算法竞赛笔记

Chunyin Chan

2025 年 2 月 21 日

目录

1 基础数据结构

1.1 link list in Python:

```
class nodes:
  def __init__(self, val=None, pre=None, next=None):
    self.val = val
    self.next = next
```

EX 1

洛谷 P1996 约瑟夫问题

题目描述

n 个人围成一圈，从第一个人开始报数, 数到 m 的人出列，再由下一个人重新从 1 开始报数，数到 m 的人再出圈，依次类推，直到所有的人都出圈，请输出依次出圈人的编号。

注意：本题和《深入浅出-基础篇》上例题的表述稍有不同。书上表述是给出淘汰 n−1 名小朋友，而该题是全部出圈。

输入格式

输入两个整数 n,m。

输出格式

输出一行 n 个整数，按顺序输出每个出圈人的编号。

输入输出样例

输入 #1

10 3

输出 #1

3 6 9 2 7 1 8 5 10 4

说明/提示

1 m,n 100

```
# 动态链表
class node:
  data = None
  next = None

n, m = map(int, input().split())
# init link list
head = node()
head.data = 1
now = head
for i in range(2, n+1):
  p = node()
  p.data = i
  now.next = p
  now = p
now.next = head
prev = now
now = head
while n > 0:
  n -= 1
  for i in range(m-1):
    prev = now
    now = now.next
  print(now.data, end='␣')
  prev.next = now.next
  now = now.next
```

```
# 列表和索引计算
n, m = map(int, input().split())
```

```
    people = list(range(1, n+1))
    rst = []
    current = 0
    index = 0
    while people:
      index = (current + m - 1) % len(people)
      rst.append(people.pop(index))
      current = index
    print('␣'.join(map(str, rst)))
```

EX 2

洛谷 P1160 队列安排

题目描述

一个学校里老师要将班上 N 个同学排成一列，同学被编号为 $1 \sim N$，他采取如下的方法：

1. 先将 1 号同学安排进队列，这时队列中只有他一个人；

2. $2 \sim N$ 号同学依次入列，编号为 i 的同学入列方式为：老师指定编号为 i 的同学站在编号为 $1 \sim (i-1)$ 中某位同学（即之前已经入列的同学）的左边或右边；

3. 从队列中去掉 M 个同学，其他同学位置顺序不变。

在所有同学按照上述方法队列排列完毕后，老师想知道从左到右所有同学的编号。

输入格式

第一行一个整数 N，表示了有 N 个同学。

第 $2 \sim N$ 行，第 i 行包含两个整数 k, p，其中 k 为小于 i 的正整数，p 为 0 或者 1。若 p 为 0，则表示将 i 号同学插入到 k 号同学的左边，p 为 1 则表示插入到右边。

第 $N+1$ 行为一个整数 M，表示去掉的同学数目。

接下来 M 行，每行一个正整数 x，表示将 x 号同学从队列中移去，如果 x 号同学已经不在队列中则忽略这一条指令。

输出格式

一行，包含最多 N 个空格隔开的整数，表示了队列从左到右所有同学的编号。

输入输出样例 #1

输入 #1

4 1 0 2 1 1 0 2 3 3

输出 #1

2 4 1

说明/提示

【样例解释】

将同学 2 插入至同学 1 左边，此时队列为：

2 1

将同学 3 插入至同学 2 右边，此时队列为：

2 3 1

将同学 4 插入至同学 1 左边，此时队列为：

2 3 4 1

将同学 3 从队列中移出，此时队列为：

2 4 1

同学 3 已经不在队列中，忽略最后一条指令

最终队列：

2 4 1

【数据范围】

对于 20% 的数据，$1 \le N \le 10$。

对于 40% 的数据，$1 \le N \le 1000$。

对于 100% 的数据，$1 < M \leq N \leq 10^5$。

```
# 链表
class nodes:
  def __init__(self, val=None, next=None, prev=None):
    self.val = val
    self.next = next
    self.prev = prev

N = int(input())
people = nodes(1)
head = people
idx = {1: people}
for i in range(2, N+1):
  node = nodes(i)
  k, p = map(int, input().split())
  k = idx[k]
  if p == 0:
    if k.prev:
      k.prev.next = node
      node.prev = k.prev
    node.next = k
    k.prev = node
    if k == head:
      head = node
  elif p == 1:
    if k.next:
      k.next.prev = node
      node.next = k.next
    node.prev = k
    k.next = node
  idx[i] = node

M = int(input())
for i in range(M):
  x = int(input())
  if x in idx.keys():
    k = idx[x]
    if k.prev:
      k.prev.next = k.next
      if k.next:
        k.next.prev = k.prev
    if k.next:
      k.next.prev = k.prev
      if k.prev:
        k.prev.next = k.next
      if k == head:
        head = k.next
    del idx[x]

while head:
  print(head.val, end='␣')
  head = head.next
```

1.2 Queue in Python

1.2.1 双端队列

```
from collections import deque # 双端队列

queue = deque(maxlen = 10) # 最大长度为10, None为无限制
queue = deque(iterable) # init queue by iterable obj.

