Formatting Submissions for a USENIX Conference: An (Incomplete) Example

XeLaTeX

Ruohong Jiao
First Institution

Second Name
Second Institution

Third Name
Third Institution

Fourth Name
Fourth Institution

Fifth Name
Fifth Institution

2020-03-28

摘要

Your abstract text goes here. Just a few facts. Whet our appetites. Not more than 200 words, if possible, and preferably closer to 150.

目录

1 引言

2 Footnotes, Verbatim, and Citations

3 Floating Figures and Lists

3.1 LaTeX-ing Your TeX File

Acknowledgments

The USENIX latex style is old and very tired, which is why there's no \acks command for you to use when acknowledging. Sorry.

Availability

USENIX program committees give extra points to submissions that are backed by artifacts that are publicly available. If you made your code or data available, it's worth mentioning this fact in a dedicated section.

参考文献

附录 A Problem List

A.1 Leetcode 1

Problem Description:

两数之和

给定一个整数数组 *nums* 和一个目标值 *target*，请你在该数组中找出和为目标值的那两个整数，并返回他们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素不能使用两遍。

Sample:

input:

```
给定 nums = [2, 7, 11, 15], target = 9
```

otput:

```
因为 nums[0] + nums[1] = 2 + 7 = 9
所以返回 [0, 1]
```

Solution (Codes at B.1):

没有说明输入数字一定是正整数，不能先排序后提取小于 *target* 的数进行求解。

懒一点，n^2 循环。勤快一点用红黑树、堆进行存储然后查询 *nlogn* 。

A.2 Leetcode 2

Problem Description:

两数相加

给出两个非空的链表用来表示两个非负的整数。其中，它们各自的位数是按照逆序的方式存储的，并且它们的每个节点只能存储一位数字。

如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。

您可以假设除了数字 0 之外，这两个数都不会以 0 开头。

Sample:

input:

```
输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)
```

otput:

```
输出：7 -> 0 -> 8
原因：342 + 465 = 807
```

Solution (Codes at B.2):

大数加法，注意指针/引用转换。

A.3 Leetcode 3

Problem Description:

无重复字符的最长子串

给定一个字符串，请你找出其中不含有重复字符的最长子串的长度。

Sample:

input:

```
输入："abcabcbb"

输入："bbbbb"

输入："pwwkew"
```

otput:

```
输出：3
解释：因为无重复字符的最长子串是 "abc"，所以其长度为 3。

输出：1
解释：因为无重复字符的最长子串是 "b"，所以其长度为 1。

输出：3
解释：因为无重复字符的最长子串是 "wke"，所以其长度为 3。
     请注意，你的答案必须是 子串 的长度，"pwke" 是一个子序列，不是子串。
```

Solution (Codes at B.3):

标记 l 和 r，遍历一遍，更新答案。时间方面，能用数组不用 STL。

A.4 Leetcode 6

Problem Description:

Z 字形变换

将一个给定字符串根据给定的行数，以从上往下、从左到右进行 Z 字形排列。

比如输入字符串为"LEETCODEISHIRING" 行数为 3 时，排列如下：

```
L   C   I   R
E T O E S I I G
E   D   H   N
```

之后，你的输出需要从左往右逐行读取，产生出一个新的字符串，比如："LCIRETOESIIGEDHN"。

Sample:

input:

```
输入：s = "LEETCODEISHIRING", numRows = 3

输入：s = "LEETCODEISHIRING", numRows = 4
```

otput:

```
输出："LCIRETOESIIGEDHN"

输出："LDREOEIIECIHNTSG"
解释：

L     D     R
E   O E   I I
E C   I H   N
T     S     G
```

Solution (Codes at B.4):
计算第一行字符 id，前后搜查。

A.5 Leetcode 11

Problem Description:
盛最多水的容器
给你 n 个非负整数 $a1,a2,\cdots,an$，每个数代表坐标中的一个点 (i,ai)。在坐标内画 n 条垂直线，垂直线 i 的两个端点分别为 (i,ai) 和 $(i,0)$。找出其中的两条线，使得它们与 x 轴共同构成的容器可以容纳最多的水。
说明：你不能倾斜容器，且 n 的值至少为 2。
Sample:
input:

```
输入：[1,8,6,2,5,4,8,3,7]
```

otput:

```
输出：49
```

Solution (Codes at B.5):
两边到中间遍历一遍，贪心选择更高的边维持原状。

A.6 Leetcode 15

Problem Description:
三数之和
给你一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a,b,c，使得 $a+b+c=0$？请你找出所有满足条件且不重复的三元组。
注意：答案中不可以包含重复的三元组。
Sample:
input:

```
给定数组 nums = [-1, 0, 1, 2, -1, -4],
```

otput:

```
满足要求的三元组集合为：
[
  [-1, 0, 1],
  [-1, -1, 2]
]
```

Solution (Codes at B.6):
不要依赖 STL，不是所有题目都需要离散化处理。对于一般的链表问题，去重遍历就可以达到离散化的效果。
先排序，指定一个数值的基础上再挑选符合条件的另外两个数值。贪心匹配两个数值的大小。

A.7 Leetcode 16

Problem Description:

最接近的三数之和

给定一个包括 n 个整数的数组 nums 和一个目标值 target。找出 nums 中的三个整数，使得它们的和与 target 最接近。返回这三个数的和。假定每组输入只存在唯一答案。

提示：

$$3 <= nums.length <= 10^3$$

$$-10^3 <= nums[i] <= 10^3$$

$$-10^4 <= target <= 10^4$$

Sample:

input:

```
输入：nums = [-1,2,1,-4], target = 1
```

otput:

```
输出：2
解释：与 target 最接近的和是 2 (-1 + 2 + 1 = 2) 。
```

Solution (Codes at B.7):

不要依赖 STL，不是所有题目都需要离散化处理。对于一般的链表问题，去重遍历就可以达到离散化的效果。先排序，指定一个数值的基础上再挑选符合条件的另外两个数值。贪心匹配两个数值的大小。

A.8 Leetcode 19

Problem Description:

删除链表的倒数第 N 个节点

给定一个链表，删除链表的倒数第 n 个节点，并且返回链表的头结点。

Sample:

input:

```
给定一个链表：1->2->3->4->5, 和 n = 2.
```

otput:

```
当删除了倒数第二个节点后，链表变为 1->2->3->5.
```

Solution (Codes at B.8):

递归记录个数。

A.9 Leetcode 25

Problem Description:

K 个一组翻转链表

给你一个链表，每 k 个节点一组进行翻转，请你返回翻转后的链表。

k 是一个正整数，它的值小于或等于链表的长度。

如果节点总数不是 k 的整数倍，那么请将最后剩余的节点保持原有顺序。

说明：

你的算法只能使用常数的额外空间。

你不能只是单纯的改变节点内部的值，而是需要实际进行节点交换。

Sample:

input:

```
给你这个链表：1->2->3->4->5
```

otput:

```
当 k = 2 时，应当返回: 2->1->4->3->5

当 k = 3 时，应当返回: 3->2->1->4->5
```

Solution (Codes at B.9):

标记子链表首尾，化简为链表反转问题，递归解决。

A.10 Leetcode 26

Problem Description:

删除排序数组中的重复项

给定一个排序数组，你需要在原地删除重复出现的元素，使得每个元素只出现一次，返回移除后数组的新长度。

不要使用额外的数组空间，你必须在原地修改输入数组并在使用 $O(1)$ 额外空间的条件下完成。

说明:

为什么返回数值是整数，但输出的答案是数组呢?

