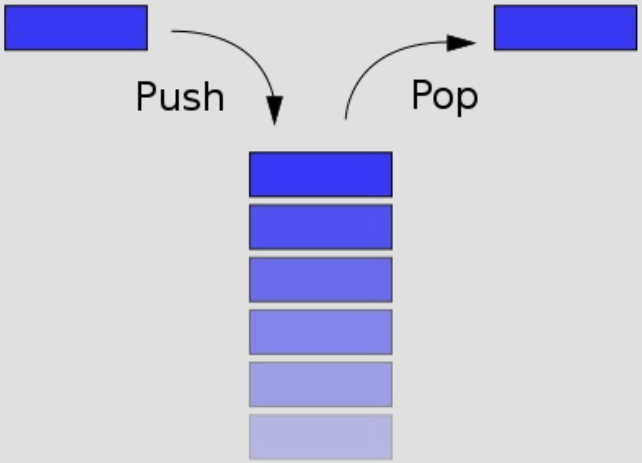
Stack, Queue, Linked List

Ghazal Minaei

Stack



خلاصه‌ای از پشته:

Push:

عنصر جدید را بالای پشته قرار می‌دهد.

Pop:

یک عنصر از سر پشته برمی‌دارد.

Top:

بالاترین عنصر را برمی‌گرداند.

مثال: بلندترین زیردنباله پرانتز گذاری صحیح

رشته‌ای از پرانتزها داده شده، طول بلندترین دنباله از پرانتز گذاری درست را بیابید.

ورودی و خروجی:

در تنها خط ورودی یک رشته از پرانتزها داده شده، در خروجی طول بلندترین دنباله از پرانتز گذاری درست را چاپ کنید.

مثال: بلندترین زیردنباله پرانتز گذاری صحیح

ورودی و خروجی نمونه:

Input: ()()()(
Output: 6

در مثال اول طولانی ترین پرانتز گذاری صحیح از اندیس ۰ تا ۵ یعنی ()() می باشد که طول آن یعنی ۶ در خروجی نمایش داده شده است.

Input: ((()))
Output: 6

در مثال دوم تنها پرانتز آخر نادرست است (باز نشده است) پس طول صحیح ۶ می شود.

مثال: بلندترین زیردنباله پرانتز گذاری صحیح

به ازای هر کاراکتر در رشته، اگر ")" بود اندیس فعلی را روی پشته قرار می‌دهیم. در غیر این صورت (یعنی کاراکتر "(" بوده) یک داده از روی پشته برمیداریم، اگر پشته خالی نبود طول بلندترین پرانتز گذاری صحیح را محاسبه می‌کنیم. برای این کار تفاوت بین اندیس فعلی و سر پشته را به دست می‌آوریم و اگر بیشتر از نتیجه قبلی بود آن را آپدیت می‌کنیم. اگر پشته خالی نشد اندیس فعلی را در پشته قرار می‌دهیم. این عدد مبنای محاسبات بعدی خواهد بود.

مثال: بلندترین زیردنباله پرانتز گذاری صحیح

```python
#for using list as stack:
#push -> stack.append
#top -> stack[-1]
#pop -> stack.pop
def findMaxLen(string):
    stack = []
    stack.append(-1)
    result = 0
    for i in range(len(string)):
        if string[i] == '(':
            stack.append(i)
        else:
            stack.pop()
            if stack:
                result = max(result, i - stack[len(stack)-1])
            else:
                stack.append(i)

    return result

print(findMaxLen(input()))
```

مثال: اعداد دودویی

یک الگوی ناقص از اعداد دودویی داده شده که بعضی از ارقام آن به صورت ? است. همه اعداد دودویی ممکن با این الگو را به صورت نزولی نمایش دهید.

ورودی و خروجی:

در یک خط یک رشته از الگوی دودویی به شما داده شده است. در خروجی تمامی حالات ممکن برای این الگو را به صورت نزولی نمایش دهید.

