Bài 1:

Bạn Bình gia cảnh khó khăn nhưng đam mê tập tạ. Để có tiền tập tạ, hàng đêm bạn Bình đi làm ma cô gác cửa cho vũ trường Heaven. Vũ trường heaven rất là đắt khách chưa mở cửa đã có khách đứng xếp hàng đầy. Quản lý vũ trường yêu cầu Bình phải tính toán sao đó sao cho khi vũ trường đầy khách thì số nam với số nữ phải xấp xỉ nhau. Bình chỉ được cho khách xếp hàng vào theo từng người một (không cần quan tâm khách có đi thành nhóm hay không, vũ trường này rất là chảnh), tuy nhiên không phải lúc nào người xếp đầu hàng cũng được vào trước mà khi hứng lên Bình có thể cho người đứng thứ nhì trong hàng vào trước, nói chung Bình có căn bản tập tạ từ nhỏ nên hoàn toàn có thể xử lý được vị khách đứng đầu, miễn sao cân đối được số lượng nam và nữ đi vào vũ trường.  
  
Xui một cái là Bình giỏi tập tạ nhưng không thông minh lắm nên khả năng ghi nhớ các con số của Bình là có giới hạn. Khi số lượng chênh lệch giữa khách nam và nữ đã vào vũ trường là quá lớn Bình sẽ bị rối não và bạn ấy sẽ thông báo là vũ trường đã hết chỗ và đuổi khách về (Bình cũng rất là chảnh).  
  
Input  
Hàng đầu tiên chứa một số nguyên dương X<100 đây con số chênh lệch giữa lượng khách nam và nữ lớn nhất mà Bình nhớ được.  
  
Tiếp theo là một dãy số 0 và 1 cách nhau bởi các khoảng trắng. Dãy này chứa không quá 101 số và kết thúc bằng số -1. Dãy này mô tả hàng người đang chờ trước vũ trường, số 0 là khách nam, số 1 là khách nữ và con số đầu tiên bên tay trái đại diện cho người đầu tiên trong hàng.  
  
Ở đây chúng ta cũng giả sử là vũ trường chứa được hết tất cả người đang đứng trong hàng.  
  
Output  
Số khách tối đa Bình sẽ cho vào vũ trường mà không quên mất con số chênh lệch giữa nam và nữ

x = int(input())

a = []

check = True

while check:

a = list(map(int, input().split()))

for j in a:

if j == -1:

check = False

a.remove(j)

nam = 0

nu = 0

long = len(a)

i = 0

while long > 0:

if abs(nam-nu) > x:

if nam > nu and a[i]==1:

nu+=1

a.remove(a[i])

long-=1

elif nu > nam and a[i] == 0:

nam+=1

a.remove(a[i])

long-=1

else:

print(nam+nu-1)

exit()

else:

if a[i] == 1:

nu+=1

a.remove(a[i])

long-=1

elif a[i] == 0:

nam+=1

a.remove(a[i])

long-=1

print(nam+nu)

Bài 2:

Từ xa xưa, hình ảnh con đò bến nước đã đi vào trong thi ca và trong tâm tưởng dân tộc. Hình ảnh con đò ngày ngày đưa khách sang sông in đậm trong tâm trí nhiều người và đôi khi còn được dùng để so sánh với nghề dạy học.  
  
Mở rộng hình ảnh của những con đò chính là những chiếc phà, không chỉ chở được khách mà còn chở được cả xe của khách sang sông. Phà là một loại tàu lớn, có 2 đầu y như nhau, xe chạy lên ở một đầu và chạy xuống ở đầu kia. Mỗi đầu của phà tương ứng với một bờ của con sông, phà luôn luôn cập đầu bên phải và bờ bên phải và ngược lại, không bao giờ đổi đầu nhằm tối ưu thời gian chuyên chơn.  
  
Trên con sông kia có một chiếc phà dài l mét hàng ngày chở khách qua sông. Bỗng một hôm phà bị hư phải nằm sửa ở tả ngạn con sông, xe đến không qua được sông xếp thành hàng dài hai bên bờ bấm kèn inh ỏi nhức óc đinh tai. Một sinh viên UIT bị kẹt trong hàng xe dài đằng đẵng kia quyết định giết thời gian bằng cách viết một chương trình tính toán số lần phà sẽ phải chạy để đưa hết hàng xe đang đợi qua sông khi nó hoạt động trở lại. Hãy giúp bạn ấy.  
  