请注意，输入数组是以引用方式传递的，这意味着在函数里修改输入数组对于调用者是可见的。

你可以想象内部操作如下:

```
输入: nums = [-1,2,1,-4], target = 1

// nums 是以“引用”方式传递的。也就是说，不对实参做任何拷贝
int len = removeDuplicates(nums);

// 在函数里修改输入数组对于调用者是可见的。
// 根据你的函数返回的长度，它会打印出数组中该长度范围内的所有元素。
for (int i = 0; i < len; i++) {
    print(nums[i]);
}
```

Sample:

input:

```
给定数组 nums = [1,1,2],

给定 nums = [0,0,1,1,1,2,2,3,3,4],
```

otput:

```
函数应该返回新的长度 2，并且原数组 nums 的前两个元素被修改为 1，2。
你不需要考虑数组中超出新长度后面的元素。

函数应该返回新的长度 5，并且原数组 nums 的前五个元素被修改为 0，1，2，3，4。
你不需要考虑数组中超出新长度后面的元素。
```

Solution (Codes at B.10):

原始数组已经排过序了。注意当数据量大的时候，判断条件越少越好。

A.11 Leetcode 42

Problem Description:

接雨水

给定 n 个非负整数表示每个宽度为 1 的柱子的高度图，计算按此排列的柱子，下雨之后能接多少雨水。

上面是由数组 $[0,1,0,2,1,0,1,3,2,1,2,1]$ 表示的高度图，在这种情况下，可以接 6 个单位的雨水（蓝色部分表示雨水）。

Sample:

input:

```
输入: [0,1,0,2,1,0,1,3,2,1,2,1]
```

otput:

```
输出: 6
```

Solution (Codes at B.11):

左右打表记录最大值，查询之后得出结果。

老问题，注意当数据量大的时候，判断条件越少越好。以及，多用 C++11 的新初始化特性可以减少时间开销。

A.12 Leetcode 56

Problem Description:

合并区间

给出一个区间的集合，请合并所有重叠的区间。。

Sample:

input:

```
输入: intervals = [[1,3],[2,6],[8,10],[15,18]]

输入: intervals = [[1,4],[4,5]]
```

otput:

```
输出: [[1,6],[8,10],[15,18]]
解释: 区间 [1,3] 和 [2,6] 重叠, 将它们合并为 [1,6].

输出: [[1,5]]
解释: 区间 [1,4] 和 [4,5] 可被视为重叠区间。
```

Solution (Codes at B.12):

自定义排序之后合并。需要注意对是 C++ 快排默认比较顺序是挨个从小到大，如果可以使用原始比较函数就不要自己重新写，还是用初始的更快。

注意 java 的自定义比较函数。

A.13 Leetcode 61

Problem Description:

旋转链表

给定一个链表，旋转链表，将链表每个节点向右移动 k 个位置，其中 k 是非负数。

Sample:

input:

```
输入: 1->2->3->4->5->NULL, k = 2

输入: 0->1->2->NULL, k = 4
```

otput:

```
输出: 4->5->1->2->3->NULL
解释:
向右旋转 1 步: 5->1->2->3->4->NULL
向右旋转 2 步: 4->5->1->2->3->NULL

输出: 2->0->1->NULL
解释:
向右旋转 1 步: 2->0->1->NULL
向右旋转 2 步: 1->2->0->NULL
向右旋转 3 步: 0->1->2->NULL
向右旋转 4 步: 2->0->1->NULL
```

Solution (Codes at B.13):

对 k 取模得到 mk，之后更改倒数第 mk 个 node 开始第子链表顺序到首位。

A.14 Leetcode 121

Problem Description:

买卖股票的最佳时机

给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。

如果你最多只允许完成一笔交易（即买入和卖出一支股票一次），设计一个算法来计算你所能获取的最大利润。

注意：你不能在买入股票前卖出股票。

Sample:

input:

```
输入: [7,1,5,3,6,4]

输入: [7,6,4,3,1]
```

otput:

```
输出: 5
解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。
     注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格；同时，你不能在买入前卖出股票。

输出: 0
解释: 在这种情况下, 没有交易完成, 所以最大利润为 0。
```

Solution (Codes at B.14):

从后向前打表记录最大值，查询之后得出结果。

A.15 Leetcode 138

Problem Description:

复制带随机指针的链表

给定一个链表，每个节点包含一个额外增加的随机指针，该指针可以指向链表中的任何节点或空节点。

要求返回这个链表的深拷贝。

我们用一个由 n 个节点组成的链表来表示输入/输出中的链表。每个节点用一个 $[val, random_index]$ 表示：

val：一个表示 Node.val 的整数。

$random_index$：随机指针指向的节点索引（范围从 0 到 n-1）；如果不指向任何节点，则为 null 。

$-10000 <= Node.val <= 10000$

$Node.random$ 为空（null）或指向链表中的节点。

节点数目不超过 1000。

Sample:

input:

```
输入: head = [[7,null],[13,0],[11,4],[10,2],[1,0]]

输入: head = [[1,1],[2,1]]
输入: head = [[3,null],[3,0],[3,null]]
输入: head = []
```

otput:

```
输出: [[7,null],[13,0],[11,4],[10,2],[1,0]]
输出: [[1,1],[2,1]]
输出: [[3,null],[3,0],[3,null]]
输出: []
解释: 给定的链表为空（空指针），因此返回 null。
```

Solution (Codes at B.15):

map 存储节点 pair。

A.16 Leetcode 141

Problem Description:

环形链表

给定一个链表，判断链表中是否有环。

为了表示给定链表中的环，我们使用整数 pos 来表示链表尾连接到链表中的位置（索引从 0 开始）。如果 pos 是-1，则在该链表中没有环。

进阶：

你能用 $O(1)$（即，常量）内存解决此问题吗？

Sample:

input:

```
输入：head = [3,2,0,-4], pos = 1

输入：head = [1,2], pos = 0

输入：head = [1], pos = -1
```

otput:

```
输出：true
解释：链表中有一个环，其尾部连接到第二个节点。

输出：true
解释：链表中有一个环，其尾部连接到第一个节点。

输出：false
解释：链表中没有环。
```

Solution (Codes at B.16):

快慢指针循环跑圈相遇查重。

A.17 Leetcode 202

Problem Description:

快乐数

编写一个算法来判断一个数 n 是不是快乐数。

快乐数定义为：对于一个正整数，每一次将该数替换为它每个位置上的数字的平方和，然后重复这个过程直到这个数变为 1，也可能是无限循环但始终变不到 1。如果可以变为 1，那么这个数就是快乐数。

如果 n 是快乐数就返回 True ；不是，则返回 False 。

Sample:

input:

```
输入：19
```

otput:

```
输出：true
解释：
12 + 92 = 82
82 + 22 = 68
62 + 82 = 100
12 + 02 + 02 = 1
```

Solution (Codes at B.17):

快慢指针循环跑圈相遇查重。

A.18 Leetcode 206

Problem Description:

反转链表

反转一个单链表。

Sample:

input:

```
输入: 1->2->3->4->5->NULL
```

otput:

```
输出: 5->4->3->2->1->NULL
```

Solution (Codes at B.18):

递归或先找到首尾节点之后 while 循环更新 next。

A.19 Leetcode 209

Problem Description:

长度最小的子数组

给定一个含有 n 个正整数的数组和一个正整数 s，找出该数组中满足其和 $>= s$ 的长度最小的连续子数组，并返回其长度。如果不存在符合条件的子数组，返回 0。

进阶：

如果你已经完成了 $O(n)$ 时间复杂度的解法, 请尝试 $O(nlogn)$ 时间复杂度的解法。

Sample:

input:

```
输入: s = 7, nums = [2,3,1,2,4,3]
```

otput:

```
输出: 2
解释: 子数组 [4,3] 是该条件下的长度最小的子数组。
```

Solution (Codes at B.19):

遍历一遍，滑动窗口更新数值。

或者前缀和，二分搜索数值。

A.20 Leetcode 387

Problem Description:

字符串中的第一个唯一字符

给定一个字符串，找到它的第一个不重复的字符，并返回它的索引。如果不存在，则返回 -1 。

Sample:

input:

```
leetcode
loveleetcode
```

otput:

```
0
2
```

Solution (Codes at B.20):

找到字符出现的首位和末位进行判断，然后取最早出现的。

A.21 Leetcode 876

Problem Description:

链表的中间结点

给定一个带有头结点 head 的非空单链表，返回链表的中间结点。
如果有两个中间结点，则返回第二个中间结点。

Sample:

input:

```
输入：[1,2,3,4,5]

输入：[1,2,3,4,5,6]
```

otput:

```
输出：此列表中的结点 3 (序列化形式：[3,4,5])
返回的结点值为 3 。 (测评系统对该结点序列化表述是 [3,4,5])。
注意，我们返回了一个 ListNode 类型的对象 ans，这样：
ans.val = 3, ans.next.val = 4, ans.next.next.val = 5, 以及 ans.next.next.next = NULL.

输出：此列表中的结点 4 (序列化形式：[4,5,6])
由于该列表有两个中间结点，值分别为 3 和 4，我们返回第二个结点。
```

Solution (Codes at B.21):

快慢指针计数。

附录 B　Code List

B.1　Leetcode 1

C++

```cpp
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  vector<int> twoSum(vector<int>& nums, int target) {
    vector<int> ans;
    for (int i = 0; i < nums.size(); i++)
      for (int j = i + 1; j < nums.size(); j++)
        if (nums[i] + nums[j] == target) {
          // u = i;
          // v = j;
          ans.push_back(i);
          ans.push_back(j);
          break;
        }
    return ans;
  }
  void input(void) {
    while (~scanf("%d %d", &n, &m))
      for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        numbers.push_back(t);
      }
  }
  void solve(void) { twoSum(numbers, m); }
  void otput(void) { printf("%d %d\n", u, v); }

