ELEMI ADATSZERKEZETEK & ADATTÍPUSOK

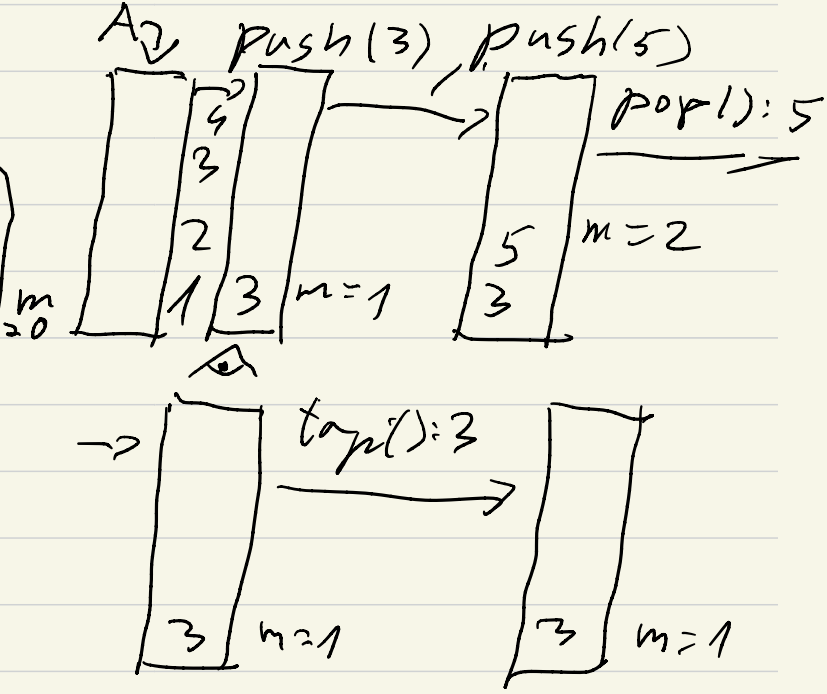
Stack

- A[]: T
- m: N
- constant m0: N+ := 16

+ Stack(max: N+ := m0) { A := new T[max]; m := 0 }
+ push(X: T)
+ pop(): T
+ top(): T
+ isEmpty(): B { return m = 0 }
+ setEmpty() { m := 0 }
+ ~Stack() { delete A }

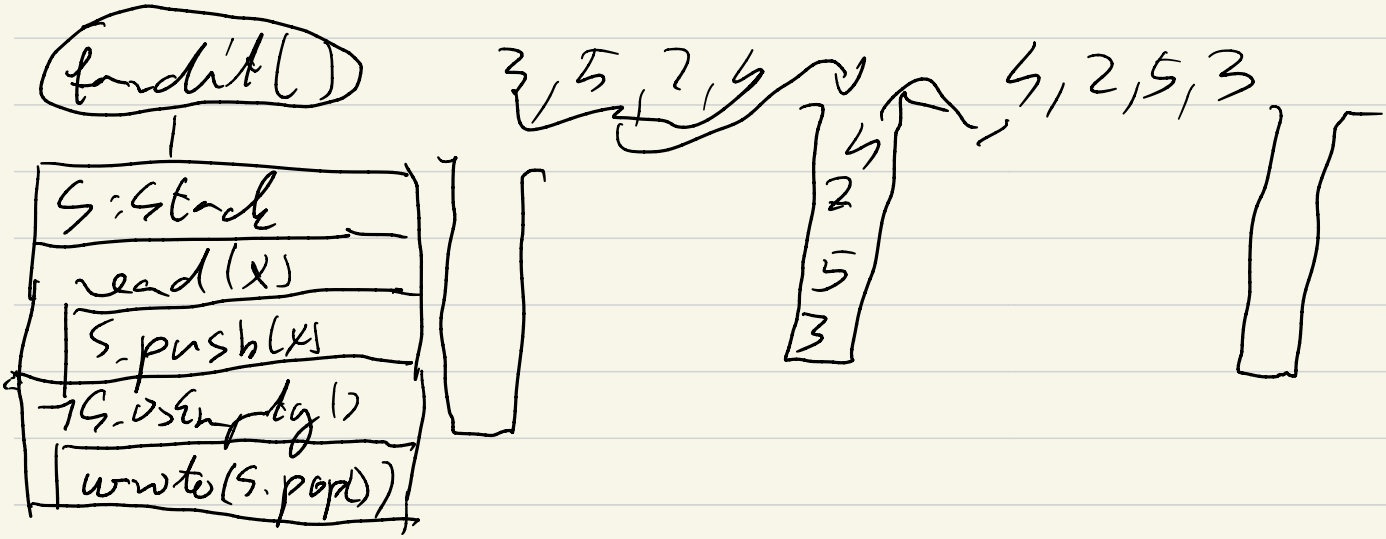
UML lista (Verem)

