جلسه هشتم

13:09 Sunday, October 17, 2021

داده ساختارهای ابتدایی : لیست و استک.

* داده ساختارهای پایه :

* آرایه : دنباله پشت سر هم از خانه‌های حافظه.

+ - به خانه با نام یک آرایه می‌توان در زمان O(1) دسترسی داشت.

- - طول آرایه ثابت است.

* مثال : ورودی : دنباله‌ای از دستورات add(x) و query(j)

add(x) : x را به انتهای دنباله اضافه کن

query(j) : عدد j ام دنباله را چاپ کن.

1 2 3 4

5 7 6 1000

add(5)

add(7)

add(6)

add(1000)

query(3) : 6

حداکثر تعداد addها 500 است.

* پیاده سازی با آرایه :

A : آرایه با اندازه 500

n : تعداد اعضای فعلی که add کردیم

add(x) : A[n]=x ، n++ O(1)

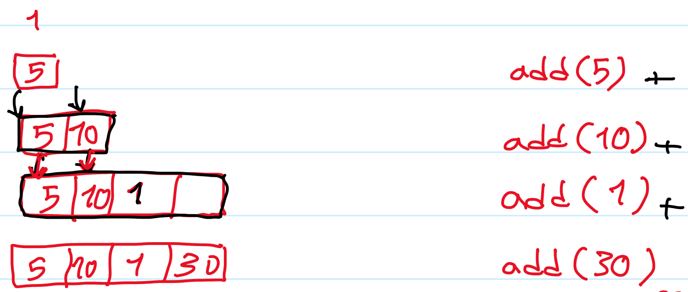
query(j) : return A[j] O(1)

سوال : اگر تعداد addها را ندانیم؟

* با یک آرایه با اندازه 1 شروع کنید.

* در هنگام add(x) ، اگر آرایه جای خالی داشت، آن را اضافه می‌کنیم و اگر آرایه پر بود، آنگاه یک آرایه با

* در هنگام add(x) ، اگر آرایه جای خالی داشت، آن را اضافه می‌کنیم و اگر آرایه پر بود، آنگاه یک آرایه با اندازه ۲ برابر می‌سازیم و تمام اعضای قبلی را به آرایه جدید add می‌کنیم و در نهایت عضو x را به آرایه جدید add می‌کنیم.



تعداد add ها (copy) بین آرایه‌ها + add به انتهای آرایه

* هزینه سرشکن هر عملیات add(x) برابر با $O(1)$ است.

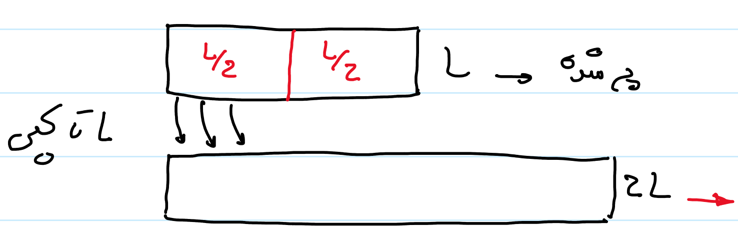
- روش حسابداری: ۱ ریال به ازای هر add یا کپی.

← add(x) : ۱ ریال هزینه را می‌دهیم.
۲ ریال داخل حساب ذخیره می‌کنیم.

← copy() : ۱ ریال از حساب پرداخت می‌کنیم.

** همیشه پول کافی در حساب برای کپی وجود دارد.

	پرداخت	حساب	
5	1	~~2~~ 1	add(5)
5 |	2	1+2=3	add(4)
⋮			



حساب؟ L/2 آخر هر کدام ۲ واحد پول در حساب ذخیره کردند.

در حساب حداقل L واحد پول است ... انجام دهیم

حساب ؟ $\frac{1}{2}$ اخر هر کدام $\frac{1}{2}$ واحد پول در حساب ذخیره کردند.

