str[i + 1] <= '9'))
      {
        i++;
        cout << str[i];
      }
      flag = 1;
    }
    else
    {
      if(str[i] == ')')
      while(!s.empty() && s.top() != '(')
      {
          cout << ' ' << s.top();
          s.pop();
      }
      s.pop();//将栈中的'(' 删除

      else if(s.empty() || m[str[i]] >
m[s.top()])
      {
        s.push(str[i]);
      }
      else
      {
        //将优先级小于str[i] 输出去 但没遇到
')' 所以 '(' 不用输出
        while( !s.empty() && s.top() !=
'(')
        {
          cout << ' ' << s.top();
          s.pop();
        }
        s.push(str[i]);
        }
      }
    }
    //将栈中剩余的符号输出来

    while(!s.empty())
    {
      cout << ' ' << s.top();
      s.pop();
    }
}
```

## 7-21 求前缀表达式的值

```
#include<bits/stdc++.h>
using namespace std;
string a[100];
int main()
{
  stack<double> s;
  char c;
  double b,d;
  int i = 0;
  int flag = 0;//标记是否错误
  while(1)
  {
    cin>>a[i++];
    c = getchar();
    if(c == '\n')
      break;
  }
  i--;
  for(int j = i;j >= 0;j--)
  {
    if(a[j] == "+" || a[j] == "-" || a[j]
== "*" || a[j] == "/")
    {
      if(s.size() < 2)//第一个错误点：当出现
运算符时，栈元素不足与用来运算。
      {
        flag = 1;
        break;
      }
      b = s.top();
      s.pop();
      d = s.top();
      s.pop();
      if(a[j] == "+") s.push(b+d);
      else if(a[j] == "-") s.push(b-d);
      else if(a[j] == "*") s.push(d*b);
      else if(a[j] == "/")
      {
        if(d == 0)//第二个坑点：除数为0.
        {
          flag = 1;
          break;
        }
        s.push(b/d);
      }
    }
    else
    {
      stringstream ss;//用string流将string
转换成double。
      ss<<a[j];
      ss>>b;
      s.push(b);
    }
  }
  if(s.size()!=1)//第三个坑点：最后输出的时候
栈里元素个数多于1。
    flag = 1;
  if(flag)
    cout<<"ERROR";
  else
    printf("%.1f",s.top());
  return 0;
}
```

## 7-22 堆栈模拟队列

```
#include <cstdio>
#include <iostream>
using namespace std;
int n1, n2, d;
char s[2];
int s1[1000], s2[1000], c1, c2;
int main()
{
  scanf("%d%d", &n1, &n2);
  if (n1 > n2)
    swap(n1, n2);
  while (scanf("%s", s) && s[0] != 'T')
  {
    if (s[0] == 'A')
    {
      scanf("%d", &d);
      if (c1 == n1)
      {
        printf("ERROR:Full\n");
      }
      else
        s1[c1++] = d;
    }
    else
    {
      if (c2)
        printf("%d\n", s2[--c2]);
      else if (c1)
      {
        while (c1)
          s2[c2++] = s1[--c1];
        printf("%d\n", s2[--c2]);
      }
      else
        printf("ERROR:Empty\n");
    }
    if (!c2 && c1 == n1)
    {
      while (c1)
        s2[c2++] = s1[--c1];
    }
  }
}
```

## 7-23 还原二叉树

```
#include <iostream>
#include <stdio.h>
#include <queue>
using namespace std;
int depth(char *a, char *b, int len){
    if(len==0) return 0;
    int i;
    //先序序列中遍历顺序:根左右,对应中序序列的
节点左边是左子树，右边是右子树
    for(i = 0; i < len; i++){
        if(b[i] == a[0]){
            break;
        }
    }
    //在b[0]~b[i-1]搜索左子树,
    int x = depth(a+1, b, i) + 1;
    //在b[i+1]~b[n-i]搜索右子树
    int y = depth(a+i+1, b+i+1, len-i-1) +
1;
    return x > y ? x : y;
}
int main(){
    char a[52]; //first_order
    char b[52]; //in_order
    int n;
    cin >> n;
    cin >> a >> b;
    cout << depth(a, b, n);
    return 0;
}
```

## 7-24 树种统计

```
#include <iostream>
#include <string>
#include <map>
#include <stdio.h>
using namespace std;
int main(){
  map<string, int> m;
  int n;
  cin>>n;
  getchar();  //这个坑踩了好久!!::当有string
类型的输入前面有其他类型输入时，用getchar()吃
回车
  for(int i = 0; i < n; i++){
      string s;
      getline(cin, s);
      //cin 不接受空格，TAB 等键的输入，遇到
这些键，字符串会终止
    //char[]可用gets(),string 类型的只能
用getline(cin,s)
    m[s]++;
 }
 for(map<string,int>::iterator it =
m.begin(); it!=m.end(); it++){
    cout<<it->first<<" ";
    printf("%.4f%%\n",
it->second/(double)n*100);
 }
 return 0;
}
```

## 7-25 朋友圈

```
#include <iostream>
#include <stdio.h>
using namespace std;
const int MAXN = 30005;
int f[MAXN], res[MAXN];
int Find(int x){
  if(f[x]!=x)
    f[x] = Find(f[x]);
  return f[x];
}
void Union(int x, int y){
  int p1 = Find(x);
  int p2 = Find(y);
  f[p1] = p2;
}
int main(){
  int n, m;
  cin >> n >> m;
  for(int i = 1; i <= n; i++){
    f[i] = i;
  }
  for(int i = 0; i < m; i++){
    int x, root;
    scanf("%d%d", &x, &root);
    root = Find(root);
    for(int k = 0; k < x-1; k++){
      int t;
      scanf("%d", &t);
      Union(root, t);
    }
  }
  int ans = 0;
  for(int i = 1; i <= n; i++){
    int root = Find(i);
    res[root]++;
    ans = max(ans, res[root]);
  }
  cout << ans;
  return 0;
}
```

## 7-26 Windows 消息队列

