Tiết kiệm điện  
Văn phòng APS hiện tại đang có N bóng đèn và có K khóa. Các khóa được nối vào các bóng đèn theo quy luật như sau:  
Khóa thứ K sẽ được nối vào các bóng đèn thứ K+n(K+1) (n>=0; 0<=K+n(K+1) <=N).  
Ví dụ:  
Khóa thứ 1 sẽ nối vào các bóng đèn số 1, 3, 5, 7,9…  
Khóa thứ 2 sẽ nối vào các bóng đèn số 2, 5, 8, 11,…  
Khóa thứ 3 sẽ nối vào các bóng đèn số 3, 7, 11, 15,…  
Nếu khóa thứ k được thay đổi trạng thái, tất cả các bóng đèn đang nối với khóa k sẽ chuyển trạng thái (Bật->Tắt, tắt->bật).  
Có một quy định được ban hành trong APS P: Nhân viên rời văn phòng cuối cùng phải có trách nhiệm tắt đèn trong văn phòng sao cho số lượng bóng đèn được tắt là lớn nhất. Tuy nhiên để thử thách tư duy của các nhân viên, nhân viên rời văn phòng cuối đó chỉ được phép thay đổi trạng thái khóa tối đa 3 lần, mỗi lần được chọn tối đa 1 khóa.  
Hãy tạo một chương trình giúp các nhân viên tìm ra phương án tối ưu để tắt đèn với tối đa 3 lần tác động vào khóa^.^  
Input:  
Dòng 1 chứa số T là số TC.  
Mỗi testcase bao gồm các thông tin sau:  
- Dòng đầu tiên chứa số lượng bóng đèn N và số Khóa K (10<=N<=100, 3<=K<=10);  
- Dòng tiếp theo là N bóng đèn, mỗi bóng đèn nhận 1 trong 2 trạng thái: 1 là bật, 0 là tắt.  
OutPut:  
In ra số lượng bóng đèn tối đa có thể tắt tại mỗi case.  
Đáp số của mỗi TC in ra trên 1 dòng theo định dạng:  
CaseTC# Answer  
Ví dụ:  
Input  
1  
10 3 (10 bóng đèn và 3 khóa)  
0 0 1 1 1 1 0 0 1 0 (lần lượt là 10 trạng thái của 10 bóng đèn, 1 là đang bật, 0 là đang tắt)  
Output:  
Case1 6  
Giải thích: Chọn tác động vào công tắc thứ 1, các bóng đèn sau sẽ bị thay đổi trạng thái:1 3 5 7 9  
Stt bóng đèn 1 2 3 4 5 6 7 8 9 10  
Ban đầu 0 0 1 1 1 1 0 0 1 0  
Sau khi tác động vào khóa 1: 1 0 0 1 0 1 1 0 0 0

package Tietkiemdien;  
  
import java.io.FileInputStream;  
import java.util.Scanner;  
  
public class NewTietkiemdien {  
static int T;  
static int K, N;  
static int[] A;  
static int[] B;  
static int[] set;  
static int[] visit;  
static int max;  
static int n;  
static int tmp;  
static void BT(int indx, int dem){  
if(dem > max) max = dem;  
if(indx == 3){  
return;  
} else {  
for(int k = 1; k <= K; k++){  
if(visit[k] == 0){  
int sl = 0;  
visit[k] = 1;  
n = 0;  
while((k + n \* (k +1 )) <= N){  
tmp = k + n \* (k+1);  
if(A[tmp] == 0) A[tmp] = 1;  
else A[tmp] = 0;  
n++;  
}  
for(int j = 1; j <= N; j++)  
if(A[j] == 0) sl++;  
BT(indx + 1, sl);  
n = 0;  
while((k + n \* (k +1 )) <= N){  
tmp = k + n \* (k+1);  
if(A[tmp] == 0) A[tmp] = 1;  
else A[tmp] = 0;  
n++;  
}  
visit[k] = 0;  
}  
}  
}  
  
}  
  
public static void main(String args[]) throws Exception{  
System.setIn(new FileInputStream("src/Tietkiemdien/input"));  
Scanner sc = new Scanner([System.in](http://system.in/));  
  
T = sc.nextInt();  
for(int tc = 1; tc <= T; tc++){  
N = sc.nextInt();  
K = sc.nextInt();  
A = new int[N + 1];  
B = new int[N + 1];  
set = new int[K + 1];  
visit = new int[K+1];  
max = 0;  
for(int i = 1; i <= N; i++)  
{  
A[i] = sc.nextInt();  
if(A[i] == 0) max++;  
}  
  
BT(0, 0);  
System.out.println("#" + tc + " " +max);  
// for(int i = 1; i <= N; i++)  
// System.out.println(A[i]);  
}  
}  
  
