```
Stack-ToP(S)
1. if (S.toP == 0) then
2.         Return "underflow"
3.     else
4.         Return S[S.toP]
```

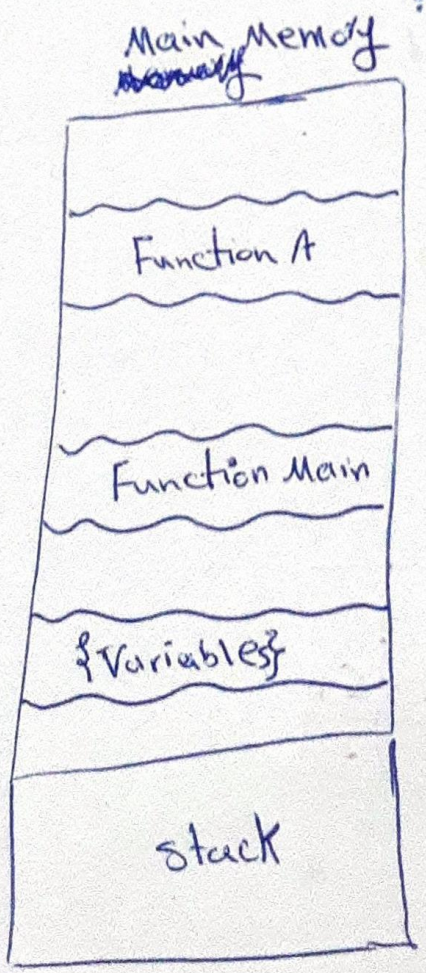
پیچیدگی زمانی الگوریتم فوق $O(1)$ است.

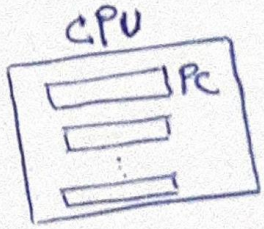
کاربردهای پشته:

۱) فراخوانی توابع: یکی از مهم‌ترین کاربردهای پشته است که در بخش‌های قبلی (الگوریتم‌های بازگشتی) تشریح کردیم.

یادآوری:

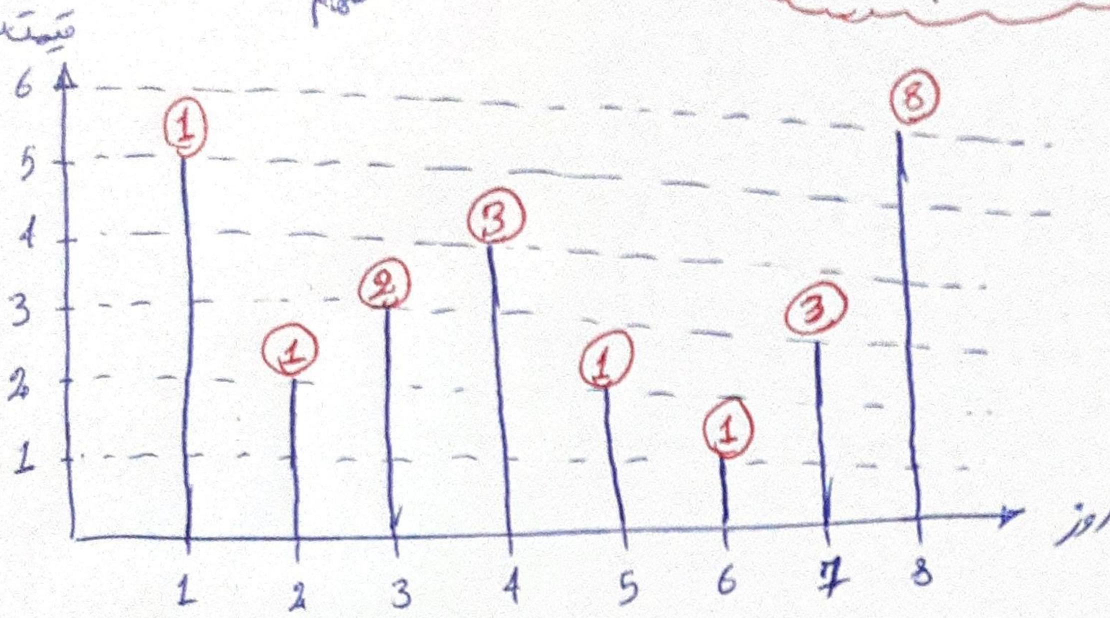
```
{Variables}
Function A
{
  ...
}
Function Main
{
  ...
  A()
  ...
}
```





PC: Program counter

۲. محاسبه دوره سهام روز i ام: یک لیست از (روز و قیمت سهام) داده شده است.