در حساب حداقل 1 واحد پول است ← می‌توانیم با آن 1 کپی انجام دهیم

* هر add ، 3 ریال ← هزینه سرشکن هر add ← $O(1)$

کپی 2 ← $\frac{1}{2}$ تا ← $\frac{1}{2}$

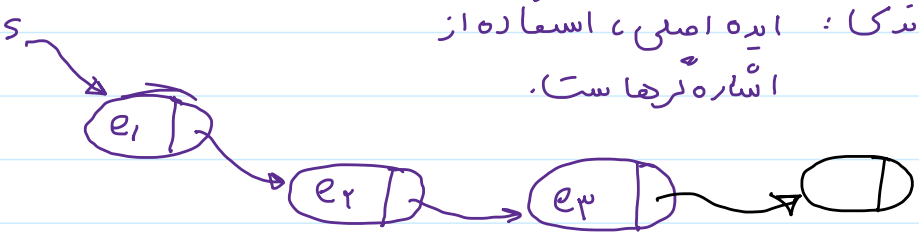
ساخت آرایه 4 ← برای ساخت آرایه ← $\frac{4}{2}$

$O(1)$ ← 7 ریال $= 4+2+1$

* سعی کنید این مثال را با تابع پتانسیل حل کنید.

«آرایه پویا»

* لیست پیوندی: ایده اصلی، استفاده از اشاره‌گرها است.

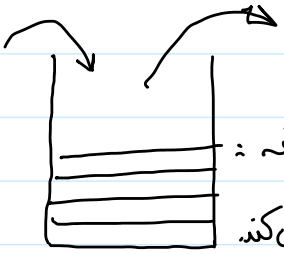


* طول لیست پیوندی نامحدود.
* دسترسی به عنصر kام لیست: $O(k)$ ← بدترین $O(n)$ (n طول لیست)

ADT : Abstract Data Type

به صورت مجرد در مورد داده ساختار و عملیاتی که نیاز دارد توضیح می‌دهد.

* ۱ استک (LIFO) stack



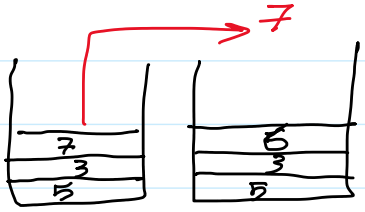
قیافه:

عملیات: Push(x): عنصر x را وارد استک می‌کند.

Pop: آخرین عنصر وارد شده در استک (بالای استک) را خارج و برمی‌گرداند.

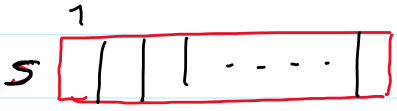
Top():

: top()
size()
: isEmpty()



مثال :
Push(5)
Push(3)
Push(7)
Pop()
Push(6)

۱- پیاده سازی با استفاده از آرایه:



t : تعداد عناصر داخل استک

t=0

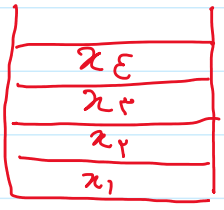
O(1)
```
Push(x){
  t++
  S[t]=x
}
```

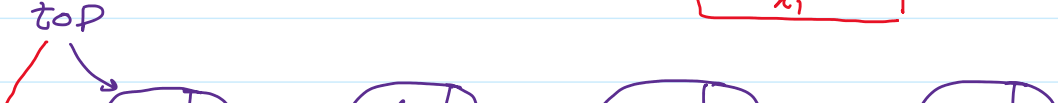
O(1)
```
Pop(){
  t--
  return S[t+1]
}
```

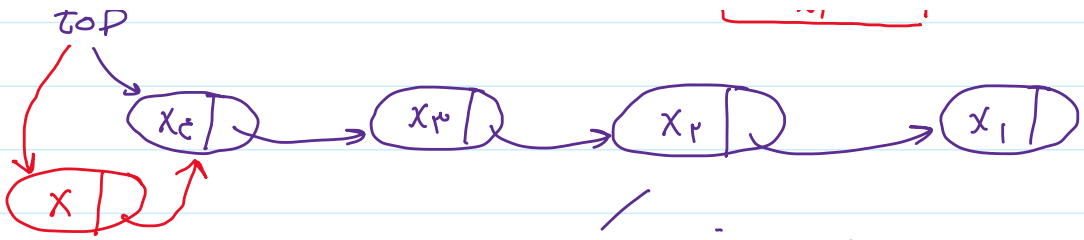
O(1) rutarn S[t] ← top()
O(1) t=?0 ← isEmpty()
O(1) t ← size()