```
#include <iostream>
#include <queue>
using namespace std;
class QueueItem
{
public:
 char name[12];
 int priority;
 friend bool operator<(QueueItem v1,
QueueItem v2)
 {
  return v2.priority < v1.priority;
 }
};
int main()
{
 std::ios::sync_with_stdio(false);
 int msgCount;
 cin >> msgCount;
 priority_queue<QueueItem> msgQueue;
 string opBuf, nameBuf, priBuf;
 for (int i = 0; i < msgCount; i++)
 {
  cin >> opBuf;
  if (opBuf == "PUT")
  {
   QueueItem *newItem = new QueueItem;
   cin >> newItem->name >>
newItem->priority;
   msgQueue.push(*newItem);
  }
  else if (opBuf == "GET")
  {
   if (!msgQueue.empty())
   {
    cout << msgQueue.top().name <<
endl;
    msgQueue.pop();
   }
   else
   {
    cout << "EMPTY QUEUE!" << endl;
   }
  }
  else
  {
   cout << "Wrong Input" << endl;
   return -1;
  }
 }
 return 0;
}
```

## 7-27 家谱处理

```
#include<cstdio>
#include<iostream>
#include<cstring>
#include<algorithm>
#include<queue>
#include<vector>
#include<map>
using namespace std;
const int inf=0x3f3f3f3f;
typedef long long ll;
#define N 1005
map<string,int>mp;
string a[10];
struct node{
 int parent,b;
 vector<int>v;
}s[N];
int main(){
  int n,m,i,j,x,y,sum=0,cnt=0;
  string str;
 scanf("%d %d",&n,&m);
 cin>>str;
 mp[str]=0;
 s[0].b=0;
 s[0].parent=-1;
 for(i=1;i<n;i++){
  char ch;
  int c=0;
  if(i==1)getchar();
  while((ch=getchar())==' ') c++;
  str=ch;
  while((ch=getchar())!='\n') str=str+ch;

  mp[str]=i;
  x=-1;
  for(j=0;j<i;j++){
   if(s[j].b==c-2){
    x=j;
   }
  }
  s[i].b=c;
  s[i].parent=x;
  s[x].v.push_back(i);
 }
 while(m--){
  for(i=1;i<=6;i++){
   cin>>a[i];
  }
  int flag=0;
  if(a[4]=="child"){
   int ch=mp[a[1]];
   int p=mp[a[6]];
   if(s[ch].parent==p) flag=1;
  }
  else if(a[4]=="ancestor"){
   int p=mp[a[1]];
   int ch=mp[a[6]];
   queue<int>q;
   if(p<ch)q.push(p);
   while(!q.empty()){
    int t=q.front();q.pop();
    if(t==ch) {
     flag=1;break;
    }
    for(i=0;i<s[t].v.size();i++){
     q.push(s[t].v[i]);
    }
   }
  }
  else if(a[4]=="sibling"){
   int x=mp[a[1]];
   int y=mp[a[6]];
   if(s[x].parent==s[y].parent&&x!=y)
flag=1;

  }
  else if(a[4]=="parent"){
   int p=mp[a[1]];
   int ch=mp[a[6]];
   if(s[ch].parent==p) flag=1;
  }
  else if(a[4]=="descendant"){
   int ch=mp[a[1]];
   int p=mp[a[6]];
   queue<int>q;
   if(ch>p)q.push(p);
   while(!q.empty()){
    int t=q.front();q.pop();
    if(t==ch) {
     flag=1;break;
    }
    for(i=0;i<s[t].v.size();i++){
     q.push(s[t].v[i]);
    }
   }
  }
  else continue;
  if(flag) puts("True");
  else puts("False");
 }
 return 0;
}
```

## 7-28 搜索树判断

```
#include<iostream>
#include<vector>
using namespace std;
const int N = 1010;
vector<int> a,pre,post;
struct Tree{
  int val;
  Tree *left,*right;
  Tree(int x){
    val = x;
    left = right = NULL;
  }
};
void build(Tree* &t,int x){
  if(!t){
        t = new Tree(x);
        return;
    }
    if(t->val <= x) build(t->right,x);
    else build(t->left,x);
}
void pre1(Tree* t){
    if(!t) return;
    pre.push_back(t->val);
    pre1(t->left);
    pre1(t->right);
    post.push_back(t->val);
}
void pre2(Tree* t){
    if(!t) return;
    pre.push_back(t->val);
    pre2(t->right);
    pre2(t->left);
    post.push_back(t->val);
}
int main(){
    int n;
    Tree *bt = NULL;
    cin>>n;
    for(int i = 1; i <= n; i++){
        int x;
        cin>>x;
        a.push_back(x);
        build(bt,x);//不管是否为镜像，先建立
搜索树
    }
    pre1(bt);//正搜索树先序遍历，并记录后序遍
历
    if(pre == a){
        cout<<"YES\n";
        for(int i = 0; i < post.size();
i++){
            if(i) cout<<' ';
            cout<<post[i];
        }
        return 0;
    }
    pre.clear(); post.clear();
    pre2(bt);//镜像搜索树先序遍历，并记录后序
遍历
    if(pre == a){
        cout<<"YES\n";
        for(int i = 0; i < post.size();
i++){
            if(i) cout<<' ';
            cout<<post[i];
        }
        return 0;
    }
    cout<<"NO";
    return 0;
}
```

## 7-29 修理牧场

```
#include <iostream>
#include <stdio.h>
#include <queue>
using namespace std;
int main(){
    int n;
    priority_queue<int, vector<int>,
greater<int>> pq;
    cin >> n;
    for(int i = 0; i < n; i++){
        int x;
        scanf("%d", &x);
        pq.push(x);
    }
    int ans = 0;
    while(pq.size()>1){
        int a = pq.top();
        pq.pop();
        int b = pq.top();
        pq.pop();
        int t = a + b;
        cout << t<<endl;
        ans += t;
        pq.push(t);
    }
    cout << ans;
    return 0;
}
```

## 7-30 目录树

