AC了前三道题，D题采用了建树的方法，所以有一半用例超时；E题没什么思路而且没什么时间了就没来得及做了。

A □□

给定五个 0~9 范围内的整数 a1, a2, a3, a4, a5。如果能从五个整数中选出三个并且这三个整数的和为 10 的倍数（包括 0），那么这五个整数的权值即为剩下两个没被选出来的整数的和对 10 取余的结果，显然如果有多个三元组满□和是 10 的倍数，剩下两个数之和对 10 取余的结果都是相同的；如果 选不出这样三个整数，则这五个整数的权值为 -1。

现在给定 T 组数据，每组数据包含五个 0~9 范围内的整数，分别求这 T 组数据中五个整数的权值。

【输□格式】

第□□□个整数 T (1<=T<=1000)，表□数据组数。

接下来 T □，每□ 5 个 0~9 的整数，表□□组数据。

【输出格式】

输出 T □，每□□个整数，表□每组数据中五个整数的权值。

【样例输□】

4

1 0 0 1 0

1 0 0 8 6

3 4 5 6 7

4 5 6 7 8

【样例输出】

2

【时空限制】

2500ms，256MB

```
//解题思路：由于数组大小只有5，尝试暴力解法
#include <iostream>
using namespace std;

int main() {
 int T, a[5];
 cin >> T;
 //T组数据
 while (T--) {
  int sum = 0, flag = 0, ans;
  for (int i = 0; i < 5; i++) {
   cin >> a[i];
   sum += a[i];
  }
  //从5个数中选出3个进行判断
  for (int i = 0; i < 5; i++) {
   for (int j = i + 1; j < 5; j++) {
    for (int k = j + 1; k < 5; k++) {
     int sumOfThree = a[i] + a[j] + a[k];
     if (sumOfThree % 10 == 0) {
      flag = 1;
      ans = (sum - sumOfThree) % 10;
      break;
     }
    }
   }
  }
  if (flag == 0) cout << -1 << endl;
  else cout << ans << endl;
 }
 return 0;
}
```

B 打地鼠

给定 n 个整数 a1, a2, …, an 和□个 d，你需要选出若□个整数，使得将这些整数从□到□排好序之 后，任意两个相邻的数之差都不□于给定的 d，问最多能选多少个数出来。

【输□格式】

第□□两个整数 n,d (1<=n<=10^5, 0<=d<=10^9)，分别表□整数个数和相邻整数差的下界。

第□□ n 个整数 a1, a2, …, an (1<=ai<=10^9, 1<=i<=n)，表□给定的 n 个整数。

【输出格式】

仅□□□个整数，表□答案。

【样例输□】

6 2

1 4 2 8 5 7

【样例输出】

3

【时空限制】

2500ms，256MB

```
//解题思路：动态规划 + 贪心算法
#include <iostream>
using namespace std;
const int MAXN = 100010;
int a[MAXN], dp[MAXN];

int main(){
 int n, d;
 cin >> n >> d;
 for (int i = 0; i < n; i++) {
  cin >> a[i];
 }
 sort(a, a + n);
 dp[0] = 1;
 for (int i = 1; i < n; i++) {
  if (a[i] - a[i - 1] >= d) {
   dp[i] = dp[i - 1] + 1;
  } else {
   dp[i] = dp[i - 1];
   a[i] = a[i - 1];
  }
 }
 cout << dp[n - 1];
 return 0;
}
```

C 排队打饭

下课了，有 n 位同学陆续赶到□堂进□排队打饭，其中第 i 位同学的到达时间为 ai，打饭耗时为 ti， 等待时间上限为 bi，即如果其在第 ai+bi 秒的时刻仍然没有轮到他开始打饭，那么他将离开打饭队列 另寻吃饭的地□。问每位同学的开始打饭时间，或者指出其提前离开了队伍（如果这样则输出 -1）。 【输□格式】