تعریف دوره سهام روز i ام: تعداد روزهای متوالی قبل از روز i ام که قیمت سهام آنها در بازه کوچکتر یا مساوی روز i ام هستند. لازم به ذکر است که خود روز i ام را هم برای این تعداد اضافه میکنیم.

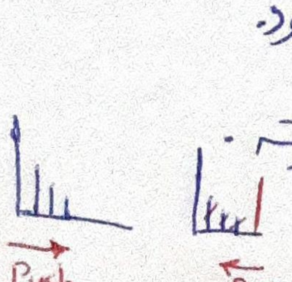
دوره سهام هر یک از روزها در شکل بالا مشخص شده اند.

سوال: یک لیست از طول n از (روز و قیمت سهام) داده شده است. دوره سهام هر یک از روزهای لیست را محاسبه کنید.

۱ راه حل ۱: از روز مورد نظر به عقب برمیگردیم و چک میکنیم شرایط دوره سهام را.

در این راه حل، اگر قیمت ها به صورت صعودی باشد، بدترین حالت اتفاق می افتد و پیچیدگی این راه حل $O(n^2)$ می شود.

۲ راه حل ۲: استفاده از پشته و نگه داشتن عناصر به ترتیب نزولی در پشته.

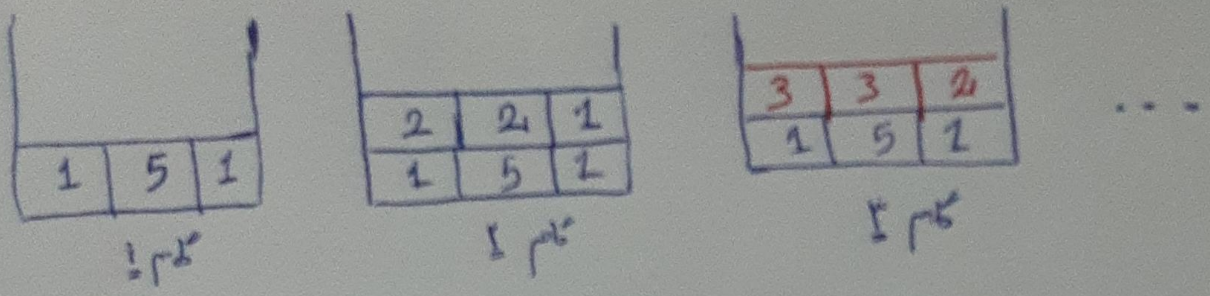


- روال Push و Pop:

- هر عنصر پشته:

دوره سهام	قیمت	روز

شکل زیر چند گام از راه حل ۲:



گام ۳، روز ۲ با دوره سهام ۱ از پشته خارج می‌شود و دوره سهام این روز به طور نهایی تعیین می‌شود و تاثیری هم در روزهای آتی ندارد. با این حال ۱ واحد برای روز سوم نیز محاسبه می‌شود (به خاطر Pop)

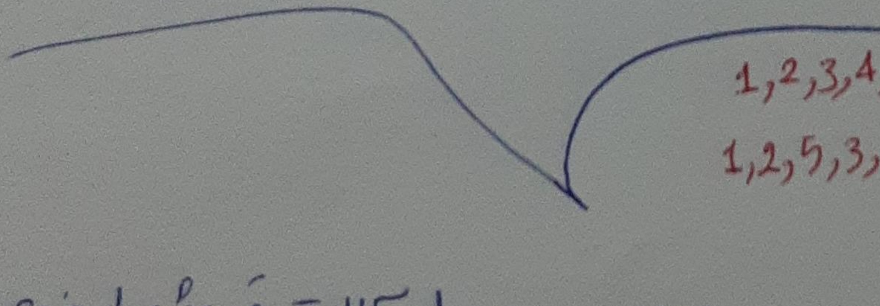
پیچیدگی زمانی راه حل دوم، از آنجاییکه هر عنصر یکبار در پشته اضافه می‌شود و یکبار هم از پشته حذف شده و سهام آن روز تعیین می‌شود، هزینه کلی $O(n)$ است.