```

```cpp
 private:
  int n, m, t;
  int u, v;
  vector<int> numbers;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();
  sol.solve();
  sol.otput();

  return 0;
}
```

Java

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public int[] twoSum(int[] nums, int target) {
    int[] ans = new int[2];
    for (int i = 0; i < nums.length; i++)
      for (int j = i + 1; j < nums.length; j++)
        if (nums[i] + nums[j] == target) {
          u = i;
          v = j;
          ans[0] = i;
          ans[1] = j;
          break;
        }
    return ans;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
    sol.solve();
    sol.otput();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      m = in.nextInt();
      for (int i = 0; i < n; i++)
        numbers[i] = in.nextInt();
    }

    in.close();
  }

  public void solve() {
```

```
    twoSum(numbers, m);
  }

  public void otput() {
    System.out.println(u + " " + v);
  }

  private int n, m, t;
  private int u, v;
  private int[] numbers = new int[10000];
}
```

B.2 Leetcode 2

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

typedef struct ListNode {
  int val;
  ListNode* next;
  ListNode(int x) : val(x), next(NULL) {}
} ListNode;

class Solution {
 public:
  ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
    ListNode* res = new ListNode(0);
    ListNode* u = l1;
    ListNode* v = l2;
    ListNode* cur = NULL;
    int t = 0;

    while (u != NULL || v != NULL || t != 0) {
      if (cur == NULL) {
        cur = res;
      } else {
        cur->next = new ListNode(0);
        cur = cur->next;
      }
      cur->next = NULL;

      cur->val = t;
      if (u != NULL) {
        cur->val += u->val;
        u = u->next;
      }
      if (v != NULL) {
        cur->val += v->val;
        v = v->next;
      }
      t = cur->val / 10;
      cur->val %= 10;
    }

    return res;
```

```
  }
  void input(void) {
    while (~scanf("%d %d", &n, &m)) {
      a = new ListNode(0);
      b = new ListNode(0);
      ListNode* u = a;
      ListNode* v = b;

      for (int i = 0; i < n; i++) {
        u->next = new ListNode(0);
        u = u->next;
        u->next = NULL;

        scanf("%d", &u->val);
      }
      for (int i = 0; i < m; i++) {
        v->next = new ListNode(0);
        v = v->next;
        v->next = NULL;

        scanf("%d", &v->val);
      }

      solve(a, b);
    }
  }
  void show(ListNode* x) {
    ListNode* cur = x;
    while (cur != NULL) {
      printf("%d", cur->val);
      cur = cur->next;
    }
  }
  void solve(ListNode* x, ListNode* y) {
    ListNode* res;
    res = addTwoNumbers(x, y);
    otput(res);
  }
  void otput(ListNode* x) {
    ListNode* cur = x->next;
    while (cur != NULL) {
      printf("%d", cur->val);
      cur = cur->next;
    }
    cout << endl;
  }

 private:
  int n, m;
  ListNode *a, *b;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
```

```java
import java.util.Scanner;

class Solution {
  public class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
      val = x;
    }
  }

  public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
    ListNode res = new ListNode(0);
    ListNode u = l1;
    ListNode v = l2;
    ListNode cur = null;
    int t = 0;

    while (u != null || v != null || t != 0) {
      if (cur == null) {
        cur = res;
      } else {
        cur.next = new ListNode(0);
        cur = cur.next;
      }
      cur.next = null;

      cur.val = t;
      if (u != null) {
        cur.val += u.val;
        u = u.next;
      }
      if (v != null) {
        cur.val += v.val;
        v = v.next;
      }
      t = cur.val / 10;
      cur.val %= 10;
    }

    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      m = in.nextInt();

      ListNode a = new ListNode(0);
      ListNode b = new ListNode(0);
      ListNode u = a;
      ListNode v = b;

      for (int i = 0; i < n; i++) {
        if (i != 0) {
```

```
            u.next = new ListNode(0);
            u = u.next;
          }
          u.val = in.nextInt();
        }
        for (int i = 0; i < m; i++) {
          if (i != 0) {
            v.next = new ListNode(0);
            v = v.next;
          }
          v.val = in.nextInt();
        }

        solve(a, b);
      }

      in.close();
    }

    public void solve(ListNode x, ListNode y) {
      ListNode res;
      res = addTwoNumbers(x, y);
      otput(res);
    }

    public void otput(ListNode x) {
      while (x != null) {
        System.out.print(x.val);
        x = x.next;
      }
      System.out.println("");
    }

    private int n, m;
}
```

B.3 Leetcode 3

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  int lengthOfLongestSubstring(string s) {
    int res = 0;
    int sz = s.length();
    int l = 0, r = 0;
    if (sz == 0) return 0;
    int mp[256];
    memset(mp, -1, 256 * sizeof(int));

    for (int i = 0; i < sz; i++)
      if (mp[s[i]] == -1) {
        r = i;
        mp[s[i]] = i;
```

```
      } else {
        if (mp[s[i]] == -2) {
          r = i;
          mp[s[i]] = i;
        } else {
          res = max(res, r - l + 1);
          int newL = mp[s[i]] + 1;
          for (int j = l; j < mp[s[i]] + 1; j++) mp[s[j]] = -2;
          l = newL;
          r = i;
          mp[s[i]] = i;
        }
      }

    res = max(res, r - l + 1);
    return res;
  }
  void input(void) {
    while (cin >> a) {
      solve(a);
    }
  }
  void solve(string x) {
    int res;
    res = lengthOfLongestSubstring(x);
    otput(res);
  }
  void otput(int x) { printf("%d\n", x); }

 private:
  string a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Arrays;
import java.util.Scanner;

class Solution {
  public int lengthOfLongestSubstring(String s) {
    int res = 0;
    int sz = s.length();
    int l = 0, r = 0;
    if (sz == 0)
      return 0;
    int[] mp = new int[256];
    Arrays.fill(mp, -1);

    for (int i = 0; i < sz; i++)
      if (mp[s.charAt(i)] == -1) {
        r = i;
        mp[s.charAt(i)] = i;
      } else {
        if (mp[s.charAt(i)] == -2) {
```

```
            r = i;
            mp[s.charAt(i)] = i;
          } else {
            res = Math.max(res, r - l + 1);
            int newL = mp[s.charAt(i)] + 1;
            for (int j = l; j < mp[s.charAt(i)] + 1; j++)
              mp[s.charAt(j)] = -2;
            l = newL;
            r = i;
            mp[s.charAt(i)] = i;
          }
        }

      res = Math.max(res, r - l + 1);
      return res;
    }

    public static void main(String[] args) throws FileNotFoundException {
      FileInputStream fin = new FileInputStream("./assets/fipt.txt");
      PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

      System.setIn(fin);
      System.setOut(fot);

      Solution sol = new Solution();

      sol.input();
    }

    public void input() {
      Scanner in = new Scanner(System.in);
      while (in.hasNext()) {
        solve(in.nextLine());
      }

      in.close();
    }

    public void solve(String x) {
      int res;
      res = lengthOfLongestSubstring(x);
      otput(res);

    }

    public void otput(int x) {
      System.out.println(x);
    }

    private String a;
}
```

B.4 Leetcode 6

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

```

```cpp
using namespace std;

class Solution {
 public:
  string convert(string s, int numRows) {
    if (s.length() <= 1) return s;
    if (numRows == 1) return s;
    string res = "";
    vector<int> v;
    int addItem = 2 * (numRows - 1);
    int sz = s.length();
    int item = 0;
    int len;
    int l, r;
    while (item < sz) {
      v.push_back(item);
      res += s[item];
      item += addItem;
    }
    v.push_back(item);
    for (len = 1; len < numRows - 1; len++) {
      for (int i = 0; i < v.size(); i++) {
        l = v[i] - len;
        r = v[i] + len;
        if (0 <= l && l < sz) {
          res += s[l];
        }
        if (0 <= r && r < sz) {
          res += s[r];
        }
      }
    }
    for (int i = 0; i < v.size(); i++) {
      r = v[i] + numRows - 1;
      if (r < sz) {
        res += s[r];
      }
    }
    return res;
  }
  void input(void) {
    while (~scanf("%d", &n)) {
      cin >> a;

      solve(a, n);
    }
  }

  void solve(string x, int y) {
    string res;
    res = convert(x, y);
    otput(res);
  }
  void otput(string x) { cout << x << endl; }

 private:
  int n;
  string a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
```

```
}
```