مثال: اعداد دودویی

ورودی و خروجی نمونه:

Input: 1?101?
Output: 111011 111010 101011 101010

دو علامت ? داریم پس ۴ الگو ممکن است که باید به ترتیب نزولی نمایش داده شود. جایگاه با ارزش‌تر را اول ۱ سپس ۰ می‌گذاریم که ترتیب نزولی حفظ شود برای ۱ دو حالت در جایگاه دوم داریم که به ترتیب ۰ و ۱ قرار می‌دهیم و همین روند را برای ۲ الگوی باقی مانده تکرار می‌کنیم.

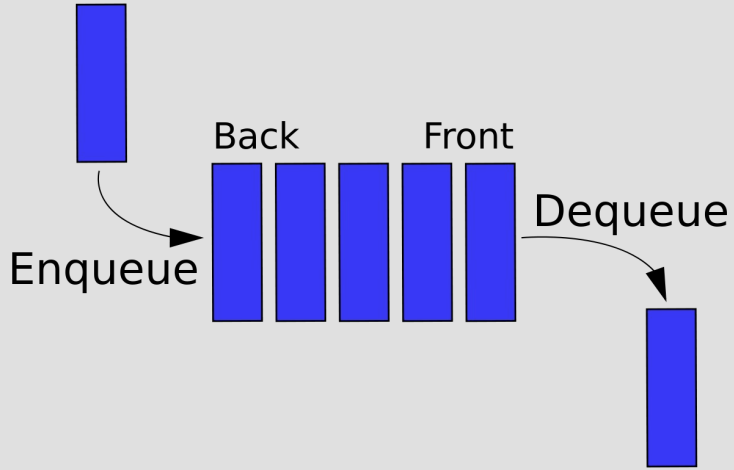
مثال: اعداد دودویی

از یک پشته استفاده می‌کنیم که داده‌های آن رشته‌های مختلف است. رشته ورودی را در پشته قرار می‌دهیم.
در هر مرحله بالاترین داده پشته را خارج کرده و بررسی می‌کنیم و اگر "?"، داشت چپ‌ترین "?" را در نظر می‌گیریم. دو تا رشته جدید که در اولی جای چپ‌ترین "?" مقدار 0 و در دومی 1 قرار دارد در پشته قرار می‌دهیم. اگر "?" در بالاترین رشته وجود نداشت آن را چاپ می‌کنیم.

مثال: اعداد دودویی

```
n = input()
stack = []
stack.append(n)
while stack:
  top = stack.pop()
  if '?' in top:
    stack.append(top.replace('?', '0', 1))
    stack.append(top.replace('?', '1', 1))
  else:
    print(top)
```

Queue



خلاصه‌ای از صف:

Enqueue:

عنصر جدید را انتهای صف قرار می‌دهد.

Dequeue:

یک عنصر از سر صف برمی‌دارد.

مثال: مرتب سازی صف

یک صف شامل اعداد ۱ تا n در ترتیب‌های تصادفی داده شده، بررسی کنید که آیا عناصر این صف را میتوان در یک صف کمکی به صورت صعودی با استفاده از یک پشته مرتب کرد یا نه. عملیات مجاز به صورت زیر می‌باشد:

pop و push عناصر پشته

dequeue عناصر از صف داده شده

enqueue عناصر در صف کمکی

نکته: دقت کنید enqueue به صف اول و dequeue از صف کمکی مجاز نیست.

ورودی و خروجی:

ورودی یک خط شامل داده‌های درون صف است. در تنها خط خروجی اگر مرتب سازی امکان پذیر بود "yes" و در غیر این صورت "no" چاپ کنید.

مثال: مرتب سازی صف

ورودی و خروجی نمونه:

Input: 5 1 2 3 4
Output: yes

این ورودی قابل مرتب‌سازی است پس در خروجی yes چاپ می‌کنیم. برای مرتب‌سازی کافی است این مراحل را طی کنیم:
۵ را درون پشته قرار می‌دهیم،
۱، ۲، ۳ و ۴ را نیز به ترتیب از صف اول خارج کرده و در صف جدید قرار می‌دهیم. در آخر ۵ را از سر پشته برداشته و در صف جدید قرار می‌گذاریم.

