# 復習済み

A-E

# A 123233

正解

## メモ

ソートによる一致を意識(集合ではなく)

# B Hurdle Parsing

正解

# C Move Segment

正解

## 自分のコード

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

s = input()

z = []

flag = 0

count = 0

i = 0

while i < n:

if flag == 0 and s[i] == "0":

count += 1

elif flag == 0 and s[i] == "1":

z.append(count)

flag = 1

count = 1

elif flag == 1 and s[i] == "0":

z.append(count)

flag = 0

count = 1

elif flag == 1 and s[i] == "1":

count += 1

i += 1

z.append(count)

for i in range(2\*k-2):

if i%2 == 0:

print("0"\*z[i], end = "")

else:

print("1"\*z[i], end = "")

print("1"\*z[2\*k-1], end = "")

print("0"\*z[2\*k-2], end = "")

i = 0

for x in z[2\*k:]:

if i%2 == 0:

print("0"\*x, end = "")

else:

print("1"\*x, end = "")

i += 1

## 解説のコード

N,K = map(int,input().split())

S = input()

# split

idx = [0] + [i for i in range(1,N) if S[i-1]!=S[i]] + [N]

splited\_S = [S[idx[i]:idx[i+1]] for i in range(len(idx)-1)]

# swap

if S[0] == '0':

kth\_1\_idx = 2\*K-1

else:

kth\_1\_idx = 2\*K-2

splited\_S[kth\_1\_idx-1], splited\_S[kth\_1\_idx] = splited\_S[kth\_1\_idx], splited\_S[kth\_1\_idx-1]

# join

T = "".join(splited\_S)

print(T)

## 復習正解コード

N, K = map(int, input().split())

S = input()

id = [None]\*3

i = 0

pre = 0

onecount = 0

while i < N:

if S[i] == "0":

if pre == 1:

if onecount == K-1:

id[0] = i

elif onecount == K:

id[2] = i

pre = 0

else:

if pre == 0:

if onecount == K-1:

id[1] = i

onecount += 1

pre = 1

i += 1

if id[2] == None:

id[2] = N

print(S[:id[0]] + S[id[1]:id[2]] + S[id[0]:id[1]] + S[id[2]:])

# D Strange Mirroring

不正解

## 自分のコード

S = input()

Q = int(input())

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

lens = len(S)

ans = []

for k in K:

bufk = (k-1)//lens

onec = bin(bufk).count('1')

buf = S[(k-1)%lens]

if onec%2 == 0:

ans.append(buf)

else:

if buf.isupper():

ans.append(buf.lower())

else:

ans.append(buf.upper())

print(\*ans)

## 復習コード

S = input()

Q = input()

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

N = len(S)

ans = []

for k in K:

mass\_id, str\_id = divmod(k-1, N)

bin\_one = bin(mass\_id).count("1")

if bin\_one%2 == 0:

ans.append(S[str\_id])

else:

ans.append(S[str\_id].swapcase())

print(\*ans)

# E 1D Bucket Tool

解いてない

unionfindで連結成分ごとに、色とその個数とそれがどの範囲にいるか左右の座標を記録した。クエリごとに全体の色の数を変更した。ちょっと改善したコードもあるが、もっと速い人もいる。めんどくさいので後回し

## 正解コード

class Unionfind:

def \_\_init\_\_(self, n):

self.A = [[-1, i, i, i] for i in range(n)]

def find(self, x):

if isinstance(self.A[x], list):

return x

else:

self.A[x] = self.find(self.A[x])

return self.A[x]

def union(self, a, b, c): # a < b

ra = self.find(a)

rb = self.find(b)

if ra == rb:

return False

if self.A[ra][0] <= self.A[rb][0]:

self.A[ra][2] = self.A[rb][2]

self.A[ra][0] += self.A[rb][0]

self.A[ra][3] = c

self.A[rb] = ra

else:

self.A[rb][1] = self.A[ra][1]

self.A[rb][0] += self.A[ra][0]

self.A[rb][3] = c

self.A[ra] = rb

return True

N, Q = map(int, input().split())

color\_count = [1]\*N

uf = Unionfind(N)

for i in range(Q):

q = input().split()

if q[0] == "1":

x = int(q[1])-1

c = int(q[2])-1

rx = uf.find(x)

now\_num = -uf.A[rx][0]

now\_c = uf.A[rx][3]

color\_count[now\_c] -= now\_num

color\_count[c] += now\_num

left = uf.A[rx][1]

right = uf.A[rx][2]

if left >= 1:

left\_c = uf.A[uf.find(left-1)][3]

if left\_c == c:

uf.union(left-1, rx, c)

if right <= N-2:

right\_c = uf.A[uf.find(right+1)][3]

if right\_c == c:

uf.union(rx, right+1, c)

rx = uf.find(x)

uf.A[rx][3] = c

else:

c = int(q[1])-1

print(color\_count[c])

## 改善コード

class Unionfind:

def \_\_init\_\_(self, n):

self.A = [-1]\*(n+1)

self.left = [i for i in range(n+1)]

self.color = [i for i in range(n+1)]

def find(self, x):

if self.A[x] < 0:

return x

else:

self.A[x] = self.find(self.A[x])

return self.A[x]

def union(self, a, b): # a < b

ra = self.find(a)

rb = self.find(b)

if ra == rb:

return False

if self.A[ra] <= self.A[rb]:

self.A[ra] += self.A[rb]

self.A[rb] = ra

else:

self.left[rb] = self.left[ra]

self.A[rb] += self.A[ra]

self.A[ra] = rb

return True

N, Q = map(int, input().split())

color\_count = [1]\*(N+1)

uf = Unionfind(N)

for i in range(Q):

q = tuple(map(int, input().split()))

if q[0] == 1:

rx = uf.find(q[1])

color\_count[uf.color[rx]] += uf.A[rx]

color\_count[q[2]] -= uf.A[rx]

left = uf.left[rx]-1

right = uf.left[rx]-uf.A[rx]

if left >= 1:

if uf.color[uf.find(left)] == q[2]:

uf.union(uf.find(left), rx)

if right <= N:

if uf.color[uf.find(right)]== q[2]:

uf.union(rx, uf.find(right))

rx = uf.find(rx)

uf.color[rx] = q[2]

else:

print(color\_count[q[1]])

# F Exchange Game

解いてない

# G Another Shuffle Window

解いてない