Input  
Mỗi test bắt đầu bằng hai con số nguyên 1≤l≤500 and 1≤m≤10000. Trong dó l là chiều dài của phà (tính bằng mét) và m là số lượng xe đang kẹt hai bên bờ. m dòng tiếp theo, đại diên cho các xe đang chờ phà theo đúng thứ tự. Mỗi dòng chứa hai thông tin là lượt là chiều dài xe (tính bằng centimét và là một số nguyên dương không quá 100000) và vị trí của bờ mà xe đang đậu (biểu diễn bằng một trong hai giá trị (“left” hoặc “right”)  
  
Output  
Số lượt qua sông phà phải chạy để chở hết số xe trên

l,m = list(map(int,input().split()))

a = []

b = []

for i in range(0,m):

k = input().split()

if 'left' in k[1]:

b.append(int(k[0]))

else:

a.append(int(k[0]))

l1 = l\*100

def chuyenhang(k,l1):

k1 = []

sum = 0

dodai = len(k)

while dodai > 0:

sum += k[0]

if sum < l1:

if dodai == 1:

k1.append(k[0])

k.remove(k[0])

dodai-=1

elif sum > l1:

k1.append(sum - k[0])

sum = 0

elif sum == l1:

k1.append(sum)

sum = 0

k.remove(k[0])

dodai -=1

return len(k1)

bt = chuyenhang(b,l1)

bp = chuyenhang(a,l1)

if bt > bp:

print(bt\*2-1)

elif bt < bp:

print(bp\*2)

elif bt == bp:

print(bt\*2)

Bài 3:

Từ xa xưa, hình ảnh con đò bến nước đã đi vào trong thi ca và trong tâm tưởng dân tộc. Hình ảnh con đò ngày ngày đưa khách sang sông in đậm trong tâm trí nhiều người và đôi khi còn được dùng để so sánh với nghề dạy học.  
  
Mở rộng hình ảnh của những con đò chính là những chiếc phà, không chỉ chở được khách mà còn chở được cả xe của khách sang sông. Phà là một loại tàu lớn, có 2 đầu y như nhau, xe chạy lên ở một đầu và chạy xuống ở đầu kia. Mỗi đầu của phà tương ứng với một bờ của con sông, phà luôn luôn cập đầu bên phải và bờ bên phải và ngược lại, không bao giờ đổi đầu nhằm tối ưu thời gian chuyên chở.  
  
Trên con sông kia có một chiếc phà dài l mét hàng ngày chở khách qua sông. Bỗng một hôm phà bị hư phải nằm sửa ở tả ngạn con sông, xe đến không qua được sông xếp thành hàng dài hai bên bờ bấm kèn inh ỏi nhức óc đinh tai. Một sinh viên UIT bị kẹt trong hàng xe dài đằng đẵng kia quyết định giết thời gian bằng cách viết một chương trình tính toán số lần phà sẽ phải chạy để đưa hết hàng xe đang đợi qua sông khi nó hoạt động trở lại. Hãy giúp bạn ấy.  
  
Input  
Mỗi test bắt đầu bằng hai con số nguyên 1≤l≤500 và 1≤m≤10000. Trong dó l là chiều dài của phà (tính bằng mét) và m là số lượng xe đang kẹt hai bên bờ.  
  
m dòng tiếp theo, đại diên cho các xe đang chờ phà theo đúng thứ tự. Mỗi dòng chứa ba thông tin là lượt là chiều dài xe (tính bằng centimét và là một số nguyên dương không quá 100000), loại xe, và vị trí của bờ mà xe đang đậu (biểu diễn bằng một trong hai chuỗi left và right.  
  
Trong đó loại xe có 5 loại là xe thường, xe cảnh sát, xe quân sự, xe chữa cháy, xe cứu thương. Mỗi loại xe có độ ưu tiên khác nhau, xe có độ ưu tiên thấp sẽ không bao giờ được lên phà trước xe có độ ưu tiên cao hơn, còn giữa hai xe có cùng độ ưu tiên thì chiếc nào tới trước lên trước. Thông tin về loại xe được cho bằng một chuỗi tiếng Anh với độ ưu tiên từ thấp tới cao lần lượt là: civilian, police, army, fire, ambulance  
  