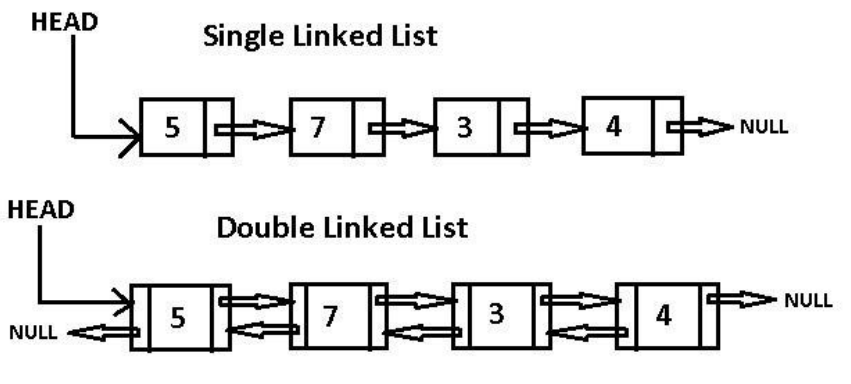
مثال: مرتب سازی صف

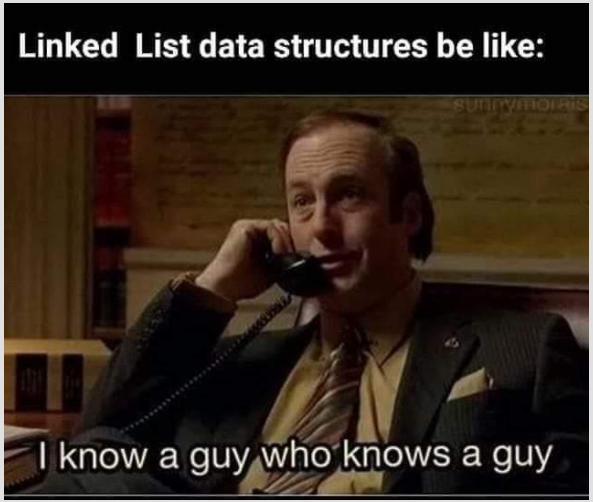
داده‌ها باید درون صف دوم قرار بگیرد پس همواره باید داده بعدی که انتظارش را داریم بالای پشته یا سر صف اول باشد. مقدار مورد انتظار اول کار ۱ می‌باشد، سپس تا زمانی که هنوز صف اول خالی نشده، بررسی می‌کنیم که داده سر صف اول یا بالای پشته مقدار مورد نظر را دارند یا خیر، اگر داشتند مقدار مورد انتظار را را یکی افزایش می‌دهیم. در غیر این صورت از صف یک داده خارج کرده و روی پشته قرار می‌دهیم. اگر این مقدار از سر پشته بیشتر بود مرتب سازی ممکن نیست.

مثال: مرتب سازی صف

```
from queue import Queue
def checkSorted(n, q):
    stack = []
    expected = 1
    front = None

    while (not q.empty()):
        front = q.queue[0]
        q.get()
        if (front == expected):
            expected += 1
        else:
            if (len(stack) == 0):
                stack.append(front)
            elif (len(stack) != 0 and stack[-1] < front):
                return False
            else:
                stack.append(front)

        while (len(stack) != 0 and
                stack[-1] == expected):
            stack.pop()
            expected += 1
    if (expected - 1 == n and len(stack) == 0):
        return True

    return False
```

```
inp = input().split()
q = Queue()
for el in inp:
    q.put(int(el))

n = q.qsize()

if checkSorted(n, q):
    print("Yes")
else:
    print("No")
```






```
while (guy):
    guy = guy.knownGuy
```

خلاصه‌ای از لیست پیوندی:

در لیست پیوندی داده‌ها در بخش‌های مختلف مموری قرار گرفته‌اند و با پوینتر بهم مرتبط می‌شوند.

در لیست پیوندی عملیات حذف داده و افزودن داده جدید در زمان ثابت انجام می‌گیرد. اما دسترسی تصادفی زمان‌بر است.