* ایراد : اندازه محدود است ← آرایه پویا

۲- پیاده سازی با استفاده از لیست پیوندی.







t : تعداد عناصر داخل استک .

: push(x) O(1)

: pop() O(1)

} → O(1)

* محدودیت اندازه نداریم .

* صف Queue (FIFO)

→ Queue

عملیات :

: enqueue(x) } enq(x)
x را به انتهای صف اضافه می‌کند .

: dequeue() } deq()
عنصری که در ابتدا صف است را حذف می‌کند و برمی‌گرداند

: size()

: front()

: isEmpty()

7 4 5

enq(5)
enq(4)
enq(7)

7 4

7

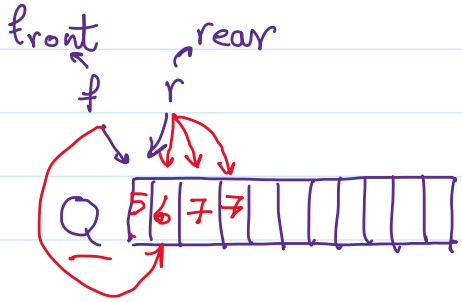
6 7

front rear

deq() ← 5 را حذف می‌کند
deq() ← 4 "
enq(6)

67

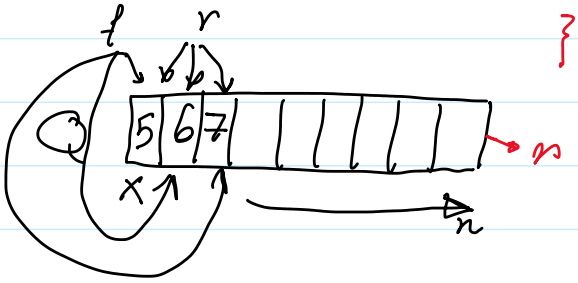
enq(6)



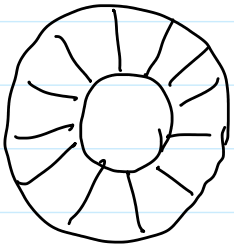
۱- پیاده سازی با استفاده از آرایه :

```
enq(x){
  Q[r] = x
  (r++)%n
}
```

```
deq(x){
  f++ %n
  return(Q[f-1])
}
```

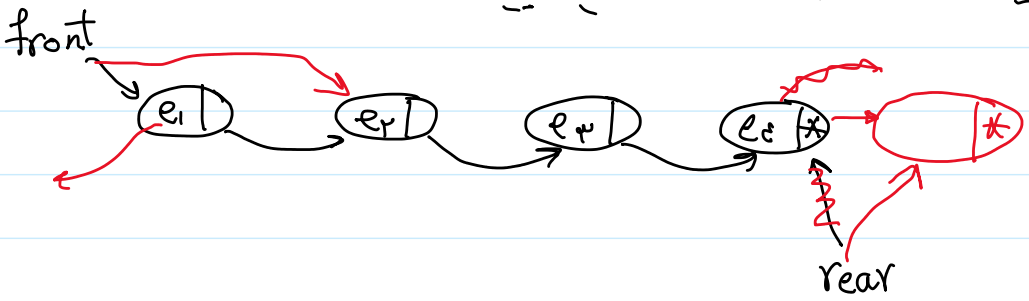


* مشکل : اندازه محدود .



برای حل این مشکل :

۳- پیاده سازی با استفاده از لیست پیوندی



O(1) Size
O(1) isEmpty
O(1) .

O(1) ← enq(e6)
O(1) ← deq()

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$$