Java

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.Scanner;

class Solution {
  public String convert(String s, int numRows) {
    if (s.length() <= 1)
      return s;
    if (numRows == 1)
      return s;
    String res = "";
    ArrayList<Integer> v = new ArrayList<Integer>();
    int addItem = 2 * (numRows - 1);
    int sz = s.length();
    int item = 0;
    int len;
    int l, r;
    while (item < sz) {
      v.add(item);

      res += s.charAt(item);
      item += addItem;
    }
    v.add(item);
    for (len = 1; len < numRows - 1; len++) {
      for (int i = 0; i < v.size(); i++) {
        l = v.get(i) - len;
        r = v.get(i) + len;
        if (0 <= l && l < sz) {
          res += s.charAt(l);
        }
        if (0 <= r && r < sz) {
          res += s.charAt(r);
        }
      }
    }
    for (int i = 0; i < v.size(); i++) {
      r = v.get(i) + numRows - 1;
      if (r < sz) {
        res += s.charAt(r);
      }
    }
    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();

```

```
      a = in.next();

      solve(a, n);
    }

    in.close();
  }

  public void solve(String x, int y) {
    String res;
    res = convert(x, y);
    otput(res);

  }

  public void otput(String x) {
    System.out.println(x);
  }

  private int n;
  private String a;
}
```

B.5 Leetcode 11

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  int maxArea(vector<int>& height) {
    int res = 0;
    int sz = height.size();
    int l = 0, r = sz - 1;
    while (l < r) {
      res = max(res, min(height[l], height[r]) * (r - l));
      if (height[l] < height[r])
        l++;
      else
        r--;
    }

    return res;
  }

  void input(void) {
    while (~scanf("%d", &n)) {
      int t;
      for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        a.push_back(t);
      }
      solve(a);
    }
  }
```

```cpp
  void solve(vector<int>& x) {
    int res;
    res = maxArea(x);
    otput(res);
  }
  void otput(int x) { printf("%d\n", x); }

 private:
  int n;
  vector<int> a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public int maxArea(int[] height) {
    int res = 0;
    int sz = height.length;
    int l = 0, r = sz - 1;
    while (l < r) {
      res = Math.max(res, Math.min(height[l], height[r]) * (r - l));
      if (height[l] < height[r])
        l++;
      else
        r--;
    }

    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      a = new int[n];
      for (int i = 0; i < n; i++)
        a[i] = in.nextInt();
      solve(a);
    }

```

```
    in.close();
  }

  public void solve(int[] x) {
    int res;
    res = maxArea(x);
    otput(res);

  }

  public void otput(int x) {
    System.out.println(x);
  }

  private int n;
  private int[] a;
}
```

B.6 Leetcode 15

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  vector<vector<int> > threeSum(vector<int>& nums) {
    vector<vector<int> > res;
    int sz = nums.size();
    int i, j, k;
    int r;
    if (sz < 3) return res;
    sort(nums.begin(), nums.end());

    for (i = 0; i < sz; i++) {
      if (nums[i] > 0) break;
      if (i > 0 && nums[i] == nums[i - 1]) continue;
      j = i + 1;
      k = sz - 1;
      while (j < k) {
        int tmp = nums[i] + nums[j] + nums[k];
        if (tmp < 0) {
          j++;
        } else if (tmp > 0) {
          k--;
        } else {
          res.push_back({nums[i], nums[j], nums[k]});
          j++;
          k--;
          while (j < k && nums[j] == nums[j - 1]) j++;
          while (j < k && nums[k] == nums[k + 1]) k--;
        }
      }
    }

    return res;
```

```
  }

  void input(void) {
    while (~scanf("%d", &n)) {
      int t;
      for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        a.push_back(t);
      }
      solve(a);
      a.clear();
    }
  }
  void solve(vector<int>& x) {
    vector<vector<int> > res;
    res = threeSum(x);
    otput(res);
  }
  void otput(vector<vector<int> > x) {
    for (int i = 0; i < x.size(); i++)
      printf("%d %d %d\n", x[i][0], x[i][1], x[i][2]);
    puts("");
  }

 private:
  int n;
  vector<int> a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

class Solution {
  public List<List<Integer>> threeSum(int[] nums) {
    List<List<Integer>> res = new ArrayList<>();
    Arrays.sort(nums);
    int sz = nums.length;
    for (int i = 0; i < sz; i++) {
      if (nums[i] > 0) {
        break;
      }
      if (i > 0 && nums[i - 1] == nums[i]) {
        continue;
      }
      int j = i + 1, k = sz - 1;
      while (j < k) {
        int t = nums[i] + nums[j] + nums[k];
        if (t > 0) {
          k--;
        } else if (t < 0) {
```

```
            j++;
          } else {
            res.add(Arrays.asList(nums[i], nums[j], nums[k]));
            j++;
            k--;
            while (j < k && nums[j - 1] == nums[j]) {
              j++;
            }
            while (j < k && nums[k] == nums[k + 1]) {
              k--;
            }
          }
        }
      }
      return res;
    }

    public static void main(String[] args) throws FileNotFoundException {
      FileInputStream fin = new FileInputStream("./assets/fipt.txt");
      PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

      System.setIn(fin);
      System.setOut(fot);

      Solution sol = new Solution();

      sol.input();
    }

    public void input() {
      Scanner in = new Scanner(System.in);
      while (in.hasNext()) {
        n = in.nextInt();
        a = new int[n];
        for (int i = 0; i < n; i++)
          a[i] = in.nextInt();
        solve(a);
      }

      in.close();
    }

    public void solve(int[] x) {
      List<List<Integer>> res;
      res = threeSum(x);
      otput(res);

    }

    public void otput(List<List<Integer>> x) {
      System.out.println(x);
    }

    private int n;
    private int[] a;
}
```

B.7 Leetcode 16

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
```

```cpp
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  int threeSumClosest(vector<int>& nums, int target) {
    int res = nums[0] + nums[1] + nums[2];
    sort(nums.begin(), nums.end());
    int sz = nums.size();
    for (int i = 0; i < sz; i++) {
      if (i != 0 && nums[i - 1] == nums[i]) {
        continue;
      }
      int j = i + 1, k = sz - 1;
      while (j < k) {
        int t = nums[i] + nums[j] + nums[k];
        if (t < target) {
          if (abs(target - t) < abs(target - res)) res = t;
          j++;
          while (j < k && nums[j - 1] == nums[j]) j++;
        } else if (t > target) {
          if (abs(target - t) < abs(target - res)) res = t;
          k--;
          while (j < k && nums[k] == nums[k + 1]) k--;
        } else {
          return target;
        }
      }
    }
    return res;
  }

  void input(void) {
    while (~scanf("%d %d", &n, &m)) {
      int t;
      for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        a.push_back(t);
      }
      solve(a, m);
      a.clear();
    }
  }
  void solve(vector<int>& x, int y) {
    int res;
    res = threeSumClosest(x, y);
    otput(res);
  }
  void otput(int x) { printf("%d\n", x); }

 private:
  int n, m;
  vector<int> a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Arrays;
import java.util.Scanner;

class Solution {
  public int threeSumClosest(int[] nums, int target) {
    int res = nums[0] + nums[1] + nums[2];
    Arrays.sort(nums);
    int sz = nums.length;
    for (int i = 0; i < sz; i++) {
      if (i != 0 && nums[i - 1] == nums[i]) {
        continue;
      }
      int j = i + 1, k = sz - 1;
      while (j < k) {
        int t = nums[i] + nums[j] + nums[k];
        if (t < target) {
          if (Math.abs(target - t) < Math.abs(target - res))
            res = t;
          j++;
          while (j < k && nums[j - 1] == nums[j])
            j++;
        } else if (t > target) {
          if (Math.abs(target - t) < Math.abs(target - res))
            res = t;
          k--;
          while (j < k && nums[k] == nums[k + 1])
            k--;
        } else {
          return target;
        }
      }
    }
    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      a = new int[n];
      for (int i = 0; i < n; i++)
        a[i] = in.nextInt();
      solve(a);
    }

    in.close();
  }

  public void solve(int[] x, int y) {
```

```
    int res;
    res = threeSumClosest(x, y);
    otput(res);

  }

  public void otput(int x) {
    System.out.println(x);
  }

  private int n;
  private int[] a;
}
```