Output  
Số lượt qua sông phà phải chạy để chở hết số xe trên

l,m = map(int,input().split())

a1 = []

a2 = []

a3 = []

a4 = []

a5 = []

b1 = []

b2 = []

b3 = []

b4 = []

b5 = []

for i in range(0,m):

k = input().split()

if 'right' in k[2]:

if 'ambulance' in k[1]:

a1.append(int(k[0]))

elif 'fire' in k[1]:

a2.append(int(k[0]))

elif 'army' in k[1]:

a3.append(int(k[0]))

elif 'police' in k[1]:

a4.append(int(k[0]))

elif 'civilian' in k[1]:

a5.append(int(k[0]))

else:

if 'ambulance' in k[1]:

b1.append(int(k[0]))

elif 'fire' in k[1]:

b2.append(int(k[0]))

elif 'army' in k[1]:

b3.append(int(k[0]))

elif 'police' in k[1]:

b4.append(int(k[0]))

elif 'civilian' in k[1]:

b5.append(int(k[0]))

def xuly(a,l1):

for i in range(0,len(a)):

if a[i] > l1:

a.remove(a[i])

return a

l1 = l\*100

a11 = xuly((a1 + a2 + a3 + a4 + a5),l1)

b11 = xuly((b1+b2+b3+b4+b5),l1)

def chuyenhang(k,l1):

k1 = []

sum = 0

dodai = len(k)

while dodai > 0:

if k[0] <= l1:

sum += k[0]

if sum < l1:

if dodai == 1:

k1.append(k[0])

k.remove(k[0])

dodai-=1

elif sum > l1:

k1.append(sum - k[0])

sum = 0

elif sum == l1:

k1.append(sum)

sum = 0

k.remove(k[0])

dodai -=1

else:

k.remove(k[0])

dodai -=1

return len(k1)

bt = chuyenhang(b11,l1)

bp = chuyenhang(a11,l1)

if bt > bp:

print(bt\*2-1)

elif bt < bp:

print(bp\*2)

elif bt == bp:

print(bt\*2)

Bài 4:

Cho một mảng số nguyên, trong đó mỗi giá trị có thể xuất hiện nhiều lần. Hãy viết chương trình chob biết số nhỏ nhất trong mảng xuất hiện bao nhiêu lần.  
  
INPUT  
Dòng đầu tiên chứa số n (n < 10000). Đây là số lượng phần tử trong mảng.  
  
n dòng tiếp theo, mỗi dòng là một số nguyên dương có giá trị không quá một tỷ.  
  
OUTPUT  
Xuất ra trên cùng một hàng “số nhỏ nhất trong mảng” và số lầ xuất hiện của số đó, cách nhau bởi khoảng trắng

n = int(input())

a = []

for i in range(0,n):

a.append(int(input()))

dem = 0

k = min(a)

for i in range(0,n):

if a[i] == k:

dem +=1

print(min(a),dem)

Bài 5:

Ta gọi mảng bội ước là mảng mà mọi phần tử của nó thõa một trong hai tính chất sau:  
  
Nó là ước của một trong các phần tử trước nó  
Nó là bội của một trong các phần tử trước nó.  
Bạn hãy viết chương trình kiểm tra xem một mảng có phải là mảng bội ước không. Và nếu không phải thì phần tử đầu tiên không thõa tính chất bội ước đó là phần tử nào?  
  
INPUT  
Dòng đầu tiên chứa số n (n < 1000000). Đây là số lượng phần tử trong mảng.  
  
n dòng tiếp theo, mỗi dòng là một số nguyên dương có giá trị không quá một tỷ.  
  
OUTPUT  
Xuất giá trị số đầu tiên không thõa tính chất của mảng bội ước. Xuất -1 nếu không tìm được số như vậy.

n = int(input())

a = []

for i in range(0,n):

a.append(int(input()))

if n > 1:

for i in range(1,n):

for j in range(0,i):

if ((a[i] % a[j] == 0) or (a[j] % a[i] == 0)):

break

else:

print(a[i])

exit()

if i == n-1:

for j in range(0,i):

if ((a[i] % a[j] == 0) or (a[j] % a[i] == 0)):

print(-1)

exit()

else:

print(a[-1])

else:

print(-1)

Bài 6:

Cho một mảng trong đó mỗi số chỉ xuất hiện 1 lần, đếm xem cần phải chèn vào mảng ít nhất bao nhiêu phần tử để mảng chứa tất cả số nguyên trên một đoạn liên tục.  
  
INPUT  
Dòng đầu tiên chứa số n (n < 100000), là số lượng phần tử của mảng.  
  
n dòng tiếp theo, mỗi dòng chứa một số nguyên, đây là các phần tử của mảng.  
  
OUTPUT  
Số phần tử tối thiểu cần thêm vào mảng để mảng chứa toàn bộ số tự nhiên liên tiếp và không chứa bất cứ số nào khác.

n = int(input())

a = []

for i in range(0,n):

a.append(int(input()))

d = min(a)

t = max(a)

if (d >= 0 and t >= 0) or (d <=0 and t<=0):

print((t-d)-n+1)

elif d < 0 and t >= 0:

print((-d)+t+1 - n)

Bài 7:

Cho tập hợp a gồm N số nguyên.  
  