LIFO tároló



{S: Stack; ...} | $AT_{push}(m) \in \Theta(1)$, $|\Theta(1)|$ | hiv. push() $\begin{cases} mT \in \Theta(1) \\ MT(m) \in \Theta(m) \end{cases}$

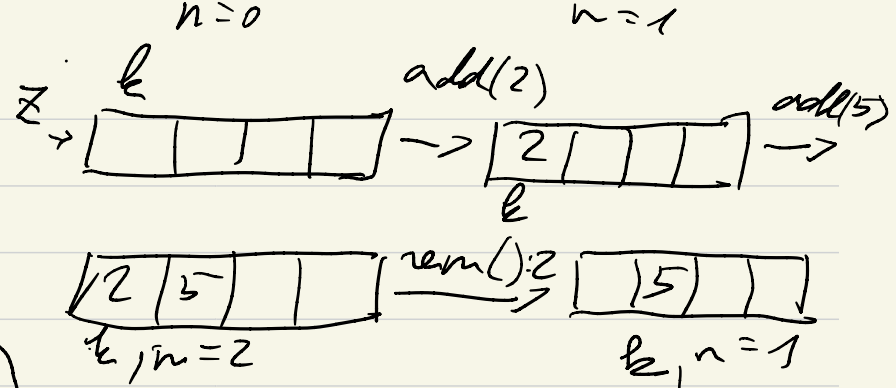
read(&x:T):B

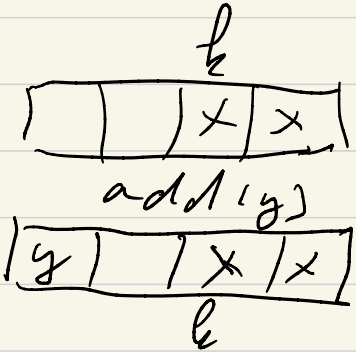

findit()
S:Stack
read(x)
S.push(x)
¬S.isEmpty()
write(S.pop())
3,5,2,4
4
2
5
3
4,2,5,3

Queue

- const $m_0 : \mathbb{N}_+ := 16$
- $Z : \mathcal{T}[\,]$
- $k, n : \mathbb{N}$

+ Queue($m : \mathbb{N}_+ := m_0$)
  {$Z := \text{new } \mathcal{T}[m]; k := n := 0$}
+ add($x : \mathcal{T}$)
+ rem() : $\mathcal{T}$
+ first() : $\mathcal{T}$
+ length() : $\mathbb{N}$ {return n}
+ setEmpty() {$k := n := 0$}
+ ~Queue() {delete Z}