```

```
queue.append(x) # add x to right side
queue.appendleft(x) # add x to left side

queue.extend(iterable) # extend right side by iterable obj.
queue.extendleft(iterable) # extend the left side by iterable obj.

queue.pop() # pop element from right side
queue.popleft() # pop element from left side

queue.remove(x) # remove x from Queue, raise error if not found
queue.clear() # clear Queue

queue.reverse() # reverse Queue
queue.insert(i, x) # insert x into queue at position i
queue.count(x) # count the number of queue elements = x
queue.index(x) # return first match position, raise error if not found
```

EX 1

洛谷 P1540 [NOIP 2010 提高组] 机器翻译

题目背景

NOIP2010 提高组 T1

题目描述

小晨的电脑上安装了一个机器翻译软件，他经常用这个软件来翻译英语文章。

这个翻译软件的原理很简单，它只是从头到尾，依次将每个英文单词用对应的中文含义来替换。对于每个英文单词，软件会先在内存中查找这个单词的中文含义，如果内存中有，软件就会用它进行翻译；如果内存中没有，软件就会在外存中的词典内查找，查出单词的中文含义然后翻译，并将这个单词和译义放入内存，以备后续的查找和翻译。

假设内存中有 M 个单元，每单元能存放一个单词和译义。每当软件将一个新单词存入内存前，如果当前内存中已存入的单词数不超过 $M-1$，软件会将新单词存入一个未使用的内存单元；若内存中已存入 M 个单词，软件会清空最早进入内存的那个单词，腾出单元来，存放新单词。

假设一篇英语文章的长度为 N 个单词。给定这篇待译文章，翻译软件需要去外存查找多少次词典？假设在翻译开始前，内存中没有任何单词。

输入格式

共 2 行。每行中两个数之间用一个空格隔开。

第一行为两个正整数 M, N，代表内存容量和文章的长度。

第二行为 N 个非负整数，按照文章的顺序，每个数（大小不超过 1000）代表一个英文单词。文章中两个单词是同一个单词，当且仅当它们对应的非负整数相同。

输出格式

一个整数，为软件需要查词典的次数。

输入输出样例 #1

输入 #1

3 7 1 2 1 5 4 4 1

输出 #1

5

说明/提示

样例解释

整个查字典过程如下：每行表示一个单词的翻译，冒号前为本次翻译后的内存状况：

1. 1：查找单词 1 并调入内存。2. 1 2：查找单词 2 并调入内存。3. 1 2：在内存中找到单词 1。4. 1 2 5：查找单词 5 并调入内存。5. 2 5 4：查找单词 4 并调入内存替代单词 1。6. 2 5 4：在内存中找到单词 4。7. 5 4 1：查找单词 1 并调入内存替代单词 2。

共计查了 5 次词典。

数据范围

- 对于 10% 的数据有 $M = 1$，$N \leq 5$；- 对于 100% 的数据有 $1 \leq M \leq 100$，$1 \leq N \leq 1000$。

```
# 队列 collections.deque
from collections import deque
from array import array

M, N = map(int, input().split())
q = deque(maxlen=M) # 双向队列
qs = set() # 用作哈希表
count = 0 # 计数
inp_ary = array('i', map(int, input().split())) # 输入数据
l = len(inp_ary)
for i in range(l):
  if inp_ary[i] not in qs:
    if len(q) == M:
      qs.remove(q.popleft())
    qs.add(inp_ary[i])
    q.append(inp_ary[i])
    count += 1
print(count)
```

1.2.2 单调队列

洛谷 P1886 滑动窗口 /【模板】单调队列

题目描述

有一个长为 n 的序列 a，以及一个大小为 k 的窗口。现在这个从左边开始向右滑动，每次滑动一个单位，求出每次滑动后窗口中的最大值和最小值。

例如，对于序列 $[1,3,-1,-3,5,3,6,7]$ 以及 $k=3$，有如下过程：

窗口位置	最小值	最大值
[1 3 -1] -3 5 3 6 7	-1	3
1 [3 -1 -3] 5 3 6 7	-3	3
1 3 [-1 -3 5] 3 6 7	-3	5
1 3 -1 [-3 5 3] 6 7	-3	5
1 3 -1 -3 [5 3 6] 7	3	6
1 3 -1 -3 5 [3 6 7]	3	7