Yêu cầu: tính tích của tập con có tích lớn nhất của a.  
  
Ví dụ: tập a có 5 số: {2, 5, 0, -7, 2}, tập con có tích lớn nhất là {2, 5, 2} với tích là 20.  
  
INPUT  
Dòng thứ nhất ghi số nguyên dương N (0 < N ≤ 30)  
  
Dòng thứ 2 ghi N số là các số của a, mỗi số có trị tuyệt đối không quá 30  
  
OUTPUT  
tích số tìm được.

n = int(input())

a = list(map(int,input().split()))

dem = 0

soam = 0

for i in range(0,n):

if a[i] == 0:

dem+=1

for i in range(0,n):

if a[i] < 0:

soam+=1

if dem == n or (dem == n - 1 and soam == 1):

print(0)

exit()

tich= 1

max=-1000000000

for i in range(0,n):

if a[i] != 0:

tich\*=a[i]

if a[i]<0 and a[i]>max:

max=a[i]

if tich<0:

print(int(tich/max))

else:

print(int(tich))

Bài 8:

Nhập một mảng các chuỗi. Xuất mảng đó ra trên màng hình và canh các chuỗi theo lề phải.  
  
INPUT  
Một loạt chuỗi. Mỗi chuỗi có độ dài không quá 1000 ký tự, không chứa khoảng trắng và được viết trên một hàng.  
  
Input có tổng cộng không quá 1000 chuỗi. Input kết thúc khi gặp chuỗi chỉ chứa dấu chấm.  
  
OUTPUT  
Xuất ra các chuỗi theo đúng thứ tự đã nhập nhưng ký tự cuối cùng của các chuỗi được canh thẳng hàng với nhau (canh hàng theo lề phải). Với các chuỗi ngắn hơn những chuỗi khác thì xuất thêm vài khoảng trắng phía trước chuỗi đó.

a = []

while True:

k = input()

if k == '.':

break

a.append(k)

def doi(k):

maxw= max(len(s) for s in k)

for s in k:

kt = maxw - len(s)

print(' ' \* kt + s)

doi(a)

Bài 9:

Khi nói đến việc tìm kiếm trong mảng, đôi khi đối tượng cần tìm không phải chỉ là một phần tử mà có thể là một cặp hoặc một nhóm phần tử thỏa mãn một hoặc một số tính chất nào đó. Ở dạng đơn giản nhất ta có thể có dạng bài tập tìm trong mảng các cặp số có tổng bằng x  
  
INPUT  
Dòng đầu tiên của mảng chứa số n (n không quá 100.000) đây là số phần tử của mảng. n dòng tiếp theo, mỗi dòng chứa một số nguyên không âm.  
  
Dòng cuối cùng của input là số nguyên dương x.  
  
OUTPUT  
xuất ra 2 số a và b trong mảng, mỗi số trên một dòng. Trong đó a ≤ b. Nếu mảng có nhiều cặp số a, b thỏa yêu cầu tìm kiếm thì xuất ra cặp có chênh lệch giữa a và b là nhỏ nhất.

n = int(input())

a = []

for i in range(0,n):

a.append(int(input()))

x = int(input())

1. sort()#timkiemnhiphan

def binary\_search(a, left, right, x):

while left <= right:

mid = (left + right) // 2

if a[mid] == x:

return mid

elif a[mid] < x:

left = mid + 1

else:

right = mid - 1

return -1

b = []

for i in range(0,n):

k = x - a[i]

s = binary\_search(a,0,n-1,k)

if s != -1 and s != i:

b.append(k)

b.append(a[i])

b.sort()

if b[int(len(b)/2)-1] > b[int(len(b)/2)]:

print(b[int(len(b)/2)])

print(b[int(len(b)/2)-1])

else:

print(b[int(len(b)/2)-1])

print(b[int(len(b)/2)])

Bài 10:

Dãy con của một mảng là một mảng thu được bằng cách xóa đi một số phần tử của mảng ban đầu. Ví dụ mảng: [3, 2, 1] có thể có các dãy con [3, 2], [2, 1], [3, 1]  
  
Hãy viết chương trình kiểm tra xem mảng này có phải là mảng con của mảng kia hay không.  
  