第□□□个整数 n (1<=n<=10^5)，表□来打饭的同学数量。

接下来 n □，每□三个整数 ai,ti,bi (1<=ai,ti,bi<=10^9, 1<=i<=n)，分别表□每位同学的到达时间、打 饭耗时、等待时间上限。

保证 a1<a2<…<an

【输出格式】

□□ n 个整数，表□每位同学的开始打饭时间或者 -1（如果该同学提前离开了队伍）。

【样例输□】

4

1 3 3

2 2 2

3 9 1

4 3 2

【样例输出】

1 4 -1 6

【时空限制】

5000ms，256MB

```
//解题思路：利用队列实现
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;

struct node{
 long long a;
 long long t;
 long long b;
};

int main() {
 int n;
 cin >> n;
 queue<node> q;
 vector<int> v;
 while (n--) { //把所有学生按照先后次序入队
  node stu;
  cin >> stu.a >> stu.t >> stu.b;
  q.push(stu);
 }
 int current = 1;
 while (!q.empty()) {
  node temp = q.front();
  q.pop();
  int ans;
  if (temp.a + temp.b >= current) { //如果未超过等待时间
   if (current > temp.a){
    ans = current;
    current += temp.t;
   } else {
    ans = temp.a;
    current = temp.a + temp.t;
   }
  } else { //如果超过等待时间
   ans = -1;
  }
  v.push_back(ans);
 }
 for (int i = 0; i < v.size(); i++) {
  if (i == 0) cout << v[i];
  else cout << " " << v[i];
 }
 return 0;
}
```

D 二叉搜索树

给定□个 1~n 的排列 P，即□度为 n，且 1~n 中所有数字都恰好出现□次的序列。现在按顺序将排列 中的元素□□插□到初始为空的□叉搜索树中（左□右□），问最后每个节点的□亲节点的元素是什 么。特别地，根节点的□亲节点元素视为 0。

【输□格式】

第□□□个整数 n (1<=n<=10^5)，表□排列 P 中的元素个数。

第□□ n 个整数 p1, p2, …, pn (1<=pi<=n, 1<=i<=n)，表□给定的排列。

【输出格式】

□□ n 个整数，其中第 i 个整数 ai 表□元素 i 对应节点的□亲节点的元素。特别地，根节点的□亲节 点元素视为 0。

【样例输□】

5

2 3 5 1 4

【样例输出】

2 0 2 5 3

【时空限制】

5000ms，256MB

我的做法是建树，但是部分用例超时了！

```
//思路：在树的结点定义时，定义一个变量，用来表示父亲结点的元素。二叉搜索树建树建树之后再中序遍历依次输出该变量即可。
#include <iostream>
using namespace std;
int n, flag = 0; //flag用来处理答案最后一个字符不输出空格

//树的结构体定义
struct node{
 int data;
 int father;
 node* lchild;
 node* rchild;
};

//新建结点
node* newNode(int x, int f){
 node* Node = new node;
 Node->data = x;
 Node->father = f;
 Node->lchild = NULL;
 Node->rchild = NULL;
 return Node;
}

//插入结点
void insert(node* &root, int x, int f){
 if (root == NULL){
  root = newNode(x, f);
  return;
 }
 f = root->data;
 if (x < root->data){
  insert(root->lchild, x, f);
 } else {
  insert(root->rchild, x, f);
 }
}

//建立二叉搜索树
node* create(int data[], int n) {
 node* root = NULL;
 for (int i = 0; i < n; i++) {
  insert(root, data[i], 0);
 }
 return root;
}

//中序遍历
void inorder(node* root) {
 if (root == NULL) return;
 inorder(root->lchild);
 if(flag != n - 1) {
  cout << root->father << " ";
  flag++;
 } else {
  cout << root->father;
 }
 inorder(root->rchild);
}

int main() {
 cin >> n;
 int a[n];
 for (int i = 0; i < n; i++){
```

```
  cin >> a[i];
 }
 node *root = create(a, n);
 inorder(root);
 return 0;
}
```