```
#include<cstdio>
#include<algorithm>
#include<iostream>
#include<string>
#include<vector>
#include<set>
#include<map>
#define MAXN 100010
using namespace std;

struct node {
  string name;
  int isCata;                // 目录文件标记
  vector<node*> child;       // 孩子指针
};
bool cmp(node* a, node* b) {
  if(a->isCata != b->isCata)    return
a->isCata > b->isCata;
  else return a->name < b->name;
}
void dfs(node* root,int level) {
  if(root == NULL)   return ;
  // 先输出自己
  for(int i  = 0; i < level; ++i)
  printf("  ");
  printf("%s\n",root->name.c_str()) ;
  // 排序所有孩子 :  目录在前，文件在后，字典
升序
  sort(root->child.begin(),root->child.end(
),cmp);
  // 向下递归
  for(int i = 0; i < root->child.size();
++i)
    dfs(root->child[i],level+1);
}
int n;
int main() {
  scanf("%d",&n);
  getchar();
  // 建立根节点
  node* root = new node;
  root->name = "root";
  root->isCata = 1;

  string tmp,str;
  node* curRoot;
  for(int j = 0; j < n; ++j) {

     // 每一个新的路径，都将根设为 root
    curRoot = root;
    getline(cin,str);
    for(int i = 0; i <= str.size(); ++i) {
      if(str[i] == '\\') {// 情况 1. 是目
录 : 切换当前目录，
        // 在当前父目录中寻找，看是否存在
        int flag = 0;
        for(int k = 0; k <
curRoot->child.size(); ++k) {
        // 1.1 有该目录
          if(curRoot->child[k]->name == tmp
&& curRoot->child[k]->isCata == 1) {
            // 则切换当前目录
            curRoot =  curRoot->child[k];
            flag = 1;
            break;
          }
        }
        // 1.2 没有该目录则创建一个
        if(!flag) {
          // 创建结点
          node* newnode = new node;
          newnode->name = tmp;
          newnode->isCata = 1;
          // 加入父目录

  curRoot->child.push_back(newnode) ;
          // 切换当前目录
          curRoot = newnode;
        }
        // 单词清零
        tmp.clear();

      // 情况 2. 是文件
      }else if(i == str.size()) {
        if(!tmp.empty()) { // 到达最后，而单
词不空，说明是文件
          // 将文件加入到父节点中
          node* newnode = new node;
          newnode->name = tmp;
          newnode->isCata = 0;

  curRoot->child.push_back(newnode) ;
        }
        tmp.clear();
      } else {                   // 情况
3. 累加单词字母
        tmp += str[i];
      }
    }
  }
  // 输出过程
  dfs(root,0);
  return 0;
}
```

## 7-31 笛卡尔树

```
#include <iostream>
#include <stdio.h>
#include <queue>
#include <climits>
using namespace std;
const int INF = INT_MAX;
const int MIN = INT_MIN;
struct Node{
    int k1, k2, lchild, rchild;
}node[1005];
bool IsVaild(int i, int k1_min, int
k1_max){
    int lchild = node[i].lchild, rchild =
node[i].rchild;
    int k1 = node[i].k1, k2 = node[i].k2;
    if(lchild == -1 && rchild == -1)    //
空树返回真
        return true;
    if(lchild!=-1){         //左树不为空
        if(node[lchild].k1 >= k1 ||
node[lchild].k1 <= k1_min)  //左树不满足BST
性质
            return false;
        if(node[lchild].k2 <= k2)       //是
否满足最小堆的性质
            return false;
    }
    if(rchild!=-1){         //右树不为空
        if(node[rchild].k1 <= k1 ||
node[rchild].k1 >= k1_max)  //右树不满足BST
性质
            return false;
        if(node[rchild].k2 <= k2)   //是否满
足最小堆的性质
            return false;
    }
    bool flag1 = true, flag2 = true;
    if(lchild != -1){
        flag1 = IsVaild(lchild, k1_min, k1);
    }
    if(rchild != -1){
        flag2 = IsVaild(rchild, k1, k1_max);
    }
    return  flag1 && flag2;
}
int main(){
    int n;
    cin >> n;
    int root = -1;
    int vis[1005]={0};
    for(int i = 0; i < n; i++){
        cin >> node[i].k1 >> node[i].k2 >> node[i].lchild >> node[i].rchild;
    }
    //找根节点
    for(int i = 0; i < n; i++){
        vis[node[i].lchild] = 1;
        vis[node[i].rchild] = 1;
    }
    for(int i = 0; i < n; i++){
        if(!vis[i]) {
            root = i;
            break;
        }
    }

    if(IsVaild(root, MIN, INF)){
        cout << "YES";
    }else{
        cout << "NO";
    }
    return 0;
}
```

## 7-32 哥尼斯堡的“七桥问题”

```
#include <bits/stdc++.h>
using namespace std;
#define MAXN 1010
int a[MAXN][MAXN]={0},vis[MAXN]={0},cnt[MAXN]= {0};
int n, m, b, c;
void dfs(int cur){
    vis[cur] = 1;
    for(int i = 1; i <= n; i++){
        if(!vis[i] && a[cur][i]){
            dfs(i);
        }
    }
    return;
}
int main(){
    cin >> n >> m;
    for(int i = 0; i < m; i++){
        scanf("%d%d", &b, &c); //别用cin
        a[b][c]=a[c][b] = 1;
        cnt[b]++; //记录每一个顶点的度数
        cnt[c]++;
    }
    dfs(1);
    int f = 1;
    for(int i=1; i<=n; i++){
        if(!vis[i] || cnt[i]%2 == 1){//是否存在奇数度数的顶点
            f = 0;
            break;
        }
    }
    cout << f;
    return 0;
}
```

## 7-33 地下迷宫探索

```
#include <bits/stdc++.h>
using namespace std;
#define MAXN 1010
int a[MAXN][MAXN]= {0}, vis[MAXN]= {0};
int cnt = 1, n, m, s, f=0;
void dfs(int x){
    if(f)
        printf(" ");
    f++;
    printf("%d", x);
    for(int i = 1; i <= n;  i++){
        if(!vis[i]&&a[x][i]){
            vis[i] = 1;
            cnt++;
            dfs(i);
            printf(" %d", x);
        }
    }
}
int main()
{
    cin >> n >> m >> s;
    for(int i = 1; i <= m; i++){
        int b,c;
        scanf("%d%d", &b, &c);
        a[b][c] = a[c][b] = 1;
    }
    vis[s] = 1;
    dfs(s);
    if(cnt < n)
        cout<<" 0";
    return 0;
}
```

## 7-34 任务调度的合理性