۳- مساله پارکینگ قطارها:

یک لیست از ترتیب ورود قطارها به پارکینگ داریم. هر قطاری که وارد پارکینگ می‌شود امکان برگشت به ورودی پارکینگ را ندارد. از بین کسانی که در پارکینگ قرار دارند کسی که زودتر وارد پارکینگ شده، امکان خروج دارد.

مثال:

5, 4, 3, 2, 1



1, 2, 3, 4, 5 ✓

1, 2, 5, 3, 4 ✗

سوال: با توجه به قواعد و لیست داده شده برای ترتیب ورود قطارها، آیا لیست تعیین شده برای خروج قطارها از پارکینگ معتبر است.

پیاده‌سازی پشته ← از طریق لیست (آرایه ساده)،
← از طریق لیست پیوندی.

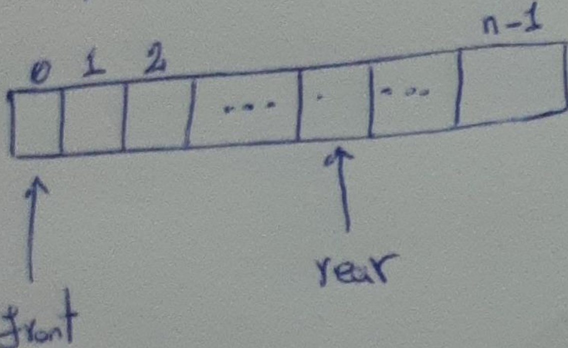
داده ساختار صف (Queue):

یک داده ساختار برای نگهداری مجموعه‌های پویاست که عناصر آن در یک ترتیب خطی قرار گرفته‌اند و برای حذف عناصر در آن از سیاست "اولین ورودی، اولین خروجی" استفاده می‌شود.

FIFO = First In First out

در ادامه دو نمایش گوناگون برای داده ساختار صف معرفی می‌کنیم:

نمایش ۱:



maxsize = n

در این نمایش:

* اگر front = rear باشد به این معناست که صف Q خالی است.
* اگر rear = maxsize باشد به این معناست که صف Q پر است.
* front به ابتدای صف اشاره می‌کند.
* rear به جایی اشاره می‌کند که داده جدید باید درج شود (انتهای صف).

عملیات روی صف - نمایش ۱:

Queue-EnQueue(Q,x): یک روال بروزرسانی است که عنصر x را به انتهای صف Q اضافه می‌کند. لازم به ذکر است که اگر صف پر باشد، پیام "overflow" برگردانده می‌شود.

```
Queue-Enqueue (Q, x)
1. if (Q.rear == Q.maxsize) then
2.     Return "overflow"
3. else
4.     Q[Q.rear] = x
5.     Q.rear = Q.rear + 1
```

پیچیدگی زمانی الگوریتم فوق $O(1)$ است.

Queue-Dequeue(Q): یک دنباله برنامه‌ای است که عنصر ابتدایی صف را حذف و برمی‌گرداند. لازم به ذکر است که اگر صف خالی باشد، پیام "underflow" برگردانده می‌شود.

```
Queue-Dequeue (Q)
1. if (front == rear) then
2.     Return "underflow"
3. else
4.     x = Q[Q.front]
5.     Q.front = Q.front + 1
6. Return x
```

مشکل نمایش ۱: پس از انجام تعدادی عمل حذف و درج در صف، با اینکه آرایه همچنان آزاد دارد، امکان درج عنصر جدید را نخواهیم داشت.