不超时的做法如下：

```
//用一个表示父亲结点的数组+一个表示层数的map数组即可表示一颗搜索二叉树的插入过程
#include <iostream>
#include <map>
using namespace std;
int f[100010]; //f[i]用来保存元素i的父亲结点

int main() {
 ios::sync_with_stdio(0); //加快cin的输入速度
 int n, mx = 0;  //n是结点个数，mx用于保存当前最大的结点
 cin >> n;
 map<int, int> m; //m保存的是元素i所在的层数
 m[0] = 0;
 map<int, int>::iterator it, it1;
 f[0] = -1;
 for (int i = 0, t; i < n; i++) {
  cin >> t;
  if (t > mx) {
   f[t] = mx;
   m[t] = m.rbegin()->second+1;
   mx = t;
  } else {
   it = m.upper_bound(t);
   it1 = it--;
   if (it->second > it1->second) {
    f[t] = it->first;
    m[t] = it->second + 1;
   } else {
    f[t] = it1->first;
    m[t] = it1->second + 1;
   }
  }
 }
 for (int i = 1; i <= n; i++) {
  if (i == 1) cout << f[i];
  else cout << " " << f[i];
 }
 return 0;
}
```

E 序列

给定□个□为 n 的序列 A，其中序列中的元素都是 0~9 之间的整数，对于□个□度同样为 n 整数序列 B，定义其权值为 |A_i-B_i| (1<=i<=n) 之和加上 (B_j-B_j+1)^2 (1<=j<n) 之和。求所有□为 n 的整数序 列中，权值最□的序列的权值是多少。

【输□格式】

第□□□个整数 n (1<=n<=10^5)，表□序列 A 的□度。

第□□ n 个整数 a1, a2, …, an (0<=ai<=9, 1<=i<=n)，表□序列 A 中的元素。

【输出格式】

仅□□□个整数，表□答案。

【样例输□】

6

1 4 2 8 5 7

【样例输出】

11

【解释】

A 数组是 [1 4 2 8 5 7]

B 数组可以是 [3 4 4 5 5 6]。

权值为 |A_i - B_i| (1<=i<=n) 之和加上 (B_j - B_j+1)^2 (1<= j <n) 之和。

权值第□部分|A_i - B_i| (1<=i<=n)之和为：

|1 - 3| + |4 - 4| + |2 - 4| + |8 - 5| + |5 - 5| + |7 - 6| = 2 + 0 + 2 + 3 + 0 + 1 = 8

权值第□部分(B_j - B_j+1)^2 (1<= j <n) 之和为：

(3 - 4)^2 + (4 - 4)^2 + (4 - 5)^2 + (5 - 5)^2 + (5 - 6)^2 = 1 + 0 + 1 + 0 + 1 = 3

所以总权值为 8 + 3 = 11。

【时空限制】

2500ms，256MB

以下题解转载于：https://blog.csdn.net/hyacinthhome/article/details/105952974

分析：序列A给定，思考下，对于b数组每个值的选取对答案的贡献是什么？？其实只有前

一个把，因为贡献只不过是加上它和它前一个值的差值的平方，和它和A数组对应位置的

差值绝对值，所以咱们就维护一个now数组,now[i]表示的是当前结束的B的这个

位置值为i，那么更新就需要前一个位置的0-9种情况取最优，那么复杂度就是

O(10*10*n)，n=1e5,那么差不多1秒内了

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
int tp1[10][10], tp2[10][10];
int now[10], now_temp[10];  //目前结尾为i的最小函数值，复杂度O(n*10*10)约等于1e7，差不多一秒
const int INF=0x3f3f3f3f;

void init()
{
 for(int i = 0; i < 10; i++){
  for(int j = 0; j < 10; j++)
  {
   tp1[i][j] = i < j ? (j - i) : (i - j);
   tp2[i][j] = (i - j) * (i - j);
  }
 }
}

int main()
{
 int n, num;
 init();
 while(scanf("%d",&n) != EOF) {
  memset(now, 0, sizeof(now));
  for(int i = 0; i < n; i++) {
   scanf("%d", &num);
   for(int j = 0; j < 10; j++) {
    int temp = INF;
    for (int k = 0; k < 10; k++) {
     int tp;
     if (i) {
      tp = now[k] + tp1[j][num] + tp2[j][k];
     }
     else {
      tp = now[k] + tp1[j][num];
     }
     if(tp < temp) temp = tp;
    }
    now_temp[j] = temp;
   }
   for(int i = 0; i < 10; i++)
    now[i] = now_temp[i];
  }
  int ans = INF;
  for(int i = 0; i < 10; i++)
   ans = min(ans, now[i]);
  printf("%d\n", ans);
```

```
 }
 return 0;
}
```