```
#include <bits/stdc++.h>
using namespace std;
#define MAXN 105
int a[MAXN] = {0}, g[MAXN][MAXN];//a[i]为节点i的入度
int main(){
    int m, n, t;
    cin >> n;
    for(int i = 1; i <= n; i++){
        cin >> m;
        for(int j = 1; j <= m; j++){
            cin >> t;
            g[t][i] = 1;
            a[i]++;
        }
    }
    queue<int> q;
    for(int i = 1; i <= n; i++){
        if(a[i]==0) q.push(i);
    }
    while(q.size() > 0){
        int cur = q.front();
        q.pop();
        for(int i = 1; i <= n; i++){
            if(g[cur][i]!=0){
                a[i]--;
                if(a[i] == 0){
                    q.push(i);
                }
            }
        }
    }
    int f = 1;
    for(int i = 1; i <= n; i++){
        if(a[i]!=0){    //经过拓扑排序后还有入度为0的点，说明有向图有回路了
            f = 0;
            break;
        }
    }
    cout << f;
}
```

## 7-35 城市间紧急救援

```
/**
 * 在进行Dijkstra算法判断距离时，距离变短：不管是什么信息（path，d，pain，num）
 * ，必定会强制更新；
 * 但是，但距离相同时，得考虑能否让花费(cost数组)，收集资源（pain数组）更优，
 * 如果更优，则更新，否则不予执行。
 */
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;
struct Node
{
 int v, w;
};
const int maxn = 510, INF = 1e9;
vector<Node> Adj[maxn]; // 邻接表
int c[maxn];    // 每个顶点的人数
int num[maxn];          // 最短路径条数
int d[maxn];    // 每个点到源点的最短距离
int pain[maxn];         // 最短路径上顶点的最大收获量
int path[maxn];         // 路径数组
bool hs[maxn];          // 顶点是否被选择
int Nv, Ne, st, ed;     // 顶点数，边数，起点，终点
int flag = 1;       // 是否输出空格
void Read()
{
 cin >> Nv >> Ne >> st >> ed;
 for (int i = 0; i < Nv; ++i)
  cin >> c[i];
 for (int i = 0; i < Ne; ++i)
 {
  int u, v, w;
  cin >> u >> v >> w;
  Adj[u].push_back({v, w}); // 无向边
  Adj[v].push_back({u, w});
 }
}
void Dijkstra()
{
 fill(d, d + maxn, INF); // 先将d数组初始化无穷大
 d[st] = 0;        // 源点到源点的距离为0;
 num[st] = 1;      // 起初存在一条最短路径
 pain[st] = c[st];     // 起点的权值直接可以收走
 for (int i = 0; i < Nv; ++i)
 {
  int u = -1, MIN = INF;
  for (int j = 0; j < Nv; ++j)
  {
   if (hs[j] == 0 && d[j] < MIN)
   {
    u = j;
    MIN = d[j];
   }
  }
  if (u == -1)
   return;
  hs[u] = 1;
  for (int j = 0; j < Adj[u].size(); ++j)
  {
   int v = Adj[u][j].v, w = Adj[u][j].w;
   if (d[u] + w < d[v]) // 如果路径更短，以下信息强制更新
   {
        d[v] = d[u] + w;
        num[v] = num[u];
        pain[v] = pain[u] + c[v];
        path[v] = u;
      }
      else if (d[u] + w == d[v])
// 路径长等相等，除了更新最短路径条数
      {                      // 还要判断该条
路径上的收获量是否更大，
        num[v] += num[u];              // 如
果是，则更新收获量，除此之外，
        if (pain[v] < pain[u] + c[v]) // 还
要更新 v 的前驱节点
        {
          pain[v] = pain[u] + c[v];
          path[v] = u;
        }
      }
    }
  }
}
void Print(int u)
{
  if (path[u] != -1)
    Print(path[u]);
  if (flag)
    flag = 0;
  else
    cout << ' ';
  cout << u;
}
int main()
{
  fill(path, path + maxn, -1);
  Read();
  Dijkstra();
  cout << num[ed] << ' ' << pain[ed] <<
endl;
  Print(ed);
  return 0;
}
```

## 7-36 社交网络图中结点的“重要性”计算

```
#include <cstdio>
#include <iostream>
#include <queue>
#include <algorithm>
using namespace std;
#define maxn 10005
double bfs(int x, int n);
vector<int> adj[maxn];  //用 vector 存边
int vis[maxn];
int main(){
  int n, m, k;
  cin >> n >> m;
  for(int i = 0; i < m; i++){
        int a, b;
        scanf("%d%d", &a, &b);
        adj[a].push_back(b);
        adj[b].push_back(a);    //别忘了是个
无向图
  }
  cin >> k;
  for(int i = 0; i < k; i++){
        int target;
        scanf("%d", &target);
        printf("Cc(%d)=%.2f\n", target,
bfs(target, n));
  }
  return 0;
}
double bfs(int x, int n){
    for(int i = 1; i <= n; i++){
        vis[i] = -1;
    }
    vis[x] = 0; //vis[i]表示点 x 到 i 的最小距
离
    double sum = 0;
    int cnt = 1;
    queue<int> q;
    q.push(x);
    while(!q.empty()){
        int cur = q.front();
        int len = adj[cur].size();
        for(int i = 0; i < len; i++){
            int next = adj[cur][i];
            if(vis[next] == -1){
                vis[next] = vis[cur] + 1;
                sum += vis[next];
                cnt++;
                q.push(next);
            }
        }
        q.pop();
    }
    if(cnt < n)
        return 0;
   // cout <<"sum:"<< sum << endl;
    double res = (n-1)/sum;
    return res;
}
```

## 7-37 模拟 EXCEL 排序

```
#include <bits/stdc++.h>
using namespace std;
struct student{
    char sno[7];
    char name[10];
    int score;
}s[100005];
bool cmp1(student a, student b){
    if(strcmp(a.sno, b.sno)>0){
        return b.sno < a.sno;
    }
    return b.sno > a.sno;
}
bool cmp2(student a, student b){
    if(strcmp(a.name, b.name)>0){
        return b.name < a.name;
    }
    return b.name > a.name;
}
bool cmp3(student a, student b){
    if(a.score == b.score){
        if(strcmp(a.sno, b.sno)>0){
            return b.sno < a.sno;
        }
        return b.sno > a.sno;
    }
    return a.score < b.score;
}
int main(){
    int n, c;
    cin >> n >> c;
    for(int i = 0; i < n; i++){
        scanf("%s%s%d", s[i].sno,
s[i].name, &s[i].score);
    }
    if(c == 1){
        sort(s, s+n, cmp1);
        for(int i = 0; i < n; i++){
            printf("%s %s %d\n", s[i].sno,
s[i].name, s[i].score);
        }
    }else if(c == 2){
        sort(s, s+n, cmp2);
        for(int i = 0; i < n; i++){
            printf("%s %s %d\n", s[i].sno,
s[i].name, s[i].score);
        }
    }else if(c == 3){
        sort(s, s+n, cmp3);
        for(int i = 0; i < n; i++){
            printf("%s %s %d\n", s[i].sno,
s[i].name, s[i].score);
        }
    }
    return 0;
}
```

## 7-38 寻找大富翁