B.8 Leetcode 19

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

typedef struct ListNode {
  int val;
  ListNode* next;
  ListNode(int x) : val(x), next(NULL) {}
} ListNode;

class Solution {
 public:
  int deleteNode(ListNode* x, int target) {
    if (x == NULL) return 0;

    int u = deleteNode(x->next, target);

    if (u != -1) {
      if (u == target) {
        ListNode* y = x->next->next;
        delete x->next;
        x->next = y;
      } else {
        return u + 1;
      }
    }

    return -1;
  }
  ListNode* removeNthFromEnd(ListNode* head, int n) {
    int u = deleteNode(head, n);
    if (u != -1) {
      ListNode* t = head;
      head = head->next;
      delete t;
    }

    return head;
  }

```

```
  void input(void) {
    while (~scanf("%d %d", &n, &m)) {
      a = new ListNode(0);
      ListNode* u = a;

      for (int i = 0; i < m; i++) {
        if (i != 0) {
          u->next = new ListNode(0);
          u = u->next;
        }

        scanf("%d", &u->val);
      }

      solve(a, n);
    }
  }
  void solve(ListNode* x, int y) {
    ListNode* res;
    res = removeNthFromEnd(x, y);
    otput(res);
  }
  void otput(ListNode* x) {
    ListNode* cur = x;
    while (cur != NULL) {
      printf("%d", cur->val);
      cur = cur->next;
    }
  }

 private:
  int n, m;
  ListNode* a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
      val = x;
    }
  }

  public int deleteNode(ListNode x, int target) {
    if (x == null)
      return 0;

    int u = deleteNode(x.next, target);
```

```java

    if (u != -1) {
      if (u == target) {
        ListNode y = x.next.next;
        x.next = y;
      } else {
        return u + 1;
      }
    }

    return -1;
  }

  public ListNode removeNthFromEnd(ListNode head, int n) {
    int u = deleteNode(head, n);
    if (u != -1) {
      head = head.next;
    }
    return head;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      m = in.nextInt();

      ListNode a = new ListNode(0);
      ListNode u = a;

      for (int i = 0; i < m; i++) {
        if (i != 0) {
          u.next = new ListNode(0);
          u = u.next;
        }
        u.val = in.nextInt();
      }

      solve(a, n);
    }

    in.close();
  }

  public void solve(ListNode x, int y) {
    ListNode res;
    res = removeNthFromEnd(x, y);
    otput(res);
  }

  public void otput(ListNode x) {
    while (x != null) {
      System.out.print(x.val);
      x = x.next;
    }
    System.out.println("");
  }
```

```

  private int n, m;
}
```

B.9 Leetcode 25

C++

```cpp
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

typedef struct ListNode {
  int val;
  ListNode* next;
  ListNode(int x) : val(x), next(NULL) {}
} ListNode;

class Solution {
 public:
  void reverseGroup(ListNode* u, ListNode* v) {
    if (u != v) {
      if (u->next == v) {
        v->next = u;
      } else {
        ListNode* t = u->next;
        reverseGroup(u->next, v);
        t->next = u;
      }
    }
  }
  ListNode* reverseKGroup(ListNode* head, int k) {
    if (k == 1) return head;

    ListNode* fakeHead = new ListNode(0);
    fakeHead->next = head;
    ListNode* pre = fakeHead;
    ListNode* cur = head;
    ListNode* u = NULL;
    ListNode* t = NULL;
    ListNode* v = NULL;
    int cnt = 0;
    while (cur != NULL) {
      cnt++;
      if (cnt == k) {
        u = pre->next;
        v = cur;
        t = cur->next;
        reverseGroup(u, v);
        pre->next = v;
        u->next = t;
        pre = u;
        cur = u;
        cnt = 0;
      }
      cur = cur->next;
    }
    return fakeHead->next;
```

```
  }
  void input(void) {
    while (~scanf("%d %d", &n, &m)) {
      a = new ListNode(0);
      ListNode* u = a;

      for (int i = 0; i < m; i++) {
        if (i != 0) {
          u->next = new ListNode(0);
          u = u->next;
        }

        scanf("%d", &u->val);
      }

      solve(a, n);
    }
  }
  void solve(ListNode* x, int y) {
    ListNode* res;
    res = reverseKGroup(x, y);
    otput(res);
  }
  void otput(ListNode* x) {
    ListNode* cur = x;
    while (cur != NULL) {
      printf("%d", cur->val);
      cur = cur->next;
    }
    cout << endl;
  }

 private:
  int n, m;
  ListNode* a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
      val = x;
    }
  }

  void reverseGroup(ListNode u, ListNode v) {
    if (u != v) {
      if (u.next == v) {
```

```
        v.next = u;
      } else {
        ListNode t = u.next;
        reverseGroup(u.next, v);
        t.next = u;
      }
    }
  }

  public ListNode reverseKGroup(ListNode head, int k) {
    if (k == 1)
      return head;
    ListNode fakeHead = new ListNode(0);
    fakeHead.next = head;
    ListNode pre = fakeHead;
    ListNode cur = head;
    ListNode u = null;
    ListNode t = null;
    ListNode v = null;
    int cnt = 0;
    while (cur != null) {
      cnt++;
      if (cnt == k) {
        u = pre.next;
        v = cur;
        t = cur.next;
        reverseGroup(u, v);
        pre.next = v;
        u.next = t;
        pre = u;
        cur = u;
        cnt = 0;
      }
      cur = cur.next;
    }
    return fakeHead.next;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      m = in.nextInt();

      ListNode a = new ListNode(0);
      ListNode u = a;

      for (int i = 0; i < m; i++) {
        if (i != 0) {
          u.next = new ListNode(0);
          u = u.next;
        }
        u.val = in.nextInt();
      }

      solve(a, n);
    }
```

```

    in.close();
  }

  public void solve(ListNode x, int y) {
    ListNode res;
    res = reverseKGroup(x, y);
    otput(res);
  }

  public void otput(ListNode x) {
    while (x != null) {
      System.out.print(x.val);
      x = x.next;
    }
    System.out.println("");
  }

  private int n, m;
}
```

B.10 Leetcode 26

C++

```cpp
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  int removeDuplicates(vector<int>& nums) {
    int res = 0;
    int sz = nums.size();
    if (sz == 0) return 0;
    int cur = 0;
    for (int i = 0; i < sz; i++)
      if (nums[cur] != nums[i]) {
        cur++;
        nums[cur] = nums[i];
      }
    res = cur + 1;
    return res;
  }

  void input(void) {
    while (~scanf("%d", &n)) {
      int t;
      for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        a.push_back(t);
      }
      solve(a);
      a.clear();
    }
  }
  void solve(vector<int>& x) {
    int res;
```

```
    res = removeDuplicates(x);
    otput(res);
  }
  void otput(int x) { printf("%d\n", x); }

 private:
  int n;
  vector<int> a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public int removeDuplicates(int[] nums) {
    int res = 0;
    int sz = nums.length;
    if (sz == 0)
      return 0;
    int cur = 0;
    for (int i = 0; i < sz; i++)
      if (nums[cur] != nums[i]) {
        cur++;
        nums[cur] = nums[i];
      }
    res = cur + 1;
    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      a = new int[n];
      for (int i = 0; i < n; i++)
        a[i] = in.nextInt();
      solve(a);
    }

    in.close();
  }
```

```

  public void solve(int[] x) {
    int res;
    res = removeDuplicates(x);
    otput(res);

  }

  public void otput(int x) {
    System.out.println(x);
  }

  private int n;
  private int[] a;
}
```

B.11 Leetcode 42

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  int trap(vector<int>& height) {
    int res = 0;
    int sz = height.size();
    if (sz == 0) return 0;

    vector<int> lmx(sz, 0), rmx(sz, 0);

    for (int i = 1, j = sz - 2; i < sz; i++, j--) {
      lmx[i] = max(lmx[i - 1], height[i - 1]);
      rmx[j] = max(rmx[j + 1], height[j + 1]);
    }
    for (int i = 1; i < sz - 1; i++)
      res += max(0, min(lmx[i], rmx[i]) - height[i]);
    return res;
  }

  void input(void) {
    while (~scanf("%d", &n)) {
      int t;
      for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        a.push_back(t);
      }
      solve(a);
      a.clear();
    }
  }
  void solve(vector<int>& x) {
    int res;
    res = trap(x);
    otput(res);
  }
```

```cpp
  void otput(int x) { printf("%d\n", x); }

 private:
  int n;
  vector<int> a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public int trap(int[] height) {
    int res = 0;
    int sz = height.length;
    if (sz == 0)
      return 0;

    int[] lmx = new int[sz];
    int[] rmx = new int[sz];

    for (int i = 1, j = sz - 2; i < sz; i++, j--) {
      lmx[i] = Math.max(lmx[i - 1], height[i - 1]);
      rmx[j] = Math.max(rmx[j + 1], height[j + 1]);
    }
    for (int i = 1; i < sz - 1; i++)
      res += Math.max(0, Math.min(lmx[i], rmx[i]) - height[i]);
    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      a = new int[n];
      for (int i = 0; i < n; i++)
        a[i] = in.nextInt();
      solve(a);
    }

    in.close();
  }
```

```
50
51    public void solve(int[] x) {
52      int res;
53      res = trap(x);
54      otput(res);
55
56    }
57
58    public void otput(int x) {
59      System.out.println(x);
60    }
61
62    private int n;
63    private int[] a;
64  }
```