خلاصه‌ای از لیست پیوندی:

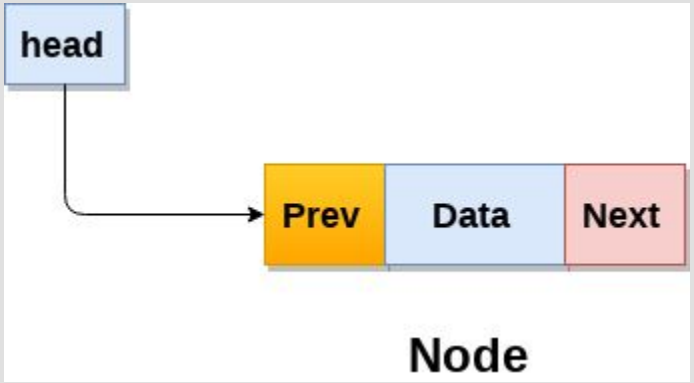
تعریف لیست پیوندی:

```
# Node class
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

# Linked List class
class LinkedList:
    def __init__(self):
        self.head = None
```

خلاصه‌ای از لیست پیوندی:

پیمایش:

```
def traverseList(self):
    node = self.head
    while (temp):
        #do something with node.data
        node = node.next
```

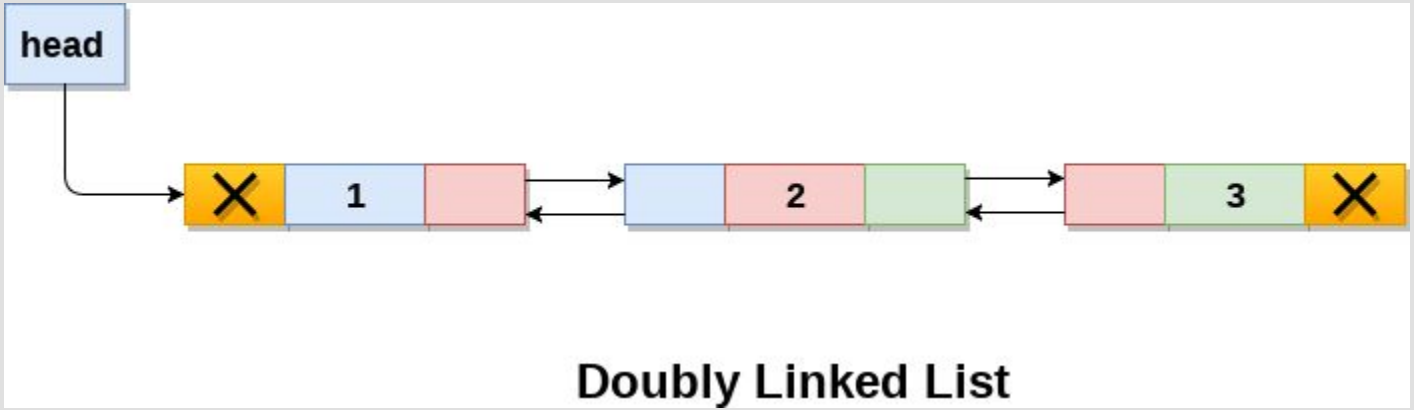


خلاصه‌ای از لیست پیوندی:

لیست پیوندی دوطرفه:

در لیست پیوندی دوطرفه هر نود یک پوینتر به داده قبلی خود نیز نگه می‌دارد.

لیست پیوندی دوطرفه از هر دو سمت قابل پیمایش است.



مثال: طول بلندترین دنباله palindrome

در یک لیست پیوندی طول بلندترین دنباله palindrome را بیابید. (استفاده از حافظه اضافی از اوردر ورودی مجاز نیست.)

ورودی و خروجی:

ورودی شامل یک خط، داده‌های درون لیست پیوندی است. در تنها خط خروجی طول بلندترین دنباله قرینه را چاپ کنید.

مثال: طول بلندترین دنباله palindrome

Input: 2 1 5 1 2 12 33
Output: 5

ورودی و خروجی نمونه:

بلندترین دنباله palindrome :

21512

است که طول آن ۵ می‌باشد.