```
#include <iostream>
#include <stdio.h>
#include <queue>
using namespace std;
int main(){
    int n, k;
    priority_queue<int, vector<int>,
less<int>> pq;//less 规定优先队列的顺序为从大
到小排列，即队头为最大的
    cin >> n >> k;
    for(int i = 0; i < n; i++){
        int x;
        scanf("%d", &x);
        pq.push(x);
    }
    int flag = 0;
    while(k-- && pq.size()>0){
        if(flag){
            cout << " ";
        }else{
            flag = 1;
        }
        int t = pq.top();
        pq.pop();
        cout <<t;
    }
    return 0;
}
```

## 7-39 魔法优惠券

```
#include <iostream>
#include <algorithm>
#include <cstdio>
using namespace std;
#define maxn 1000005
int main(){
    int a[maxn], b[maxn];
    int n;
    cin >> n;
    for(int i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    cin >> n;
    for(int i = 0; i < n; i++){
        scanf("%d", &b[i]);
    }
    sort(a, a+n);
    sort(b, b+n);
    int sum = 0;
    int i;
    for(i = n-1; i>=0; i--){   //正*正 = 正
        if(a[i]>0 && b[i]>0){
            sum += a[i]*b[i];
        }else{
            break;
        }
    }
    for(int j = 0; j < i; j++){     //负*负
= 负
        if(a[j]<0 && b[j]<0){
            sum += a[j]*b[j];
        }else{
            break;
        }
    }
    cout << sum;
    return 0;
}
```

## 7-40 奥运排行榜

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
typedef pair<double, int> PII;
bool cmp(PII a, PII b)
{
  return a > b;
}
int main()
{
  int n, m;
  cin >> n >> m;
  vector<vector<PII>> v(5);
  for (int i = 0; i < n; i++)
  {
    double a, b, c;
    cin >> a >> b >> c;
    v[1].push_back({a, i});
    v[2].push_back({b, i});
    v[3].push_back({a / c, i});
    v[4].push_back({b / c, i});
  }
  for (int i = 1; i <= 4; i++)
    sort(v[i].begin(), v[i].end(), cmp);
  bool flag = 0;
  for (int i = 1; i <= m; i++)
  {
    int t, rank = n * 2, r;
    cin >> t;
    for (int j = 1; j <= 4; j++)
      for (int k = 0; k < v[j].size(); k++)
        if (v[j][k].second == t)
          for (int l = 0; l < v[j].size();
l++)
            if (v[j][l].first ==
v[j][k].first)
              if (l + 1 < rank)
              {
                rank = l + 1;
                r = j;
              }
    if (!flag)
      flag = 1;
    else
      cout << ' ';
    printf("%d:%d", rank, r);
  }
  return 0;
}
```

## 7-41 PAT 排名汇总

```
/**
  利用结构体数组, 重写送 sort 方法
*/
#include<bits/stdc++.h>
using namespace std;
struct Node{
  string id;
  int grate;
  int examranking;//考试排名
  int examsite;//考点
  int siteranking;//考点排名
};
//这是定义一个递减的 sort
bool sort_Grate(Node a,Node b){
  if( a.grate == b.grate )
    return a.id < b.id;

  return a.grate > b.grate;
}
int main(){

  int N;
  int sum = 0;
  Node *stu = new Node[30005];
  cin >> N;

  for( int i = 0; i < N; i++ ){

    int K;
    cin >> K;
    for( int j = sum; j < sum + K; j++ ){
      cin >> stu[j].id >> stu[j].grate;
      stu[j]. examsite = i + 1;//记录考点

    }

    sort(stu+sum,stu+sum+K,sort_Grate);//这
是处理每个考点内的排名
    int count = 1;//记录排名

    for( int j = sum; j < sum + K; j++ ){
      if( j == sum){//处理第一个为排名第一的
        stu[j].siteranking = count;
      }else{
        if( stu[j].grate == stu[j -
1].grate)
          stu[j].siteranking = stu[j -
1].siteranking;
        else
          stu[j].siteranking = count;
      }
      count++;
    }
    sum += K;
  }
  //还剩下总排名未处理
  sort(stu,stu+sum,sort_Grate); //每一次
sort 排序都是将 结构体里的变量都进行了排序 (当
然我们按照成绩进行排序)

  for( int j = 0; j < sum; j++ ){
    if( stu[j].grate == stu[j - 1].grate)
      stu[j].examranking = stu[j -
1].examranking;
    else
      stu[j].examranking = j + 1;
  }
  cout << sum << endl;
  for( int i = 0; i < sum; i++ ){
    cout << stu[i].id << ' ' <<
stu[i].examranking << ' ' <<
stu[i].examsite << ' ' <<
stu[i].siteranking << endl;
  }
}
```

## 7-42 整型关键字的散列映射

```
#include <iostream>
#include <unordered_map>
#include <string>
using namespace std;
int vis[10005], Hash[10005];
int main()
{
    int n, m;
    cin >> n >> m;
    int x;
    for (int i = 0; i < n; i++)
    {
        cin >> x;
        int flag = 0;
        for (int j = 0; j < m; j++)
        {
            if (Hash[j] == x)
            {
                flag = 1;
                cout << " " << j;
                break;
            }
        }
        if (flag == 1)
            continue; //如果一个数有重复的数
字, 输出第一次的位置
        int pos = x % m;
        while (vis[pos])
            pos = (pos + 1) % m;
        vis[pos] = 1;
        Hash[pos] = x;
        if (!i)
            cout << pos;
        else
            cout << " " << pos;
    }
    return 0;
}
```