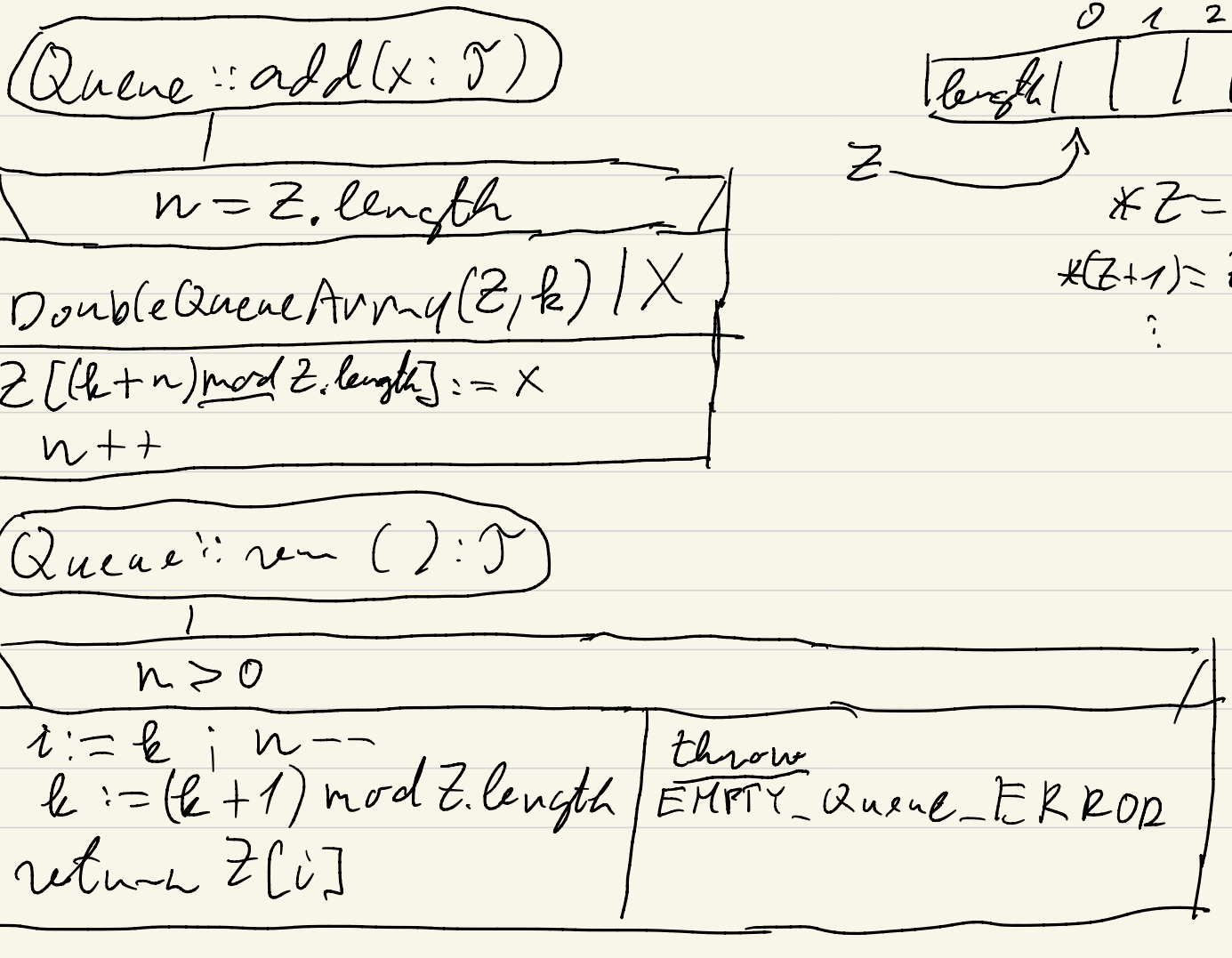




$\Theta(1)$

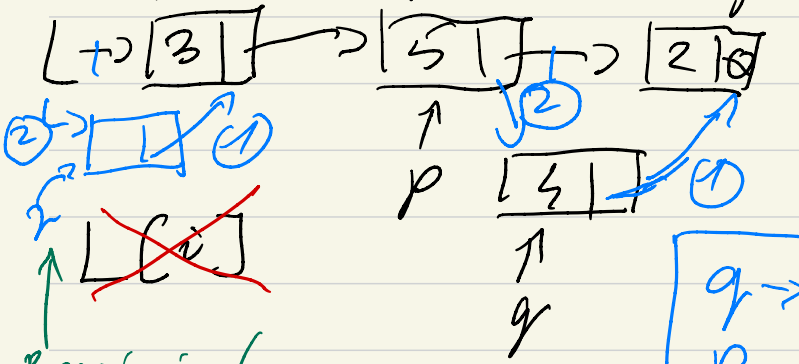
Műveletigény: $\Theta(1)$, bármely add(x)

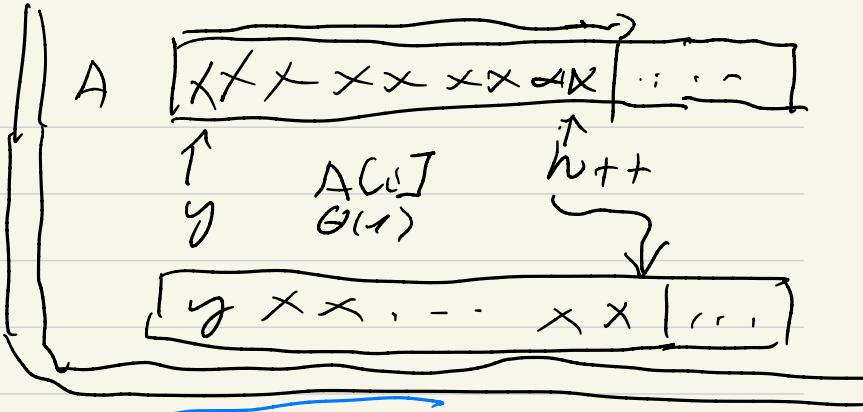
$\begin{cases} MT(n) \in \Theta(n) \\ AT(n) \in \Theta(1) \\ mT(n) \in \Theta(1) \end{cases}$ { AMORTIZÁLT műv. igény számítás


