BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG: ĐH GIAO THÔNG VẬN TẢI

KHOA: CNTT

-----📖-----

BÀI TẬP LỚN

Môn Học : Thuật toán và ứng dụng

Giáo viên hướng dẫn: Phạm Xuân Tích

Sinh viên thực hiện: Nguyễn Việt Anh

Lớp: Công nghệ thông tin 1

Khóa: 61

Bài 1:

Bài 1:

Bài toán đặt ra là có n người được đánh số từ 1 đến n và bắt đầu từ người số 1 cứ đếm đến người thứ k thì loại ra khỏi vòng tròn và lại bắt đầu chơi tiếp từ người

thứ k+1 được đếm từ 1 cứ tiếp tục như vậy vì đứng thành vòng tròn nên lần lượt sẽ loại hết chỉ còn người cuối cùng. Hãy cho biết chỉ số của người cuối cùng là bao nhiêu?

Input

Dòng đầu có hai số nguyên dương n và k 1<k<n<=1000

Output

Một số nguyên dương duy nhất là chỉ số của người cuối cùng

Ý tưởng:

Sử dụng queue

Đầu tiên đưa n từ 1 đến n vào queue(tương ứng n người)

Khi nào mà số phần tử trong queue lớn hơn 1 ta thực hiện:

+Đưa k-1 người đầu vào cuối queue(cuối hàng). Mỗi lần đưa thì xóa bỏ người đó khỏi queue

+Xóa bỏ người thứ k (đưa người thứ k ra khỏi hàng).

Người cuối cùng còn lại trong queue là kết quả

*Giải tay:

Ví dụ n = 13, k = 3

Queue ban đầu: [1,2,3,4,5,6,7,8,9,10,11,12,13]

+lặp 1: Đưa 1, 2 vào cuối, xóa 1,2. Loại 3. Queue: [4,5,6,7,8,9,10,11,12,13,1,2]

+lặp 2: Đưa 4,5 vào cuối, xóa 4,5. Loại 6 : Queue :[7,8,9,10,11,12,13,1,2,4,5]

+Lặp 3: Đưa 7,8 vào cuối, xóa 7,8 .Loại 9 Queue :[10,11,12,13,1,2,4,5,7,8]

+Lặp 4: Đưa 10,11 vào cuối, xóa 10,11 Loại 12 Queue:[13,1,2,4,5,7,8,10,11]

+Lặp 5: Đưa 13, 1 vào cuối, xóa 13,1 Loại 2 Queue [4,5,7,8,10,11,13,1]

+Lặp 6: Đưa 4,5 vào cuối, xóa 4,5 Loại 7 Queue[8,10,11,13,1,4,5]

+lặp 7: Đưa 8,10 vào cuối, xóa 8,10 loại 11 queue [13,1,4,5,8,10]

+Lặp 8: Đưa 13,1 vào cuối, xóa 13,1 Loại 4 queue [5,8,10,13,1]

+Lặp 9: Đưa 5,8 vào cuối, xóa 5,8 Loại 10 queue [13,1,5,8]

+Lặp 10: Đưa 13,1 vào cuối, xóa 13,1 Loại 5 queue [8,13,1]

+Lặp 11: Đưa 8,13 vào cuối, xóa 8, 13 Loại 1 queue [8,13]

+Lặp 12: Đưa 8,13 vào cuối, xóa 8,13 Loại 8 queue [13]

Size = 1 nên dừng lại.

Kết quả : 13

*Code:

```
import queue
if __name__ == '__main__':
n, k = map(int, input().split())
q = queue.Queue()
for i in range(1, n + 1, 1):
q.put(i)
while q.qsize() > 1:
for i in range(1, k):
x = q.get()
q.put(x)
q.get()
print(q.get())
```

Bài 2: Một tòa nhà cao n tầng được đánh số từ tầng 1 đến tầng n có lắp 1 thang máy. Hiện nay thang máy bị trục trặc do đó mỗi lần có thể di chuyển lên đúng k

tầng hoặc di chuyển xuống đúng m tầng nếu có đủ không gian để di chuyển. Thang máy đang ở tầng s một người ở đó muốn di chuyển đến tầng f hỏi số bước di chuyển ít nhất để đến được tầng f nếu không di chuyển được xuất ra -1.