输入格式

输入一共有两行，第一行有两个正整数 n,k。第二行 n 个整数，表示序列 a

输出格式

输出共两行，第一行为每次窗口滑动的最小值第二行为每次窗口滑动的最大值

输入输出样例 #1

输入 #1

8 3 1 3 -1 -3 5 3 6 7

输出 #1

-1 -3 -3 -3 3 3 3 3 5 5 6 7

说明/提示

【数据范围】对于 50% 的数据，$1 \leq n \leq 10^5$；对于 100% 的数据，$1 \leq k \leq n \leq 10^6$，$a_i \in [-2^{31}, 2^{31})$。

```
from collections import deque
from array import array
import sys
```

```
4
5   # input
6   n, k = map(int, input().split())
7   a = list(map(int, input().split()))
8
9   min_q = deque()
10  max_q = deque()
11  min_rst = []
12  max_rst = []
13
14  for i in range(n):
15    # minimum queue
16    while min_q and a[min_q[-1]] > a[i]:
17      min_q.pop()
18    min_q.append(i)
19    # maximum queue
20    while max_q and a[max_q[-1]] < a[i]:
21      max_q.pop()
22    max_q.append(i)
23
24    # 开始记录
25    if i >= k - 1:
26      min_rst.append(a[min_q[0]])
27      max_rst.append(a[max_q[0]])
28
29    # 弹出窗口外的元素
30    if min_q and min_q[0] <= i - k + 1:
31      min_q.popleft()
32    if max_q and max_q[0] <= i - k + 1:
33      max_q.popleft()
34
35  # output
36  print('␣'.join(map(str, min_rst)))
37  print('␣'.join(map(str, max_rst)))
```

1.3 Stack in Python

使用 deque 作为栈.

hdu 1062
翻转字符串
input:
olleh !dlrow
output:
hello world!

```
1   from collections import deque
2     s = deque()
3     inp = list(input().split())
4     for item in inp:
5       # enter stack
6       for i in item:
7         s.append(i)
8       while len(s) > 0:
9         print(s.pop(), end='')
10      print('␣', end='')
```

1.3.1 单调栈

洛谷 P2947 [USACO09MAR] Look Up S

题目描述

Farmer John's N (1 <= N <= 100,000) cows, conveniently numbered 1..N, are once again standing in a row. Cow i has height H_i (1 <= H_i <= 1,000,000).

Each cow is looking to her left toward those with higher index numbers. We say that cow i 'looks up' to cow j if i < j and H_i < H_j. For each cow i, FJ would like to know the index of the first cow in line looked up to by cow i.

Note: about 50

约翰的 $N(1 \le N \le 10^5)$ 头奶牛站成一排，奶牛 i 的身高是 $H_i(1 \le H_i \le 10^6)$。现在，每只奶牛都在向右看齐。对于奶牛 i，如果奶牛 j 满足 $i < j$ 且 $H_i < H_j$，我们可以说奶牛 i 可以仰望奶牛 j。求出每只奶牛离她最近的仰望对象。

Input

输入格式

1. Line 1: A single integer: N

Lines 2..N+1: Line i+1 contains the single integer: H_i

第 1 行输入 N，之后每行输入一个身高 H_i。

输出格式

Lines 1..N: Line i contains a single integer representing the smallest index of a cow up to which cow i looks. If no such cow exists, print 0.

共 N 行，按顺序每行输出一只奶牛的最近仰望对象，如果没有仰望对象，输出 0。

输入输出样例 # 1

输入 # 1

```
6
3
2
6
1
1
2
```

输出 # 1

```
3
3
0
6
6
0
```

说明/提示

FJ has six cows of heights 3, 2, 6, 1, 1, and 2.

Cows 1 and 2 both look up to cow 3; cows 4 and 5 both look up to cow 6; and cows 3 and 6 do not look up to any cow.

【输入说明】6 头奶牛的身高分别为 3,2,6,1,1,2。

【输出说明】奶牛 #1,#2 仰望奶牛 #3，奶牛 #4,#5 仰望奶牛 #6，奶牛 #3 和 #6 没有仰望对象。

【数据规模】

对于 20% 的数据：$1 \le N \le 10$；

对于 50% 的数据：$1 \le N \le 10^3$；

对于 100% 的数据：$1 \le N \le 10^5, 1 \le H_i \le 10^6$。