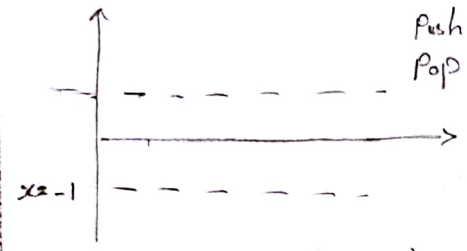
علی نظری ۹۹۴۱۰۷۵


```
Define StackA
Define StackB

function push(element):
    StackA.push(element)
    temp = StackB.pop()
    StackB.push(temp)
    if (StackB.isEmpty() or element < temp):
        StackB.push(element)

function pop():
    temp = StackB.pop()
    StackB.push(temp)
    element = StackA.pop()
    if (element == temp):
        StackB.pop()
    return element

function findMin():
    return StackB.pop()
```

در این الگوریتم که با ۲ استک پیاده‌سازی شده، استک B همواره کوچکترین مقدار استک A را در بالاترین مکان خود نگه می‌دارد تا با pop کردن آن به راحتی به آن دست یابیم.

(۶)

عملوند باید تعداد push ها از تعداد pop ها بیشتر باشد. اگر روی نمودار این وضعیت را مدل کنیم، باید جوری حرکت کنیم که محور $x=-1$ را قطع نکنیم. n بار باید push کنیم و n بار باید pop کنیم.



تعداد کل حالات: $\binom{2n}{n}$

حالات نامطلوب (pop n+1): $\binom{2n}{n+1}$

$\Rightarrow$ جواب: $\binom{2n}{n} - \binom{2n}{n+1}$

عدد کاتالان $= \dfrac{\binom{2n}{n}}{n+1} = \dfrac{2n!}{(n+1)!\,(n-1)!} = \dfrac{2n!}{n!\;n!}$

الف

ب) اگر سه عضو خیالی a, b, c را درون دنباله در نظر بگیریم به طوری که از نظر جایگاه در دنباله $[c] > [b] > [a]$ است، ما در خروجی به هیچ عنوان نمی‌توانیم ترتیب جدید زیر را در دنباله خروجی داشته باشیم: $[b] > [a] > [c]$ چون در این صورت یعنی a قبل از b pop شده که به خاطر pop شدن c قبل b و a، وقوع این رخداد غیرممکن است.

ج) دنباله‌ی $I[n]$ را دنباله ورودی و دنباله $O[n]$ را خروجی می‌گیریم.

```
i = 0
j = 0
for( i ; i<n ; i+=1 ):
    Stack.push( I[i] )
    while( Stack.top == O[j] )
        Stack.pop()
        j+=1

if( Stack.isEmpty ):
    return True
else:
    return False
```