INPUT  
Hai số m, n lần lượt là số phần tử của 02 mảng. Giá trị mỗi số không quá một tỷ  
  
Dòng tiếp theo chứa m số nguyên đây là các phần tử của mảng thứ nhất  
  
Dòng cuối cùng của input chứa n số nguyên, đây là các phần tử của mảng thứ hai  
  
OUTPUT  
Nếu trong 02 mảng trên, mảng này là dãy con của mảng kia xuất ra màn hình chuỗi “TRUE” nếu không xuất ra chuỗi “FALSE

m,n = map(int,input().split())

a = list(map(int,input().split()))

b = list(map(int,input().split()))

if m > n:

k = 0

for i in range(0,n):

for j in range(k + 1,m):

if b[i] == a[j]:

k = j

if i == n - 1:

print("TRUE")

exit()

break

elif b[i] != a[j]:

if j == m - 1:

print("FALSE")

exit()

if k == m - 1 and b[i] != a[k] :

print("FALSE")

exit()

elif m < n:

k = 0

for i in range(0,m):

for j in range(k + 1,n):

if a[i] == b[j]:

k = j

if i == m - 1:

print("TRUE")

exit()

break

elif a[i] != b[j]:

if j == n - 1:

print("FALSE")

exit()

if k == n - 1 and a[i] != b[k] :

print("FALSE")

exit()

Bài 11:

Phép XOR bitwise thì hầu hết ai có học lập trình đều đã nghe rồi. Ai chưa nghe thì có thể xem lại wikipedia:  
  
A bitwise XOR is a binary operation that takes two bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only one of the bits is 1, but will be 0 if both are 0 or both are 1. In this we perform the comparison of two bits, being 1 if the two bits are different, and 0 if they are the same. For example:  
  
0101 (decimal 5)  
XOR 0011 (decimal 3)  
= 0110 (decimal 6)  
The bitwise XOR may be used to invert selected bits in a register (also called toggle or flip). Any bit may be toggled by XORing it with 1. For example, given the bit pattern 0010 (decimal 2) the second and fourth bits may be toggled by a bitwise XOR with a bit pattern containing 1 in the second and fourth positions:  
  
0010 (decimal 2)  
XOR 1010 (decimal 10)  
= 1000 (decimal 8)  
This technique may be used to manipulate bit patterns representing sets of Boolean states.  
  
Phép toán này có rất nhiều điểm thú vị, đầu tiên là nó có tính chất giao hoán và kết hợp. Nghĩa là khi các bạn có một tập số và các bạn lần lượt XOR các số này với nhau thì kết quả sẽ không đổi dù các bạn có dùng trật tự nào để XOR. Tính chất này khá là giống với phép nhân và phép cộng. Bạn Bình đang nghiên cứu về số học và rất hứng thú với khái niệm integer partition, tức là viết một số nguyên thành tổng của một dãy số nguyên khác. Bạn Bình bèn liên tưởng đến nhập môn lập trình và không biết có cách nào có thể viết một số nguyên thành kết quả XOR của một dãy số nguyên khác hay không.  
  
Bây giờ Bình cần tìm một dãy số dài nhất có thể, gồm toàn các số nguyên dương nhỏ hơn hoặc bằng m, trong đó mỗi số chỉ xuất hiện một lần và khi XOR tất cả chúng lại với nhau thì kết quả bằng m, bạn giúp Bình code đi nha.  
  
INPUT  
Dòng đầu tiên là t, số lượng test cases (t < 1000)  
  
t dòng tiếp theo, mỗi dòng chứa một số nguyên dương m (m < 10000)  
  
OUTPUT  
Ứng với mỗi test case, xuất ra hai dòng. Đầu tiên là độ dài của dãy tìm được. Dòng tiếp theo là các số trong dãy, viết theo thứ tự tăng dần và nếu có nhiều dãy cùng độ dài, hãy xuất ra dãy có trật tự nhỏ nhất.

n = int(input())

a = []

for i in range(0,n):

a.append(int(input()))

for i in range(0,n):

if a[i] == 1:

print(1)

print(1)

continue

if a[i] == 2:

print(1)

print(2)

continue

if a[i] == 3:

print(2)

print(1,2)

continue

target = 2

for j in range(3,a[i] - 1):

target ^= j

target1 = 1^target^(a[i]-1)^a[i]

if target1 == a[i]:

print(a[i])

for j in range(1, a[i]+1):

print(j,end =' ')

print()

continue

target1 = 1^target^(a[i]-1)

target2 = 1^target^a[i]

if target1 == a[i]:

print(a[i] - 1)

for j in range(1, a[i]):

print(j,end=' ')

print()

continue

if target2 == a[i]:

print(a[i] - 1)

for j in range(1, a[i]-1):

print(j, end=' ')

print(a[i])

continue

target1 = target^(a[i]-1)^a[i]

if target1 == a[i]:

print(a[i]-1)

for j in range(2, a[i]+1):

print(j,end=' ')

print()

continue