Queue::add(x:T)
n = Z.length
DoubleQueueArray(Z,k) | X
Z[(k+n) mod Z.length] := x
n++
0 1 2 .. Z.length-1
length
Z
*Z = Z[0]
*(Z+1) = Z[1]
Queue::rem():T
n > 0
i := k ; n--
k := (k+1) mod Z.length
return Z[i]
throw
EMPTY_Queue_ERROR

LÁNCOLT LISTÁK

<3, 5, 2> reprezentációja






```
q->next := p->next
p->next := q
```

Beszúrás(*p) után

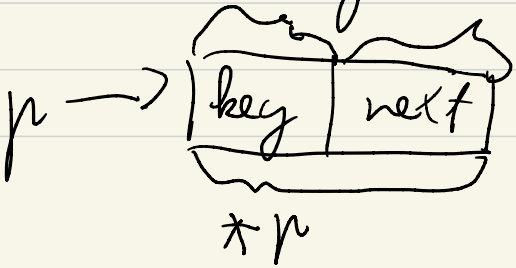
```
E1
+key: S
+next: E1*
:= Θ
```

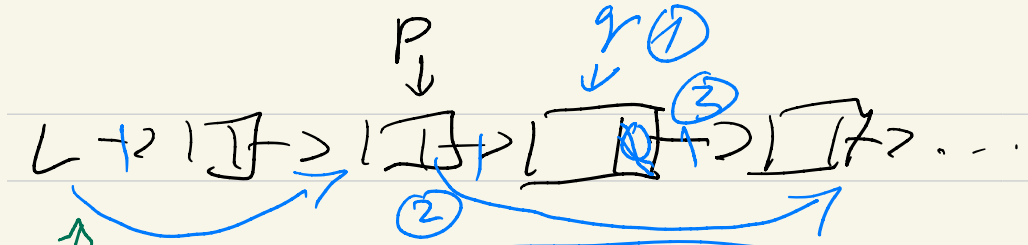
p->key

(*p).key

p->next

(*p).next





$p \rightarrow next \neq 0$

elem următor

```
q := p->next
p->next := q->next
q->next := 0
```

(*p) urmează elem următor

SLL - de (lines: $L = 0$)

Láncolt listák

egyirányú

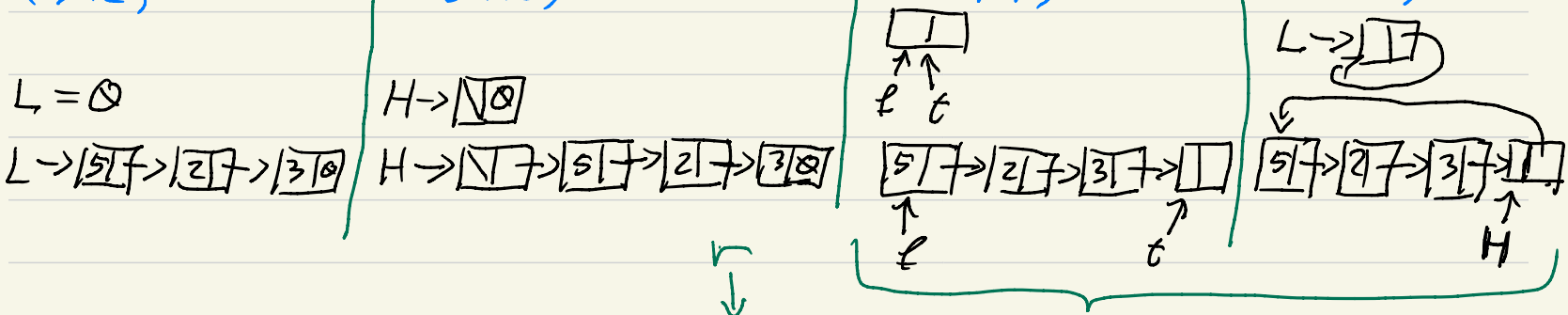
kétirányú

egyszerű (S1L)

fejelemes (H1L)

