## Queue

Hàng đợi (queue) là cấu trúc dữ liệu được xử lý theo nguyên tắc *vào trước ra trước* (FIFO: Firrst In, Firrst Out) đúng như khi ta xếp hàng chờ được phục vụ. Mỗi hàng đợi q khi đó sẽ có hai cửa phân biệt: cửa vào và cửa ra.

#### Các thao tác cơ bản trên queue

Trong Python, hàng đợi được cài đặt qua kiểu dữ liệu danh sách (list) như sau:

* Khởi tạo hàng đợi rỗng: q = []
* Nạp đối tượng x vào cuối hàng đợi q: q.append(): đối tượng x sẽ được nạp vào cuối q, dung lượng của q sẽ được tăng thêm 1 đơn vị.
* Xem phần tử cuối hàng đợi q: q[-1]
* Xem phần tử đầu hàng đợi q: q[0]
* Lấy đối tượng đầu hàng đợi ra khỏi hàng đợi: x = st.pop(0), dung lượng của q sẽ được giảm bớt 1 đơn vị.
* Kiểm tra hàng đợi q rỗng? len(q) == 0
* Hiển thị nội dung hàng đợi: print(q)

### Josephus Problem

Trò chơi chọn duy nhất một tù trưởng trong số n người có tài ngang nhau được thực hiện như sau:

Xếp vòng tròn n người. Bắt đầu từ người thứ nhất đếm k người, loại khỏi vòng tròn người thứ k. Khi nào còn lại 1 người thì người đó được bầu làm tù trưởng.

Ví dụ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A green circle with numbers  Description automatically generated | n = 12, k = 5   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Thành viên | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | Trật tự loại |  | 10 | 3 | 5 | 1 | 11 | 8 | 7 | 4 | 2 | 9 | 6 |   Trúng cử: 1 |

#### Algorithm

Bài này có nhiều cách giải.

Cách giải thứ nhất. Mô phỏng qua dãy số. Gọi a là dãy số 1..n. Duyệt a theo vòng tròn gặp người thứ k thì loại bằng phép gán trị a[v] = 0. Độ phức tạp O(n)

Cách giải thứ 2. Mô phỏng qua hàng đợi q. Gọi q là dãy số với các giá trị khởi đầu 1..n. Duyệt q để chuyển k-1 người từ đầu q về cuối q, loại phần tử thứ k khỏi q. Người cuối cùng còn lại trong q là kết quả. Mô hình hóa theo hàng đợi có tính trực quan hơn theo dãy số về thao tác loại bỏ phần tử. Giả sử ta có vòng tròn n = 12 người và ta phải loại bỏ người thứ k = 5 sau mỗi lượt đếm theo bước 5. Ta có

Khởi trị: q = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

Lượt đếm thứ nhất q = [6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4] (loại người số 5)

Lượt đếm thứ hai q = [11, 12, 1, 2, 3, 4, 6, 7, 8, 9] (loại người số 10)

...

#### Độ phức tạp

O(n)

#### Program

# Josephus Problem JP

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def JP(n, k):

if n == 1: return 1

if k == 1: return n

q = list(range(1,n+1))

print('Init:', q)

for i in range(1,n): # lặp n-1 lần

# chuyển k-1 phần tử từ đầu vè cuối

for j in range(1,k): q.append(q.pop(0))

q.pop(0) # bỏ phần tử thứ k

print(q)

return q[0] # phần tử còn lại cuối cùng

# APPLICATION

print(JP(12, 5))

print(" T h e E n d")

#### Output

Init: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

[6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4]

[11, 12, 1, 2, 3, 4, 6, 7, 8, 9]

[4, 6, 7, 8, 9, 11, 12, 1, 2]

[11, 12, 1, 2, 4, 6, 7, 8]

[6, 7, 8, 11, 12, 1, 2]

[1, 2, 6, 7, 8, 11]

[11, 1, 2, 6, 7]

[11, 1, 2, 6]

[1, 2, 6]

[6, 1]

[1]