}

package luyende;  
  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
public class BatTatDen {  
Scanner sc = new Scanner([System.in](http://system.in/));  
int t, n, k, ans;  
int arr[], visit[];  
  
void init() {  
ans = 0;  
arr = new int[n + 1];  
visit = new int[n + 1];  
}  
  
int checkStatus() {  
int count = 0;  
for (int i = 1; i <= n; i++) {  
if (arr[i] == 0) {  
count++;  
}  
}  
return count;  
}  
  
void changeStatus(int index) {  
// arr[index] = 1 - arr[index];  
for (int i = 0; i <= n; i++) {  
if ((index + i \* (index + 1)) >= 0 && (index + i \* (index + 1)) <= n) {  
arr[index + i \* (index + 1)] = 1 - arr[index + i \* (index + 1)];  
}  
}  
}  
  
void backTrack(int num, int dem) {  
if (checkStatus() == n) {  
ans = n;  
return;  
}  
if (num == 3 + 1) {  
int result = checkStatus();  
if (result > ans) {  
ans = result;  
}  
return;  
}  
for (int i = 1; i <= k; i++) {  
if (visit[i] == 0) {  
visit[i] = 1;  
// ko thay đổi trạng thái K  
backTrack(num + 1, dem);  
// thay đổi trạng thái K  
changeStatus(i);  
backTrack(num + 1, dem + checkStatus());  
changeStatus(i);  
// changeStatus(num); chỗ này là sai vì ở trên backTrack(num+1) mà ở dưới changeStatus(num) đủ thấy 2 biến num ở 2 chỗ khác nhau   
// num ở trên được cộng 1 nên lớn hơn num ở dưới  
visit[i] = 0;  
}  
}  
// backTrack(num + 1, dem);  
// changeStatus(num);  
// backTrack(num + 1, dem + checkStatus());  
// changeStatus(num);  
}  
  
void solution() {  
t = sc.nextInt();  
for (int tc = 1; tc <= t; tc++) {  
n = sc.nextInt();  
k = sc.nextInt();  
init();  
for (int i = 1; i <= n; i++) {  
arr[i] = sc.nextInt();  
}  
backTrack(1, 0);  
System.out.println("#" + tc + " " + ans);  
}  
}  
  
public static void main(String[] args) throws Exception {  
System.setIn(new FileInputStream("BatTatDen.txt"));  
BatTatDen so = new BatTatDen();  
so.solution();  
}  
  
}

Top of Form

Bottom of Form

Bài Tiết kiệm điện

đề bài đúng hơn thì phải là có k khóa từ 1 tới k chứ ko phải là k khóa đặt trong n bóng đèn đó

package luyende;  
  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
public class BatTatDen {  
Scanner sc = new Scanner([System.in](http://system.in/));  
int t, n, k, ans;  
int arr[], visit[];  
  
void init() {  
ans = 0;  
arr = new int[n + 1];  
visit = new int[n + 1];  
}  
  
int checkStatus() {  
int count = 0;  
for (int i = 1; i <= n; i++) {  
if (arr[i] == 0) {  
count++;  
}  
}  
return count;  
}  
  
void changeStatus(int index) {  
// arr[index] = 1 - arr[index];  
for (int i = 0; i <= n; i++) {  
if ((index + i \* (index + 1)) >= 0 && (index + i \* (index + 1)) <= n) {  
arr[index + i \* (index + 1)] = 1 - arr[index + i \* (index + 1)];  
}  
}  
}  
  
void backTrack(int num, int dem) {  
if (checkStatus() == n) {  
ans = n;  
return;  
}  
if (num == 3 + 1) {  
int result = checkStatus();  
if (result > ans) {  
ans = result;  
}  
return;  
}  
for (int i = 1; i <= k; i++) {  
if (visit[i] == 0) {  
visit[i] = 1;  
// ko thay đổi trạng thái K  
backTrack(num + 1, dem);  
// thay đổi trạng thái K  
changeStatus(i);  
backTrack(num + 1, dem + checkStatus());  
changeStatus(i);  
visit[i] = 0;  
}  
}  
// backTrack(num + 1, dem);  
// changeStatus(num);  
// backTrack(num + 1, dem + checkStatus());  
// changeStatus(num);  
}  
  
void solution() {  
t = sc.nextInt();  
for (int tc = 1; tc <= t; tc++) {  
n = sc.nextInt();  
k = sc.nextInt();  
init();  
for (int i = 1; i <= n; i++) {  
arr[i] = sc.nextInt();  
}  
backTrack(1, 0);  
System.out.println("#" + tc + " " + ans);  
}  
}  
  
public static void main(String[] args) throws Exception {  
System.setIn(new FileInputStream("BatTatDen.txt"));  
BatTatDen so = new BatTatDen();  
so.solution();  
}  
  
}