Input

Dòng đầu chứa n là số tầng của tòa nhà (1<n<300)

Dòng thứ 2 chứa 2 số k và m tương ứng là số tầng lên và xuống mỗi lần di chuyển của thang máy (1<=k,m<=50)

Dòng thứ 3 chứa s và f (1<=s,f<=n) là vị trí tầng xuất phát và đích đến

Output

Số bước ít nhất di chuyển thang máy, nếu không đến được đích xuất ra -1

Ý tưởng Dùng thuật toán BFS(tìm kiếm theo chiều rộng)

Đầu vào n: số tầng, k và m ứng là số tầng lên và xuống mỗi lần di chuyển, s và f là vị trí tầng xuất phát và đích

Dùng queue lưu các tầng có thể đi

Dict lưu thông tin (keys là tầng, values là số bước)

Đưa tầng xuất phát s values = 0 vào Dict

Ta sẽ thực hiện những bước sau đến khi nào queue rỗng:

Lấy phần tử đầu của queue : u = Q.get()

1. TH có thể đi lên được(u+k <= n) và tầng đó chưa đi qua ta(u+ k not in Dict.keys()) sẽ đưa tầng đó vào queue và Dict[u+k] = Dict[u]+1
2. TH có thể đi xuống được (u-m >=1) và tầng đó chưa đi qua(u – m not in Dict.keys()) sẽ đưa tầng đó vào queue và Dict[u- m] = Dict[u]+1

Nếu như có đường đi đến tầng f tức f in Dict.keys() thì in ra số bước ít nhất đến f (print(Dict[f])), không thì in ra -1.

*Giải tay:

Ví dụ n = 5, k = 2, m = 3, s =1, f= 4

Đầu tiên: Đưa s = 1 vào queue, xét D[s] = 0

Khi này queue có 1 phần tử:

Lấy phần tử đầu: u = Q.get(),

Xét đi lên: u+k = 1 + 2 = 3 <=5 và 3 not in D.keys(): Đưa 3 vào queue, D[3] = D[1]+1 = 1.

Xét đi xuống: u – m = 1 – 3 = -2 <1 => không thỏa mãn

Khi này queue có 1 phần tử là 3.

Lấy phần tử đầu: u = Q.get(),

Xét đi lên: u+k = 3 + 2 = 5 <=5 và 5 not in D.keys(): Đưa 5 vào queue, D[5] = D[3]+1 = 2.

Xét đi xuống: u – m = 3 – 3 = 0 <1 => không thỏa mãn

Khi này queue có 1 phần tử là 5.

Lấy phần tử đầu: u = Q.get(),

Xét đi lên: u+k = 5 + 2 = 7 > 5 => không thỏa mãn.

Xét đi xuống: u – m = 5 – 3 = 2 >=1 và 2 not in D.keys() : Đưa 2 vào queue , D[2] = D[5] +1 =3.

Khi này queue có 1 phần tử là 2.

Lấy phần tử đầu: u = Q.get(),

Xét đi lên: u+k = 2 + 2 = 4 <= 5 và 4 not in D.keys(): Đưa 4 vào queue, D[4] = D[2]+1 = 4

Xét đi xuống: u – m = 2 – 3 = -1 <1 => không thỏa mãn

Khi này queue có 1 phần tử là 4.

Lấy phần tử đầu: u = Q.get(),

Xét đi lên: u+k = 4 + 2 = 6 >= 5 => không thỏa mãn

Xét đi xuống: u – m = 4 – 3 = 1 >=1 nhưng 1 in D.keyss() => không thỏa mãn

Khi này queue rỗng. Dừng vòng lặp

Duyệt các phần tử trên D.keys() có 4 nên kết quả là D[4] = 4

```
5
2 3
1 4
4
Cac tang di qua:1
Cac tang di qua:3
Cac tang di qua:5
Cac tang di qua:2
Cac tang di qua:4
```

*Code bằng python

```
import queue
def bfs():
n = int(input()) # n tang
k, m = map(int, input().split()) #k là số tâng lên, m là số tâng xuống
s, f = map(int, input().split()) # s là vị trí xuất phát, f là tâng đích
Q = queue.Queue() # hàng đợi chứa các tầng đi được
Q.put(s) # đưa vị trí xuất phát vào hàng đợi
D = {s: 0} # map, #key là số tầng, values là số bước
while Q.qsize():
u = Q.get()
#đi lên
if u + k <= n and u + k not in D.keys():# nếu còn đi lên được và tầng đó chưa đi qua
D[u + k] = D[u] + 1 #tang so buoc
Q.put(u + k) # thêm vào hàng đợi
# đi xuống
if u - m >= 1 and u - m not in D.keys(): # nếu còn đi xuống được và tầng đó chưa đi qua
D[u - m] = D[u] + 1 # tang so buoc
Q.put(u - m) # thêm vào hàng đợi
if f in D.keys():print(D[f]) # nếu đi đến được tầng đích(tầng đích có trong D) thì in ra số bước D[f]
else:print(-1) # không đi đến được tầng đích.
if __name__=='__main__':
bfs()
```