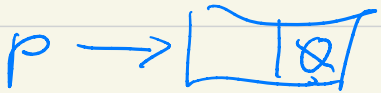
végelemes (1LT)

ciklikus (C1L)



SOK reprezentációjához

p := new E1

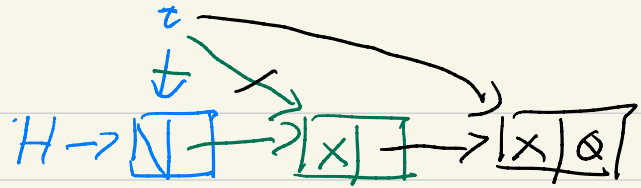


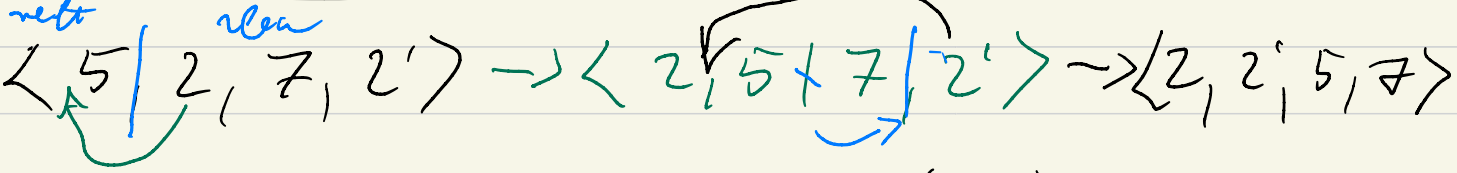
delete q; ...; r->next := s

???

read_HL():E14

```
t := H := new E1
read(x)
    t := t->next := new E1
    t->key := x
```

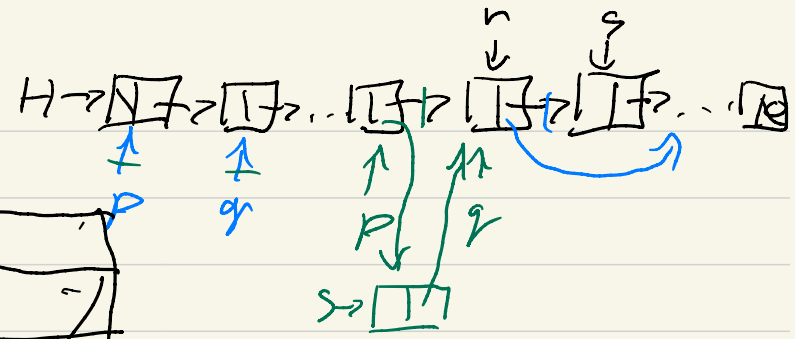




Insertion_sort (IS)

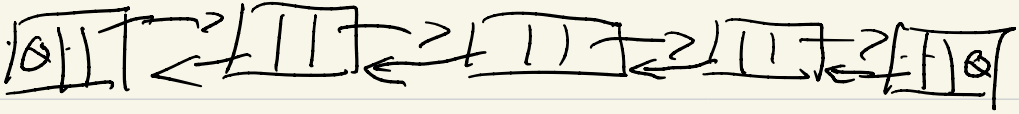
IS_HIL(H:E1*)

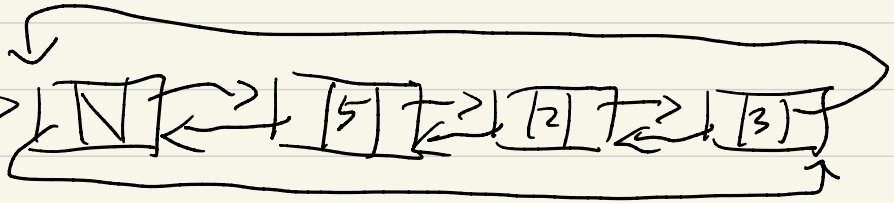
```
r := H->next
r ≠ Ω
s := r->next
s ≠ Ω
r->key ≤ s->key
r := s->next
q->key ≥ s->key
p := r ; q := q->next
s->next := q
p->next := s
s := r->next
```