## 7-43 字符串关键字的散列映射

```
#include <iostream>
#include <cstring>
using namespace std;
inline int read();
int extend_mod(int base, int mod)
{
  return ((base % mod) + mod) % mod;
}
class HashTable
{
  int *p_hashTable = nullptr;
  char **p_keyTable = nullptr;
  int p_size = 0;
  int hash(char str[9])
  {
    return ((str[7] - 'A') + ((str[6] -
'A') << 5) + ((str[5] - 'A') << 10)) %
p_size;
  }
public:
  HashTable(int size) : p_size(size)
  {
    p_hashTable = new int[size];
    p_keyTable = new char *[size];
    memset(p_hashTable, 0, sizeof(int) *
size);
  }
  int insert(char input[9])
  {
    int hashIndex = hash(input);
    // cout << hashIndex << "#\n";
    // Hash conflict check
    if (p_hashTable[hashIndex])
    {
      int sqrBase = 0, hashAdd, hashMinus;
      while (1)
      {
        hashAdd = extend_mod(hashIndex +
sqrBase * sqrBase, p_size);
        hashMinus = extend_mod(hashIndex -
sqrBase * sqrBase, p_size);
        if (p_hashTable[hashAdd] &&
strcmp(p_keyTable[hashAdd], input) == 0)
        {
          return hashAdd;
        }
        if (p_hashTable[hashMinus] &&
strcmp(p_keyTable[hashMinus], input) == 0)
        {
          return hashMinus;
        }
        if (!p_hashTable[hashAdd])
        {
          p_keyTable[hashAdd] = new char[9];
          strcpy(p_keyTable[hashAdd],
input);
          p_hashTable[hashAdd]++;
          return hashAdd;
        }
        if (!p_hashTable[hashMinus])
        {
          p_keyTable[hashMinus] = new
char[9];
          strcpy(p_keyTable[hashMinus],
input);
          p_hashTable[hashMinus]++;
          return hashMinus;
        }
        sqrBase++;
      }
    }
    else
    {
      p_keyTable[hashIndex] = new char[9];
      strcpy(p_keyTable[hashIndex], input);
      p_hashTable[hashIndex]++;
    }
    return hashIndex;
  }
};
int main(void)
{
  char input[9];
  char buffer[9];
  int strCount, hashLength;
  cin >> strCount >> hashLength;
  HashTable table(hashLength);
  for (int i = 0; i < strCount; i++)
  {
    cin >> input;
    sprintf(buffer, "AAAAAAAA");
    sprintf(buffer, "%8s", input);
    printf(" %d" + !i,
table.insert(buffer));
  }
  return 0;
}
```

## 7-44 基于词频的文件相似度

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
typedef pair<double, int> PII;
bool cmp(PII a, PII b)
{
  return a > b;
}
int main()
{
  int n, m;
  cin >> n >> m;
  vector<vector<PII>> v(5);
  for (int i = 0; i < n; i++)
  {
    double a, b, c;
    cin >> a >> b >> c;
    v[1].push_back({a, i});
    v[2].push_back({b, i});
    v[3].push_back({a / c, i});
    v[4].push_back({b / c, i});
  }
  for (int i = 1; i <= 4; i++)
    sort(v[i].begin(), v[i].end(), cmp);
  bool flag = 0;
  for (int i = 1; i <= m; i++)
  {
    int t, rank = n * 2, r;
    cin >> t;
    for (int j = 1; j <= 4; j++)
      for (int k = 0; k < v[j].size(); k++)
        if (v[j][k].second == t)
          for (int l = 0; l < v[j].size();
l++)
            if (v[j][l].first ==
v[j][k].first)
              if (l + 1 < rank)
              {
                rank = l + 1;
                r = j;
              }
    if (!flag)
      flag = 1;
    else
      cout << ' ';
    printf("%d:%d", rank, r);
  }
  return 0;
}
a #include<cstdio>
#include <algorithm>
#include <map>
#include <cctype>
  using namespace std;
int n, m;
char s[11], ch;
int com[101][101], num[101], sn;
map<string, bool> mp[101];
int main()
{
  scanf("%d", &n);
  for (int i = 1; i <= n; i++)
  {
    while ((ch = tolower(getchar())) !=
'#')
    {
      if (ch >= 'a' && ch <= 'z')
      {
        if (sn < 10)
          s[sn++] = ch;
      }
      else
      {
        s[sn] = 0;
        if (sn > 2)
          mp[i][s] = 1;
        sn = 0;
      }
    }
    for (map<string, bool>::iterator it =
mp[i].begin(); it != mp[i].end(); it++)
    {
      for (int j = 0; j < i; j++)
      {
        com[i][j] = com[j][i] +=
mp[j][it->first];
      }
    }
    com[i][i] = num[i] = mp[i].size();
  }
  scanf("%d", &m);
  int a, b;
  for (int i = 0; i < m; i++)
  {
    scanf("%d%d", &a, &b);
    printf("%.1f%%\n", com[a][b] * 100.0 /
(num[a] + num[b] - com[a][b]));
  }
}
```

## 7-45 航空公司 VIP 客户查询

```
 #include <iostream>
#include <cstring>
#include <algorithm>
#include <unordered_map>

using namespace std;

unordered_map<string,int> mp;

int main()
{

    int n,t;
    scanf("%d%d",&n,&t);

    for(int i=0;i<n;++i)
    {
        string id;
        int num;
        id.resize(18);
        scanf("%s%d",&id[0],&num);

        if(num < t)
            mp[id] += t;
        else
            mp[id] += num;
    }

    int m;
    scanf("%d",&m);

    for(int i=0;i<m;++i)
    {
        string id;
        id.resize(18);
        scanf("%s",&id[0]);
        if(mp[id] == 0)
            puts("No Info");
        else
            printf("%d\n",mp[id]);
    }
    return 0;
}
```

## 7-46 新浪微博热门话题