B.12 Leetcode 56

C++

```
1   #include <algorithm>
2   #include <cstdio>
3   #include <cstdlib>
4   #include <cstring>
5   #include <iostream>
6   #include <map>
7   #include <queue>
8   #include <stack>
9   #include <string>
10  #include <vector>
11
12  using namespace std;
13
14  class Solution {
15   public:
16
17    vector<vector<int> > merge(vector<vector<int> >& intervals) {
18      vector<vector<int> > res;
19      sort(intervals.begin(), intervals.end());
20
21      for (int i = 0; i < intervals.size(); i++) {
22        if (res.empty()) {
23          res.push_back(intervals[i]);
24        } else {
25          if (res.back()[1] < intervals[i][0]) {
26            res.push_back(intervals[i]);
27          } else {
28            if (res.back()[1] < intervals[i][1]) res.back()[1] = intervals[i][1];
29          }
30        }
31      }
32
33      return res;
34    }
35    void input(void) {
36      while (~scanf("%d", &n)) {
37        int u, v;
38        for (int i = 0; i < n; i++) {
39          vector<int> t;
40          scanf("%d %d", &u, &v);
41
42          t.push_back(u);
43          t.push_back(v);
44          a.push_back(t);
45        }
46
47        solve(a);
```

```
      a.clear();
    }
  }
  void solve(vector<vector<int> >& x) {
    vector<vector<int> > res;
    res = merge(x);
    otput(res);
  }
  void otput(vector<vector<int> >& x) {
    vector<vector<int> > cur = x;
    for (int i = 0; i < cur.size(); i++)
      printf("%d %d\n", cur[i][0], cur[i][1]);
  }

 private:
  int n;
  vector<vector<int> > a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

class Solution {
  public int[][] merge(int[][] intervals) {
    ArrayList<int[]> res = new ArrayList<int[]>();
    Arrays.sort(intervals, new Comparator<int[]>() {
      @Override
      public int compare(int[] l, int[] r) {
        return l[0] - r[0];
      }
    });
    int sz = intervals.length;
    for (int i = 0; i < sz; i++) {
      int l = intervals[i][0];
      int r = intervals[i][1];
      while (i < sz - 1 && intervals[i + 1][0] <= r) {
        r = Math.max(r, intervals[i + 1][1]);
        i++;
      }
      res.add(new int[] { l, r });
    }

    return res.toArray(new int[res.size()][2]);
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
```

```
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();

      vals = new int[n][2];

      for (int i = 0; i < n; i++) {
        vals[i][0] = in.nextInt();
        vals[i][1] = in.nextInt();
      }

      solve(vals);
    }

    in.close();
  }

  public void solve(int[][] x) {
    int[][] res;
    res = merge(x);
    otput(res);

  }

  public void otput(int[][] x) {
    int sz = x.length;
    for (int i = 0; i < sz; i++)
      System.out.println(x[i][0] + " " + x[i][1]);
  }

  private int n;
  private int[][] vals;
}
```

B.13 Leetcode 61

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

typedef struct ListNode {
  int val;
  ListNode* next;
  ListNode(int x) : val(x), next(NULL) {}
} ListNode;

class Solution {
 public:
```

```
  ListNode* rotateRight(ListNode* head, int k) {
    if (head == NULL) return head;
    ListNode* res = head;
    ListNode* cur = head;
    ListNode* pre = NULL;
    ListNode* st = NULL;
    ListNode* en = NULL;
    vector<ListNode*> v;
    while (cur != NULL) {
      v.push_back(cur);

      if (cur->next == NULL) en = cur;
      cur = cur->next;
    }

    int lenOfList = v.size();
    int mk = k % lenOfList;
    if (mk != 0) {
      pre = v[lenOfList - mk - 1];
      st = v[lenOfList - mk];
      pre->next = NULL;
      en->next = res;
      res = st;
    }

    return res;
  }

  void input(void) {
    while (~scanf("%d %d", &n, &m)) {
      a = new ListNode(0);
      ListNode* u = a;

      for (int i = 0; i < m; i++) {
        if (i != 0) {
          u->next = new ListNode(0);
          u = u->next;
        }

        scanf("%d", &u->val);
      }

      solve(a, n);
    }
  }
  void solve(ListNode* x, int y) {
    ListNode* res;
    res = rotateRight(x, y);
    otput(res);
  }
  void otput(ListNode* x) {
    ListNode* cur = x;
    while (cur != NULL) {
      printf("%d", cur->val);
      cur = cur->next;
    }
    cout << endl;
  }

 private:
  int n, m;
  ListNode* a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;
```

```

  sol.input();

  return 0;
}
```

Java

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.LinkedList;
import java.util.Scanner;

class Solution {
  public class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
      val = x;
    }
  }

  public ListNode rotateRight(ListNode head, int k) {
    if (head == null)
      return head;
    ListNode res = head;
    ListNode cur = head;
    ListNode pre = null;
    ListNode st = null;
    ListNode en = null;
    LinkedList<ListNode> v = new LinkedList<ListNode>();
    while (cur != null) {
      v.add(cur);

      if (cur.next == null)
        en = cur;
      cur = cur.next;
    }

    int lenOfList = v.size();
    int mk = k % lenOfList;
    if (mk != 0) {
      pre = v.get(lenOfList - mk - 1);
      st = v.get(lenOfList - mk);
      pre.next = null;
      en.next = res;
      res = st;
    }

    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
```

```
    while (in.hasNext()) {
      n = in.nextInt();
      m = in.nextInt();

      ListNode a = new ListNode(0);
      ListNode u = a;

      for (int i = 0; i < m; i++) {
        if (i != 0) {
          u.next = new ListNode(0);
          u = u.next;
        }
        u.val = in.nextInt();
      }

      solve(a, n);
    }

    in.close();
  }

  public void solve(ListNode x, int y) {
    ListNode res;
    res = rotateRight(x, y);
    otput(res);
  }

  public void otput(ListNode x) {
    while (x != null) {
      System.out.print(x.val);
      x = x.next;
    }
    System.out.println("");
  }

  private int n, m;
}
```

B.14 Leetcode 121

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  int maxProfit(vector<int>& prices) {
    int res = 0;
    int rmx = 0;
    for (int i = prices.size() - 2; i >= 0; i--) {
      rmx = max(rmx, prices[i + 1]);
      res = max(res, max(0, (rmx - prices[i])));
    }
    return res;
  }

```

```
  void input(void) {
    while (~scanf("%d", &n)) {
      int t;
      for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        a.push_back(t);
      }
      solve(a);
      a.clear();
    }
  }
  void solve(vector<int>& x) {
    int res;
    res = maxProfit(x);
    otput(res);
  }
  void otput(int x) { printf("%d\n", x); }

 private:
  int n;
  vector<int> a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public int maxProfit(int[] prices) {
    int res = 0;
    int rmx = 0;
    for (int i = prices.length - 2; i >= 0; i--) {
      rmx = Math.max(rmx, prices[i + 1]);
      res = Math.max(res, Math.max(0, (rmx - prices[i])));
    }
    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
```

```
      a = new int[n];
      for (int i = 0; i < n; i++)
        a[i] = in.nextInt();
      solve(a);
    }

    in.close();
  }

  public void solve(int[] x) {
    int res;
    res = maxProfit(x);
    otput(res);