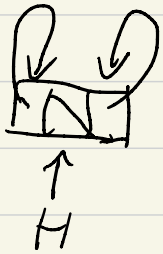


$mT(n) \in \Theta(n)$

$\left.\begin{matrix} MT(n) \\ AT(n) \end{matrix}\right\} \in \Theta(n^2)$

S2L L→ 

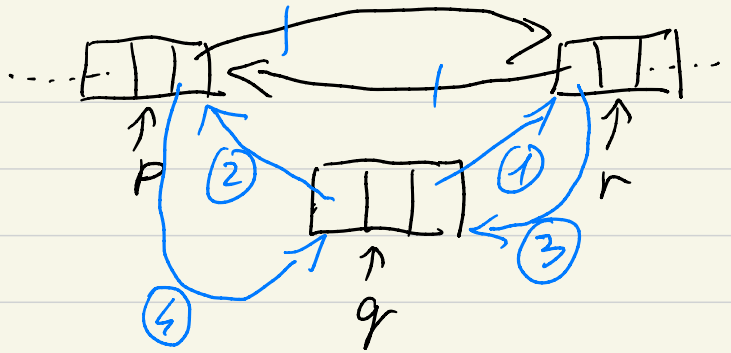
C2L H→ 



```
read_C2L():E2*
H:=new E2
read(x)
  p:=new E2
  p->key:=x
  precede(p,H)
```

```
precede(q,r:E2*)
p:=r->prev
insert(p,q,r)
```

```
E2
+key:ℑ
+next,prev:E2*
+E2(){
    next:=prev:=this
}
```

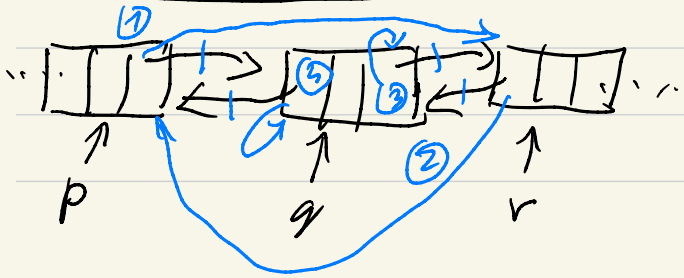


insert(p,q,r:E2*)

```
q->next:=r
q->prev:=p
r->prev:=q
p->next:=q
```

follow(p,q:E2*)

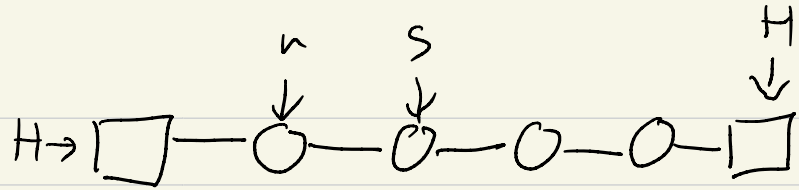
```
r:=p->next
insert(p,q,r)
```

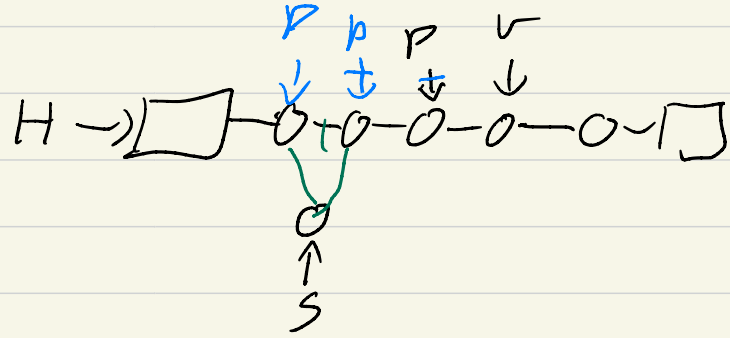


unlink(q:E2*)

```
p:=q->prev; r:=q->next
p->next:=r
r->prev:=p
q->prev:=q->next:=q
```

```
IS_CZL(H:EZ*)
r:=H->next; s:=r->next
s≠H
    r->key≤s->key
r       unlink(s)
:=      p:=r->prev
s       p≠H∧p->key>s->key
            p:=p->prev
        follow(p,s)
    s:=r->next
```





$mT(n) \in \Theta(n)$

$\left.\begin{matrix} MT(n) \\ AT(n) \end{matrix}\right\} \in \Theta(n^2)$