1

T h e E n d

#### Bình luận 1

Một phương án giải khác dựa trên hàm đệ quy như sau. Gọi V(n, k) là số hiệu của người còn lại khi thực hiện trò chơi Josephus với n người và bước đếm k. Ta thấy,

nếu n = 1 thì V(n, k) = 1

nếu n > 1 thì V(n,k) = (V(n-1, k) + (k-1)) % n + 1

Bạn có thể cài đặt phương án này qua hàm lặp Rest như sau:

def Rest(n, k):

if n == 1: return 1

k -= 1; v = 1

for i in range(2, n+1): v = ((v + k) % i) + 1

return v

## Dequeue

Hàng đợi hai đầu (dequeue) là cấu trúc dữ liệu được xử lý theo nguyên tắc hàng đợi (FIFO) theo cả hai đầu, nghĩa là ta có thể

* nạp dữ liệu vào đầu trái, lấy dữ liệu ra khỏi đầu phải của q, hoặc
* nạp dữ liệu vào đầu phải, lấy dữ liệu ra khỏi đầu trái của q

tùy theo yêu cầu của bài toán.

#### Các thao tác cơ bản trên dequeue

Trong Python, dequeue được cài đặt qua kiểu dữ liệu danh sách (list) như sau:

* Khởi tạo hàng đợi hai đầu rỗng: qq = []
* Nạp đối tượng x vào cuối (đầu phải) qq: qq.append(): đối tượng x sẽ được nạp vào cuối qq, dung lượng của qq sẽ được tăng thêm 1 đơn vị.
* Lấy đối tượng ra khỏi cuối (đầu phải) qq: x = qq.pop(), dung lượng của qq sẽ được giảm bớt 1 đơn vị.
* Nạp đối tượng x vào đầu (trái) qq: qq.insert(0,x): đối tượng x sẽ được nạp vào đầu (trái) qq, dung lượng của qq sẽ được tăng thêm 1 đơn vị.
* Lấy đối tượng ra khỏi đầu (trái) qq: x = qq.pop(0), dung lượng của qq sẽ được giảm bớt 1 đơn vị.
* Xem phần tử cuối (đầu phải) qq: qq[-1]
* Xem phần tử đầu (trái) qq: qq[0]
* Kiểm tra hàng đợi qq rỗng? len(qq) == 0
* Hiển thị nội dung hàng đợi: print(qq)

### Josephus with Multikey

*Chúng ta làm khó thêm bài Josephus như sau. Thay vì chọn một giá trị k để xác định người thứ k trong vòng tròn sẽ bị loại khỏi vòng tròn ta xét một dãy số nguyên key gọi là khóa. Mỗi lần thực hiện phép đếm-loại ta lấy tuần tự theo vòng tròn một giá trị k trong dãy khóa key. Nếu giá trị k là số dương thì ta sẽ đếm theo chiều kim đồng hồ k-1 vị trí, loại người thứ k khỏi vòng tròn rồi tạm dừng tại vị trí sát trái người vừa bị loại. Ngược lại, khi k < 0 thì ta sẽ đếm ngược chiều kim đồng hồ k-1 vị trí, loại người thứ k khỏi vòng tròn rồi tạm dừng tại vị trí sát phải người vừa bị loại. Người cuối cùng còn lại sẽ là người thắng cuộc.*

#### Algorithm

Với một dequeue qq ta giải bài này như sau:

Init: qq = [1, 2,...,n]

lặp n-1 lần:

lấy k tiếp theo trong key

Nếu k > 0:

chuyển k-1 người từ đầu trái qq qua đầu phải

loại phần tử thứ k (tại đầu trái qq)

Nếu k < 0:

k = abs(k)-1

chuyển k-1 phần tử cuối qq về đầu

xóa phần tử cuối qq

chuyển 1 pt cuối qq về đầu làm điểm tạm dừng

#### Minh họa n = 8, key = [3, -4, 2, -3]

Init qq = [1, 2, 3, 4, 5, 6, 7, 8]

k=3: qq = [4, 5, 6, 7, 8, 1, 2], xóa 3

k=-4: qq = [7, 1, 2, 4, 5, 6], xóa 8

k=2: qq = [2, 4, 5, 6, 7], xóa 1

k=-3: qq = [5, 7, 2, 4], xóa 6

k=3: qq = [4, 5, 7], xóa 2

k=-4: qq = [7, 5], xóa 4

k=2: qq = [7], xóa 5

#### Program

# Josephus Problem with Multikey

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def JP(n, key):

m = len(key)

if n == 1: return n

qq = list(range(1, n+1))

print('Init:', qq)

j = -1 # index for key

for i in range(1, n):

j = (j+1) % m

k, x = key[j], 0 # k:khóa, x:phần tử cần xóa

if k > 0:

# move k-1 elements from head to end

for v in range(1,k): qq.append(qq.pop(0))

x = qq.pop(0) # delete head

else: # k < 0

k1 = abs(k)-1

# Move k1-1 elements from end to the head

for v in range(1, k1): qq.insert(0,qq.pop())

x = qq.pop() # delete the end

# Move one element from end to the head

qq.insert(0,qq.pop())

print('k =',k, ' delete', x)

print(qq)

return qq.pop()

# APPLICATION

print(JP(8, [3, -4, 2, -3]))

print(" T h e E n d")

#### Output

Init: [1, 2, 3, 4, 5, 6, 7, 8]

k = 3 delete 3

[4, 5, 6, 7, 8, 1, 2]

k = -4 delete 8

[7, 1, 2, 4, 5, 6]

k = 2 delete 1

[2, 4, 5, 6, 7]

k = -3 delete 6

[5, 7, 2, 4]

k = 3 delete 2

[4, 5, 7]

k = -4 delete 4

[7, 5]

k = 2 delete 5

[7]

7

T h e E n d

#### Độ phức tạp

O(n)

### Josephus Cypher

Điều thú vị là bài Josephus with multikey gợi ý cho chúng ta xây dựng một loại *mật mã* mới. Mỗi loại mật mã gồm hai hàm: mã hóa và giải mã.

Hàm mã hóa Encode(s, key) nhận vào một string cần mã hoa theo dãy khóa key. Hàm cho ra string w là dãy các kí tự bị xóa khỏi s trong trò chơi Josephus with multikey. Ta gọi s là *bản rõ*, w là *bản mật*.

Hàm giải mã Decode(w, key) nhận vào bản mật w và dãy khóa key. Hàm cho ra bản rõ s. Dễ thấy hai hàm này thực hiện các thao tác xuôi-ngược nhau.

Vậy ta chỉ cần một hàn JP(s, key, code=1) với tham biến ngầm định code bao quát cả hai trường hợp. Nếu code = 1 thì ta có hàm Encode; ngược lại, khi code != 1 thì ta có hàm Decode.

Hàm JP(s, key) sẽ xếp các đối tượng trong s thành vòng tròn và lần lượt đếm theo khóa k trong key để chuyển mỗi đối tượng cần loại khỏi vòng tròn sang string w.

# Josephus Cypher

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def JP(s, key, code = 1):

# code = 1: encoding; code != 1: decoding

n, m = len(s), len(key)

if n == 1 or m == 0: return s

qq = list(range(n))

w = ['#']\*n

j = -1 # index for key

for i in range(n-1):

j = (j+1) % m

k, x = key[j], 0 # k:khóa, x:phần tử cần xóa

if k > 0:

# move k-1 elements from head to end

for v in range(1,k): qq.append(qq.pop(0))

x = qq.pop(0) # delete head

else: # k < 0

k = -k-1

# Move k-1 elements from end to the head

for v in range(1, k): qq.insert(0,qq.pop())

x = qq.pop() # delete the end

# Move one element from end to the head

qq.insert(0,qq.pop())

if code: w[i] = s[x]

else: w[x] = s[i]

# The last element

x = qq[0]

if code: w[n-1] = s[x]

else: w[x] = s[n-1]

return ''.join(w)

# APPLICATION

s = 'Con co bay la bay la \n Bay tu cua phu bay ra canh dong...'

print('Input:', s)

key = [3, -4, 2, -3]

w = JP(s, key)

print('Encoding:', w)

u = JP(w, key, 0)

print('Decoding:', u)

print(" T h e E n d")

#### Output

Input: Con co bay la bay la

Bay tu cua phu bay ra canh dong...

Encoding: n.C.ongd.h ao crny bau paa chu uay t

lBy baa aay lcob

Decoding: Con co bay la bay la

Bay tu cua phu bay ra canh dong...

T h e E n d