```
#include <iostream>
#include <map>
#include <string>
#include <cstring>
using namespace std;
map<string, long long int> num; // 某话题出
现次数
int main()
{
  long long int N;
  cin >> N;
  cin.get(); // 吸收回车
  char ch;
  char s[150];    // 临时储存字符串
  int sp = 0;          // sp 的指针
  long long int sum = 0; // 还有 sum 条并列热
门话题
  long long int max = 0; // 最热门话题出现次
数
  char hot[150];           // 存储最热门的
话题
  for (long long int i = 0; i < N; i++)
  {
    map<string, bool> flag; // 标记字符串是
否在本行出现过
    while ((ch = tolower(cin.get())) !=
'\n')
    {
      if (ch == '#')
      {
        sp = 0;
        while ((ch = tolower(cin.get())) !=
'#')
        {
          if ((ch >= 'a' && ch <= 'z') ||
(ch >= '0' && ch <= '9'))
          {
            s[sp++] = ch;
          }
          else
          {
            if (sp != 0 && (s[sp - 1] >= 'a'
&& s[sp - 1] <= 'z') || (s[sp - 1] >= '0'
&& s[sp - 1] <= '9'))
            {
              s[sp++] = ' ';
      }
     }
    }
    if (sp != 0 && s[sp - 1] == ' ')
     s[sp - 1] = '\0';
    else
     s[sp] = '\0';
    if (!flag[s])
    {
     num[s]++;
     flag[s] = true;
     if (num[s] > max)
     {
      max = num[s];
      sum = 0;
      strcpy(hot, s);
     }
     else if (num[s] == max)
     {
      sum++;
      if (strcmp(s, hot) < 0)
       strcpy(hot, s);
     }
    }
   }
  }
 }
 hot[0] = toupper(hot[0]);
 cout << hot << endl
   << max << endl;
 if (sum > 0)
 {
  cout << "And " << sum << " more ...";
 }
 return 0;
}
```

## 7-47 打印选课学生名单

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
  int n, k, c, m;
  scanf("%d %d", &n, &k);
  string name;
  name.resize(5);
  vector<vector<string>> info(k,
vector<string>());
  for (int i = 0; i < n; ++i)
  {
    scanf("%s %d", &name[0], &c);
    for (int j = 0; j < c; ++j)
    {
      scanf("%d", &m);
      info[m - 1].push_back(name);
    }
  }
  for (int i = 0; i < k; ++i)
  {
    printf("%d %d\n", i + 1,
info[i].size());
    sort(info[i].begin(),
info[i].end());
    for (auto &w: info[i])
      puts(w.c_str());
  }
}
```

## 7-48 银行排队问题之单窗口“夹塞”版

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <map>
using namespace std;
const int maxn = 10010;
struct Node
{
 string name;
 int arr, ser;
} a[maxn];
int main()
{
 map<string, int> hs;
 map<string, string> fre;
 int n, m;
 cin >> n >> m;
 for (int i = 0; i < m; ++i)
 {
  string root;
  int l;
  cin >> l;
  for (int j = 0; j < l; ++j)
  {
   string name;
   cin >> name;
   if (j == 0)
    root = name;
   hs[name] = 0;
   fre[name] = root;
  }
 }
 for (int i = 0; i < n; ++i)
 {
  string name;
  int arr, ser;
  cin >> name >> arr >> ser;
  ser = min(60, ser);
  a[i] = {name, arr, ser};
 }
 double wait = 0;
 double res = 0;
 double total = 0;
 for (int i = 0; i < n; ++i)
 {
  string name = a[i].name;
  int arr = a[i].arr;
  int ser = a[i].ser;
  if (hs[name] == 0)
  {
   if (arr > total)
    total = arr + ser;
   else
   {
    wait = total - arr;
    if (wait < 0)
     wait = 0;
    res += wait;
    total += ser;
   }
   cout << name << endl;
   hs[name] = 1;
  }
  else
   continue;
  for (int j = i + 1; j < n; ++j)
  {
   if (hs[a[j].name] == 0 && fre[name]
== fre[a[j].name])
   {
    if (a[j].arr <= total)
    {
     wait = total - a[j].arr;
     if (wait < 0)
      wait = 0;
     res += wait;
     total += a[j].ser;
     hs[a[j].name] = 1;
     cout << a[j].name << endl;
    }
   }
  }
 }
 printf("%.1f\n", res / n);
 return 0;
}
```

## 7-49 打印学生选课清单

```
#include <iostream>
#include <stdlib.h>
#include <cstring>
#include<stdio.h>
#define c2n(i) sname[i]-'A'
#define slink
s[c2n(0)][c2n(1)][c2n(2)][sname[3]-'0']

using namespace std;


typedef struct NODE
{
  int lesson;
  struct NODE *next;
} listt,*List;

//注意理解这个递归插入，神奇之处在于可以往两边
增加结点
List insertList(List cur,int lname)
{
 //先按从大到小，便于后续递归计算总数
 /**
  1.cur 为空时:往最后插入
  2.课程编号大于当前结点课程编号时:往最前面
插入，返回新结点指针
  3.否则指针后移与下一个结点比较。
**/
if(!cur||cur->lesson<lname)
  {
    List node = new listt();
    node->lesson = lname;
    node->next = cur;
    return node;
  }else //head 课程编号小，往后放
    cur->next =
insertList(cur->next,lname);
  return cur; //返回头指针
}


//递归输出顺便统计总数
void print(List l,int n=0)
{

  if(l)
  {
    print(l->next,++n);
    cout<<" "<<l->lesson;
  }
  else
    cout<<n;
}

List s[26][26][26][10];
int main()
{

  for(int i = 0 ; i<26 ; i++)
    for(int j = 0 ; j<26 ; j++)
      for(int k = 0 ; k<26 ; k++)
        for(int x = 0 ; x<10 ; x++)
          s[i][j][k][x] = NULL;


  int sn,ln,lsn,lname;
  char sname[5]; //注意必须多定义 1 个，否则
cin 输入时超界溢出影响其他值，测试结果导致
lname 为 0
  cin>>sn>>ln;
  for(int i = 0 ; i<ln ; i++)
```

```
    {
        cin>>lname>>lsn;
        for(int j = 0 ; j<lsn ; j++)
        {
            cin>>sname;
            slink =
insertList(slink,lname);
        }
    }
    for(int i = 0 ; i<sn ; i++)
    {
        cin>>sname;
        cout<<sname<<" ";
        print(slink);
        cout<<endl;
    }
    return 0;
}
```

## 7-50 畅通工程之局部最小花费问题