  }

  public void otput(int x) {
    System.out.println(x);
  }

  private int n;
  private int[] a;
}
```

B.15 Leetcode 138

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Node {
 public:
  int val;
  Node* next;
  Node* random;

  Node(int _val) {
    val = _val;
    next = NULL;
    random = NULL;
  }
};
class Solution {
 public:
  Node* copyRandomList(Node* head) {
    if (head == NULL) return NULL;

    Node* res = new Node(head->val);
    Node* cur_h = head;
    Node* cur_r = res;
    map<Node*, Node*> mp;
    mp.insert(pair<Node*, Node*>(NULL, NULL));

    while (cur_h != NULL) {
      if (cur_h != head) {
        cur_r->next = new Node(cur_h->val);
```

```
      cur_r = cur_r->next;
    }
    mp.insert(pair<Node*, Node*>(cur_h, cur_r));
    cur_h = cur_h->next;
  }

  cur_h = head;
  cur_r = res;
  while (cur_h != NULL) {
    cur_r->random = mp[cur_h->random];
    cur_r = cur_r->next;

    cur_h = cur_h->next;
  }

  return res;
}
void input(void) {
  while (~scanf("%d", &n)) {
    a = new Node(0);
    Node* u = a;
    int x;
    vector<int> v;
    vector<Node*> record;

    for (int i = 0; i < n; i++) {
      if (i != 0) {
        u->next = new Node(0);
        u = u->next;
      }

      scanf("%d %d", &u->val, &x);
      v.push_back(x);
      record.push_back(u);
    }
    for (int i = 0; i < n; i++) {
      if (i + 1 < n) {
        record[i]->next = record[i + 1];
      }
      if (v[i] == 11111) {
        continue;
      }
      record[i]->random = record[v[i]];
    }

    solve(a);
  }
}
void solve(Node* x) {
  Node* res;
  res = copyRandomList(x);
  otput(res);
}
void otput(Node* x) {
  Node* cur = x;
  int cnt = 0;
  map<Node*, int> mp;
  while (cur != NULL) {
    mp.insert(pair<Node*, int>(cur, cnt));
    cnt++;
    cur = cur->next;
  }
  cur = x;
  while (cur != NULL) {
    printf("%d ", cur->val);
    if (cur->random == NULL)
      printf("null\n");
    else
      printf("%d\n", mp[cur->random]);
```

```
109
110       cur = cur->next;
111     }
112   }
113
114  private:
115   int n;
116   Node* a;
117 };
118
119 int main() {
120   freopen("./assets/fipt.txt", "r", stdin);
121   freopen("./assets/fopt.txt", "w", stdout);
122
123   Solution sol;
124
125   sol.input();
126
127   return 0;
128 }
```

Java

```
1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.HashMap;
6  import java.util.LinkedList;
7  import java.util.Map;
8  import java.util.Scanner;
9
10 class Node {
11   int val;
12   Node next;
13   Node random;
14
15   public Node(int val) {
16     this.val = val;
17     this.next = null;
18     this.random = null;
19   }
20 }
21
22 class Solution {
23   public Node copyRandomList(Node head) {
24     if (head == null)
25       return null;
26
27     Node res = new Node(head.val);
28     Node cur_h = head;
29     Node cur_r = res;
30     Map<Node, Node> mp = new HashMap<Node, Node>();
31     mp.put(null, null);
32
33     while (cur_h != null) {
34       if (cur_h != head) {
35         cur_r.next = new Node(cur_h.val);
36         cur_r = cur_r.next;
37       }
38       mp.put(cur_h, cur_r);
39       cur_h = cur_h.next;
40     }
41
42     cur_h = head;
43     cur_r = res;
44     while (cur_h != null) {
45       cur_r.random = mp.get(cur_h.random);
46       cur_r = cur_r.next;
```

```
47
48        cur_h = cur_h.next;
49      }
50
51      return res;
52    }
53
54    public static void main(String[] args) throws FileNotFoundException {
55      FileInputStream fin = new FileInputStream("./assets/fipt.txt");
56      PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
57
58      System.setIn(fin);
59      System.setOut(fot);
60
61      Solution sol = new Solution();
62
63      sol.input();
64    }
65
66    public void input() {
67      Scanner in = new Scanner(System.in);
68      while (in.hasNext()) {
69        n = in.nextInt();
70
71        Node a = new Node(0);
72        Node u = a;
73        LinkedList<Integer> v = new LinkedList<Integer>();
74        LinkedList<Node> record = new LinkedList<Node>();
75
76        for (int i = 0; i < n; i++) {
77          if (i != 0) {
78            u.next = new Node(0);
79            u = u.next;
80          }
81          u.val = in.nextInt();
82
83          v.add(in.nextInt());
84          record.add(u);
85        }
86        for (int i = 0; i < n; i++) {
87          if (i + 1 < n) {
88            record.get(i).next = record.get(i + 1);
89          }
90          if (v.get(i) == 11111) {
91            continue;
92          }
93          record.get(i).random = record.get(v.get(i));
94        }
95
96        solve(a);
97      }
98
99      in.close();
100   }
101
102   public void solve(Node x) {
103     Node res;
104     res = copyRandomList(x);
105     otput(res);
106
107   }
108
109   public void otput(Node x) {
110     Node cur = x;
111     int cnt = 0;
112     Map<Node, Integer> mp = new HashMap<Node, Integer>();
113     while (cur != null) {
114       mp.put(cur, cnt);
115       cnt++;
```

```
      cur = cur.next;
    }
    cur = x;
    while (cur != null) {
      System.out.print(cur.val + " ");
      if (cur.random == null)
        System.out.println("null");
      else
        System.out.println(mp.get(cur.random));

      cur = cur.next;
    }

  }

  private int n, m;
}
```

B.16 Leetcode 141

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

typedef struct ListNode {
  int val;
  ListNode* next;
  ListNode(int x) : val(x), next(NULL) {}
} ListNode;

class Solution {
 public:
  bool hasCycle(ListNode* head) {
    if (NULL == head) return false;
    ListNode *slow = head, *fast = head->next;
    while (NULL != fast) {
      if (slow == fast) return true;
      slow = slow->next;
      fast = fast->next;
      if (NULL != fast) fast = fast->next;
    }
    return false;
  }

  void input(void) {
    while (~scanf("%d %d", &n, &m)) {
      int t;

      scanf("%d", &t);
      a = new ListNode(t);
      ListNode* cur = a;

      for (int i = 1; i < n; i++) {
        scanf("%d", &t);
        cur->next = new ListNode(t);
        cur = cur->next;
```

```
      }
      ListNode* back = cur;
      cur = a;
      if (m >= 0) {
        for (int i = 0; i < m - 1; i++) cur = cur->next;
        back->next = cur;
      }

      solve(a);
    }
  }

  void solve(ListNode* x) {
    bool res;
    res = hasCycle(x);
    otput(res);
  }
  void otput(int x) { printf("%d\n", x); }

 private:
  int n, m;
  ListNode* a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
      val = x;
      next = null;
    }
  }

  public boolean hasCycle(ListNode head) {
    if (null == head)
      return false;
    ListNode slow = head, fast = head.next;
    while (null != fast) {
      if (slow == fast)
        return true;
      slow = slow.next;
      fast = fast.next;
      if (null != fast)
        fast = fast.next;
    }
    return false;
  }

```

```java
  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      m = in.nextInt();
      a = new ListNode(0);

      int cnt = 0;
      for (int i = 0; i < n; i++) {
        if (cnt == 0) {
          a.val = in.nextInt();
        } else {
          a.next = new ListNode(in.nextInt());
        }
      }

      solve(a);
    }

    in.close();
  }

  public void solve(ListNode x) {
    boolean res;
    res = hasCycle(x);
    otput(res);

  }

  public void otput(boolean x) {
    System.out.println(x);
  }

  private int n, m;
  private ListNode a;
}
```

B.17 Leetcode 202

C++

```cpp
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
```

```
 public:
  int cal(int x) {
    int res = 0;
    int t;
    while (x != 0) {
      t = x % 10;
      x /= 10;
      res += t * t;
    }
    return res;
  }
  bool isHappy(int n) {
    if (n == 1) return true;
    int slow = n, fast = cal(n);
    while (slow != fast) {
      if (slow == 1 || fast == 1) return true;
      slow = cal(slow);
      fast = cal(cal(fast));
    }
    return false;
  }

  void input(void) {
    while (~scanf("%d", &n)) {
      solve(n);
    }
  }

  void solve(int x) {
    bool res;
    res = isHappy(x);
    otput(res);
  }
  void otput(int x) { printf("%d\n", x); }

 private:
  int n, m;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public int cal(int x) {
    int res = 0;
    int t;
    while (x != 0) {
      t = x % 10;
      x /= 10;
      res += t * t;
    }
    return res;
  }
```

```java

  public boolean isHappy(int n) {
    if (n == 1)
      return true;
    int slow = n, fast = cal(n);
    while (slow != fast) {
      if (slow == 1 || fast == 1)
        return true;
      slow = cal(slow);
      fast = cal(cal(fast));
    }
    return false;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      solve(n);
    }

    in.close();
  }

  public void solve(int x) {
    boolean res;
    res = isHappy(x);
    otput(res);