مثال: طول بلندترین دنباله palindrome

برای اینکه از حافظه اضافی استفاده نکنیم، همه گره‌ها را پیمایش می‌کنیم و هر گره را برعکس می‌کنیم یعنی next آن از این به بعد به گره قبلی اشاره خواهد کرد. در هر مرحله طول بلندترین دنباله palindrome را محاسبه می‌کنیم. برای این کار لیست پیوندی ایجاد شده با head گره فعلی (که حالا برعکس است) را با لیست بعد از این گره مقایسه می‌کنیم. تابع countCommon این مقایسه را برعهده دارد.

برای مثال این الگوریتم را روی مثال داده شده اجرا می‌کنیم:

لیست پیوندی ابتدای کار به صورت روبرو است:

33 -> 12 -> 2 -> 1 -> 5 -> 1 -> 2

سپس گره اول برعکس می‌شود که به none اشاره می کند:

None <- 33 12 -> 2 -> 1 -> 5 -> 1 -> 2

سپس تابع countCommon را یک بار برای None و ۱۲ و یک بار هم برای ۳۳ و ۱۲ صدا می‌زنیم(یکی برای حالت فرد و دیگری زوج) که صفر برمی‌گرداند، سپس به سراغ گره بعدی می‌رویم و آن را برعکس می‌کنیم:

None <- 33 <- 12 2 -> 1 -> 5 -> 1 -> 2

تابع countCommon را یک بار برای ۳۳ و ۲ و یک بار هم برای ۳۳ و ۱۲ صدا می‌زنیم که صفر برمی‌گرداند، سپس به سراغ گره بعدی می‌رویم و آن را برعکس می‌کنیم:

None <- 33 <- 12 <-2 1 -> 5 -> 1 -> 2

مثال: طول بلندترین دنباله palindrome

تابع countCommon را یک بار برای ۱۲ و ۱ و یک بار هم برای ۲ و ۱ صدا می‌زنیم که باز هم صفر برمی‌گرداند، سپس به سراغ گره بعدی می‌رویم و آن را برعکس می‌کنیم:

None <- 33 <- 12 <-2 <- 1   5 -> 1 -> 2

تابع countCommon را یک بار برای ۲ و ۵ و یک بار هم برای ۱ و ۵ صدا می‌زنیم که باز هم صفر برمی‌گرداند، سپس به سراغ گره بعدی می‌رویم و آن را برعکس می‌کنیم:

None <- 33 <- 12 <-2 <- 1 <- 5   1 -> 2

تابع countCommon را یک بار برای ۱ و ۱ و یک بار هم برای ۵ و ۱ صدا می‌زنیم که اولی ۲ برمی‌گرداند آن را در ۲ ضرب کرده و با یک جمع می کنیم(چون حالت فرد است). این ۲ در خروجی تابع، نشان دهنده آن است که لیست پیوندی شروع شده از ۱ اول با لیست شروع شده از دومین ۱، ۲ تا داده یکسان دارد.

2 <- 1 <- 5   1 -> 2

سپس به سراغ گره بعدی می‌رویم و این روند را تکرار می‌کنیم تا به گره آخر برسیم. پاسخ ۵ خواهد بود.

مثال: طول بلندترین دنباله palindrome

```python
class Node:
    def __init__(self, data, next):
        self.data = data
        self.next = next

class LinkedList:
    def __init__(self):
        self.head = None

    def insert(self, newData):
        new_node = Node(newData, self.head)
        self.head = new_node

    def maxPalindrome(self) :
        result = 0
        prev = None
        current = self.head

        while (current != None) :
            next = current.next
            current.next = prev
            result = max(result, 2 * countCommon(prev, next) + 1)
            result = max(result, 2 * countCommon(current, next))
            prev = current
            current = next
        return result
```

```python
def countCommon(a, b) :
    count = 0
    while (a != None and b != None) :
        if (a.data == b.data) :
            count = count + 1
        else:
            break
        a = a.next
        b = b.next
    return count

def makeLinkedList(arr):
    linkedlist = LinkedList()
    for data in arr:
        linkedlist.insert(data)

    return linkedlist

inp = [int(x) for x in (input()).split(" ")]
linkedlist = makeLinkedList(inp)
print(linkedlist.maxPalindrome())
```