```
#include <iostream>
#include <vector>
#include <algorithm>
#define INF 9999999
using namespace std;
struct edge {
    int a, b, cost;
    bool operator<(const edge e) const {
        return cost < e.cost;
    }
};
int parent[200] = {};
int findParent(int num) {
    if(parent[num] == num) return num;
    return parent[num] =
findParent(parent[num]);
}
void Union(int a, int b) {
    int pa = findParent(a);
    int pb = findParent(b);
    if(pa != pb) parent[pb] = pa;
}
int main() {
    int N, cnt, a, b, mon, build;
    vector<edge> v;
    scanf("%d", &N);
    cnt = N * (N - 1) / 2;
    // 初始化集合
    for(int i = 0; i < 200; i++)
        parent[i] = i;
    // 读数
    while(cnt--) {
        scanf("%d%d%d%d", &a, &b, &mon,
&build);
        if(build) Union(a, b);
        else v.push_back({a, b, mon});
    }
    int sum = 0;
    // Kruskal
    sort(v.begin(), v.end());
    for(int i = 0; i < v.size(); i++) {
        if(findParent(v[i].a) !=
findParent(v[i].b)) {
            sum += v[i].cost;
            Union(v[i].a, v[i].b);
        }
    }
    printf("%d", sum);
    return 0;
}
```

## 7-51 两个有序链表序列的合并

```
#include <iostream>
#include <cstring>
#define ARRAY_MEMORY_SIZE 100010
#define DATA_MAX_SIZE 10000
using namespace std;
class ListNode
{
public:
  int data;
  ListNode *next;
};
class List
{
public:
  List() { head = nullptr; }
  void push_back(int data)
  {
    ListNode *newNode = new ListNode;
    newNode->next = nullptr;
    newNode->data = data;
    if (head == nullptr)
    {
      head = newNode;
      end = newNode;
    }
    else
    {
      end->next = newNode;
      end = newNode;
    }
  }
  ListNode *get_head() { return head; }
private:
  ListNode *head;
  ListNode *end;
};
int main(void)
{
  List list1, list2;
  int buf;
  while (scanf("%d", &buf) == 1 && buf != -
1)
  {
    list1.push_back(buf);
  }
  while (scanf("%d", &buf) == 1 && buf != -
1)
  {
    list2.push_back(buf);
  }
  List ans;
  ListNode *scan1 = list1.get_head();
  ListNode *scan2 = list2.get_head();
  while (scan1 != nullptr && scan2 !=
nullptr)
  {
    if (scan1->data < scan2->data)
    {
      ans.push_back(scan1->data);
      if (scan1 != nullptr)
        scan1 = scan1->next;
    }
    else
    {
      ans.push_back(scan2->data);
      if (scan1 != nullptr)
        scan2 = scan2->next;
    }
  }
  scan1 = scan1 == nullptr ? scan2 : scan1;
  while (scan1 != nullptr)
  {
    ans.push_back(scan1->data);
    scan1 = scan1->next;
  }
  scan1 = ans.get_head();
  if (scan1 == nullptr)
  {
    printf("NULL");
  }
  else
  {
    bool isFirst = true;
    while (scan1 != nullptr)
    {
      printf(" %d" + isFirst, scan1->data);
      isFirst = false;
      scan1 = scan1->next;
    }
  }
}
```

## 7-52 两个有序链表序列的交集

```
#include <iostream>
#include <cstring>
#define ARRAY_MEMORY_SIZE 100010
#define DATA_MAX_SIZE 10000
using namespace std;

class ListNode
{
public:
  int data;
  ListNode *next;
};

class List
{
public:
  List() { head = nullptr; }

  void push_back(int data)
  {
    ListNode *newNode = new ListNode;
    newNode->next = nullptr;
    newNode->data = data;

    if (head == nullptr)
    {
      head = newNode;
      end = newNode;
    }
    else
    {
      end->next = newNode;
      end = newNode;
    }
  }

  ListNode *get_head() { return head; }

private:
  ListNode *head;
  ListNode *end;
};

int main(void)
{
  List list1, list2;
  int buf;
  while (scanf("%d", &buf) == 1 && buf != -
1)
  {
    list1.push_back(buf);
  }

  while (scanf("%d", &buf) == 1 && buf != -
1)
  {
    list2.push_back(buf);
  }

  List ans;
  ListNode *scan1 = list1.get_head();
  ListNode *scan2 = list2.get_head();
  while (scan1 != nullptr && scan2 !=
nullptr)
  {
    if (scan1->data == scan2->data)
    {
      ans.push_back(scan1->data);
      if (scan1 != nullptr)
        scan1 = scan1->next;
      if (scan1 != nullptr)
        scan2 = scan2->next;
    }
    else
    {
      if (scan1->data < scan2->data)
      {
        if (scan1 != nullptr)
          scan1 = scan1->next;
      }
      else
      {
        if (scan1 != nullptr)
          scan2 = scan2->next;
      }
    }
  }

  scan1 = ans.get_head();
  if (scan1 == nullptr)
  {
    printf("NULL");
  }
  else
  {
    bool isFirst = true;
    while (scan1 != nullptr)
    {
      printf(" %d" + isFirst, scan1->data);
      isFirst = false;
      scan1 = scan1->next;
    }
  }
}
```

### 7-53 两个有序序列的中位数

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int a[100009];
int b[100009];
int binarysearch1(int l1, int r1, int l2, 
int r2)
{
  int mid1 = (l1 + r1) / 2;
  int mid2 = (l2 + r2 + 1) / 2; //+1 处理,
可以在遇到偶数序列时处理

  if (l1 == r1 && l2 != r2)
  {
    if (a[l1] > b[r2])
      return b[r2];
    else
      return a[l1];
  }
  if (l2 == r2 && l1 != r1)
  {
    if (b[l2] > a[r1])
      return a[r1];
    else
      return b[l2];
  }
  if (l1 == r1 && l2 == r2)
  {
    if (a[l1] > b[l2])
      return b[l2];
    else
      return a[l1];
  }

  else
  {
    if (l1 == r1 - 1 && l2 == r2 - 1) // 当
a、b 数组都只剩两个数字时
    {
      mid1 = (l1 + r1) / 2;
      mid2 = (l2 + r2) / 2;
    }
    if (a[mid1] == b[mid2])
      return a[mid1];
    else if (a[mid1] > b[mid2])
      binarysearch1(l1, mid1, mid2, r2);
    else
      binarysearch1(mid1, r1, l2, mid2);
  }
}
int main()
{
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
    cin >> a[i];
  for (int i = 0; i < n; i++)
    cin >> b[i];
  cout << binarysearch1(0, n - 1, 0, n - 1)
<< endl;
}
```