  }

  public void otput(boolean x) {
    System.out.println(x);
  }

  private int n, m;
  private ListNode a;
}
```

B.18 Leetcode 206

C++

```cpp
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;
```

```cpp

typedef struct ListNode {
  int val;
  ListNode* next;
  ListNode(int x) : val(x), next(NULL) {}
} ListNode;

class Solution {
 public:
  ListNode* reverseList(ListNode* head) {
    if (head == NULL) return NULL;

    ListNode* st = head;
    ListNode* en = NULL;
    ListNode* cur = head;
    while (cur->next != NULL) {
      cur = cur->next;
    }
    en = cur;
    while (st != en) {
      cur = st->next;
      st->next = en->next;
      en->next = st;
      st = cur;
    }
    return en;
  }
  void input(void) {
    while (~scanf("%d", &n)) {
      a = new ListNode(0);
      ListNode* u = a;
      int x;

      for (int i = 0; i < n; i++) {
        if (i != 0) {
          u->next = new ListNode(0);
          u = u->next;
        }

        scanf("%d", &u->val);
      }

      solve(a);
    }
  }
  void solve(ListNode* x) {
    ListNode* res;
    res = reverseList(x);
    otput(res);
  }
  void otput(ListNode* x) {
    ListNode* cur = x;
    while (cur != NULL) {
      printf("%d\n", cur->val);
      cur = cur->next;
    }
  }

 private:
  int n;
  ListNode* a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

```

```
  sol.input();

  return 0;
}
```

Java

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
      val = x;
    }
  }

  public ListNode reverseList(ListNode head) {
    if (head == null)
      return null;

    ListNode st = head;
    ListNode en = null;
    ListNode cur = head;
    while (cur.next != null) {
      cur = cur.next;
    }
    en = cur;
    while (st != en) {
      cur = st.next;
      st.next = en.next;
      en.next = st;
      st = cur;
    }
    return en;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();

      ListNode a = new ListNode(0);
      ListNode u = a;

      for (int i = 0; i < n; i++) {
        if (i != 0) {
          u.next = new ListNode(0);
          u = u.next;
        }
        u.val = in.nextInt();
```

```
      }

      solve(a);
    }

    in.close();
  }

  public void solve(ListNode x) {
    ListNode res;
    res = reverseList(x);
    otput(res);

  }

  public void otput(ListNode x) {
    ListNode cur = x;
    while (cur != null) {
      System.out.println(cur.val);
      cur = cur.next;
    }

  }

  private int n, m;
}
```

B.19 Leetcode 209

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  int minSubArrayLen(int s, vector<int>& nums) {
    int res = nums.size();
    int sz = res;
    int sum = 0;
    int u = 0, v = 0;
    bool occur = false;

    for (v = 0; v < sz; v++) {
      sum += nums[v];
      while (sum >= s && u <= v) {
        occur = true;
        res = min(res, v - u + 1);
        sum -= nums[u];
        u++;
      }
    }
    if (occur == false) return 0;

    return res;
  }

```

```
  void input(void) {
    while (~scanf("%d %d", &n, &m)) {
      int t;
      for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        a.push_back(t);
      }
      solve(m, a);
      a.clear();
    }
  }
  void solve(int s, vector<int>& x) {
    int res;
    res = minSubArrayLen(s, x);
    otput(res);
  }
  void otput(int x) { printf("%d\n", x); }

 private:
  int n, m;
  vector<int> a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public int minSubArrayLen(int s, int[] nums) {
    int res = nums.length;
    int sz = res;
    int sum = 0;
    int u = 0, v = 0;
    boolean occur = false;

    for (v = 0; v < sz; v++) {
      sum += nums[v];
      while (sum >= s && u <= v) {
        occur = true;
        res = Math.min(res, v - u + 1);
        sum -= nums[u];
        u++;
      }
    }
    if (occur == false)
      return 0;

    return res;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

```

```
    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      m = in.nextInt();
      a = new int[n];
      for (int i = 0; i < n; i++)
        a[i] = in.nextInt();
      solve(m, a);
    }

    in.close();
  }

  public void solve(int s, int[] x) {
    int res;
    res = minSubArrayLen(s, x);
    otput(res);

  }

  public void otput(int x) {
    System.out.println(x);
  }

  private int n, m;
  private int[] a;
}
```

B.20 Leetcode 387

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

class Solution {
 public:
  int firstUniqChar(string s) {
    int ans = -1;
    int u, v;
    for (char ch = 'a'; ch <= 'z'; ch++) {
      u = s.find(ch);
      v = s.rfind(ch);
      if (u == v && u != -1) {
        if (ans == -1) {
          ans = u;
        } else {
          if (u < ans) ans = u;
```

```
        }
      }
    }
    return ans;
  }
  void input(void) {
    while (cin >> str) {
      solve(str);
    }
  }
  void solve(string s) {
    int res;
    res = firstUniqChar(s);
    otput(res);
  }
  void otput(int id) { printf("%d\n", id); }

 private:
  int n, m, t;
  string str;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public int firstUniqChar(String s) {
    int ans = -1;
    int u, v;
    for (char ch = 'a'; ch <= 'z'; ch++) {
      u = s.indexOf(ch);
      v = s.lastIndexOf(ch);
      if (u == v && u != -1) {
        if (ans == -1) {
          ans = u;
        } else {
          if (u < ans)
            ans = u;
        }
      }
    }
    return ans;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

```

```
    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      String str = in.nextLine();
      solve(str);
    }

    in.close();
  }

  public void solve(String s) {
    int res;
    res = firstUniqChar(s);
    otput(res);
  }

  public void otput(int idx) {
    System.out.println(idx);
  }

  private int n, m, t;
  private int u, v;
  private int[] numbers = new int[10000];
}
```

B.21 Leetcode 876

C++

```
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using namespace std;

typedef struct ListNode {
  int val;
  ListNode* next;
  ListNode(int x) : val(x), next(NULL) {}
} ListNode;

class Solution {
 public:
 public:
  ListNode* middleNode(ListNode* head) {
    if (NULL == head) return NULL;
    ListNode* slow = head;
    ListNode* fast = head->next;
    while (fast != NULL) {
      slow = slow->next;
      fast = fast->next;
      if (fast == NULL) {
        return slow;
      } else {
        fast = fast->next;
      }
    }
```

```
    return slow;
  }

  void input(void) {
    while (~scanf("%d", &n)) {
      a = new ListNode(0);
      ListNode* cur = a;
      for (int i = 0; i < n; i++) {
        int t;
        scanf("%d", &t);
        if (i == 0) {
          cur->val = t;
        } else {
          cur->next = new ListNode(t);
          cur = cur->next;
        }
      }
      solve(a);
    }
  }

  void solve(ListNode* x) {
    ListNode* res;
    res = middleNode(x);
    otput(res);
  }
  void otput(ListNode* x) {
    ListNode* cur = x;
    while (cur != NULL) {
      printf("%d\n", cur->val);
      cur = cur->next;
    }
  }

 private:
  int n, m;
  ListNode* a;
};

int main() {
  freopen("./assets/fipt.txt", "r", stdin);
  freopen("./assets/fopt.txt", "w", stdout);

  Solution sol;

  sol.input();

  return 0;
}
```

Java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

class Solution {
  public class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
      val = x;
    }
  }

  public ListNode middleNode(ListNode head) {
```

```java
    if (null == head)
      return null;
    ListNode slow = head;
    ListNode fast = head.next;
    while (fast != null) {
      slow = slow.next;
      fast = fast.next;
      if (fast == null) {
        return slow;
      } else {
        fast = fast.next;
      }
    }
    return slow;
  }

  public static void main(String[] args) throws FileNotFoundException {
    FileInputStream fin = new FileInputStream("./assets/fipt.txt");
    PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));

    System.setIn(fin);
    System.setOut(fot);

    Solution sol = new Solution();

    sol.input();
  }

  public void input() {
    Scanner in = new Scanner(System.in);
    while (in.hasNext()) {
      n = in.nextInt();
      a = new ListNode(0);
      ListNode cur = a;
      for (int i = 0; i < n; i++) {
        if (i == 0) {
          cur.val = in.nextInt();
        } else {
          cur.next = new ListNode(in.nextInt());
          cur = cur.next;
        }
      }
      solve(a);
    }

    in.close();
  }

  public void solve(ListNode x) {
    ListNode res;
    res = middleNode(x);
    otput(res);

  }

  public void otput(ListNode x) {
    ListNode cur = x;
    while (cur != null) {
      System.out.println(cur.val);
      cur = cur.next;
    }
  }

  private int n, m;
  private ListNode a;
}
```