

算法竞赛笔记

Chunyin Chan

2025 年 2 月 20 日

目录

1 基础数据结构	0
1.1 Link List	0
1.1.1 link list in python:	0
1.2 Queue	4
1.2.1 Queue in Python	4

1 基础数据结构

1.1 Link List

1.1.1 link list in python:

```
1 class nodes:
2     def __init__(self, val=None, pre=None, next=None):
3         self.val = val
4         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

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 #1

输入 #1

```
...
10 3
...
```

输出 #1

```
...
3 6 9 2 7 1 8 5 10 4
...
```

说明/提示

$1 \leq m, n \leq 100$

```
1  # 动态链表
2  class node:
3      data = None
4      next = None
5
6  n, m = map(int, input().split())
7  # init link list
8  head = node()
9  head.data = 1
10 now = head
11 for i in range(2, n+1):
12     p = node()
13     p.data = i
14     now.next = p
15     now = p
16 now.next = head
17 prev = now
18 now = head
19 while n > 0:
20     n -= 1
21     for i in range(m-1):
22         prev = now
23         now = now.next
24     print(now.data, end=' ')
25     prev.next = now.next
26     now = now.next
```

```
1  # 列表和索引计算
2  n, m = map(int, input().split())
3  people = list(range(1, n+1))
4  rst = []
5  current = 0
6  index = 0
7  while people:
8      index = (current + m - 1) % len(people)
9      rst.append(people.pop(index))
10     current = index
11 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 $1 \sim (i-1)$ 的同学之前或之后；
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

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

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， $1 \leq N \leq 10$ 。

对于 \$40\%\$ 的数据， $1 \leq N \leq 1000$ 。

对于 \$100\%\$ 的数据， $1 \leq M \leq N \leq 10^5$ 。

```
1  # 链表
2  class nodes:
3      def __init__(self, val=None, next=None, prev=None):
4          self.val = val
5          self.next = next
6          self.prev = prev
7
8  N = int(input())
9  people = nodes(1)
10 head = people
11 idx = {1: people}
12 for i in range(2, N+1):
13     node = nodes(i)
14     k, p = map(int, input().split())
15     k = idx[k]
16     if p == 0:
17         if k.prev:
18             k.prev.next = node
19             node.prev = k.prev
20         node.next = k
21         k.prev = node
22         if k == head:
23             head = node
24     elif p == 1:
25         if k.next:
26             k.next.prev = node
27             node.next = k.next
28         node.prev = k
29         k.next = node
30     idx[i] = node
31
32 M = int(input())
33 for i in range(M):
```

```
34     x = int(input())
35     if x in idx.keys():
36         k = idx[x]
37         if k.prev:
38             k.prev.next = k.next
39             if k.next:
40                 k.next.prev = k.prev
41         if k.next:
42             k.next.prev = k.prev
43             if k.prev:
44                 k.prev.next = k.next
45             if k == head:
46                 head = k.next
47     del idx[x]
48
49     while head:
50         print(head.val, end=' ')
51         head = head.next
```

1.2 Queue

1.2.1 Queue in Python

双端队列

```
1  from collections import deque # 双端队列
2
3  queue = deque(maxlen = 10) # 最大长度为10, None为无限制
4  queue = deque(iterable) # init queue by iterable obj.
5
6  queue.append(x) # add x to right side
7  queue.appendleft(x) # add x to left side
8
9  queue.extend(iterable) # extend right side by iterable obj.
10 queue.extendleft(iterable) # extend the left side by iterable obj.
11
12 queue.pop() # pop element from right side
13 queue.popleft() # pop element from left side
14
15 queue.remove(x) # remove x from Queue, raise error if not found
16 queue.clear() # clear Queue
17
18 queue.reverse() # reverse Queue
19 queue.insert(i, x) # insert x into queue at position i
20 queue.count(x) # count the number of queue elements = x
21 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该单词的译义，就不再存入，取而代之。如果单词不在内存中，则将新单词存入内存，以后遇到相同的单词时，就不再翻译了。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，当且仅当它们的大小相同。

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 5^5 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 $1 \leq M \leq 100$, $1 \leq N \leq 1000$ 。

```

1  # 队列 collections.deque
2  from collections import deque
3  from array import array
4
5  M, N = map(int, input().split())
6  q = deque(maxlen=M) # 双向队列
7  qs = set() # 用作哈希表
8  count = 0 # 计数
9  inp_ary = array('i', map(int, input().split())) # 输入数据
10 l = len(inp_ary)
11 for i in range(l):
12     if inp_ary[i] not in qs:
13         if len(q) == M:
14             qs.remove(q.popleft())
15             qs.add(inp_ary[i])
16             q.append(inp_ary[i])
17             count += 1
18 print(count)

```

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，求出每个位置

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

```

 $\def\arraystretch{1.2}$ 
\begin{array}{|c|c|c|}\hline
\textsf{窗口位置} & \textsf{最小值} & \textsf{最大值} \\ \hline
[1 \quad 3 \quad -1] & -3 & 3 \\ \hline
[3 \quad -1 \quad -3] & -3 & 3 \\ \hline
[1 \quad 3 \quad [-1 \quad -3 \quad 5]] & -3 & 5 \\ \hline
[1 \quad 3 \quad -1 \quad [-3 \quad 5 \quad 3]] & -3 & 5 \\ \hline
[1 \quad 3 \quad -1 \quad -3 \quad [5 \quad 3 \quad 6]] & 3 & 6 \\ \hline
[1 \quad 3 \quad -1 \quad -3 \quad 5 \quad [3 \quad 6 \quad 7]] & 3 & 7 \\ \hline
\end{array}

```

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， $1 \leq k \leq n \leq 10^6$ ， $a_i \in [-2^{31}, 2^{31})$ 。

```
1  from collections import deque  
2  from array import array  
3  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)))
```
