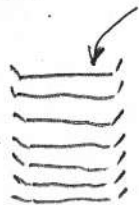


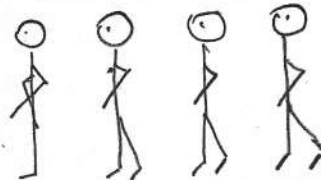
STOGovi I REDovi

dinamički skupovi koje nalazimo u svakodnevnom životu

stog tanjura



red klijenata u banci



tanjur stavljamo na vrh stoga } last-in first-out (LIFO)
-ili uzimamo s vrha stoga

klijent dolazi na kraj reda } first-in first-out (FIFO)
se uslužuje na početku reda

promatramo ih kao dinamičke skupove pa ćemo razmotriti osnovne operacije INSERT i DELETE.

STOGovi

insert operacija - Push

delete operacija - Pop

razlikujemo dvije implementacije stogova:

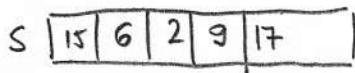
* pomoću nizova

* pomoću povezanih listi

Implementacija stoga pomoću polja



↑
S.top = 4




Push(

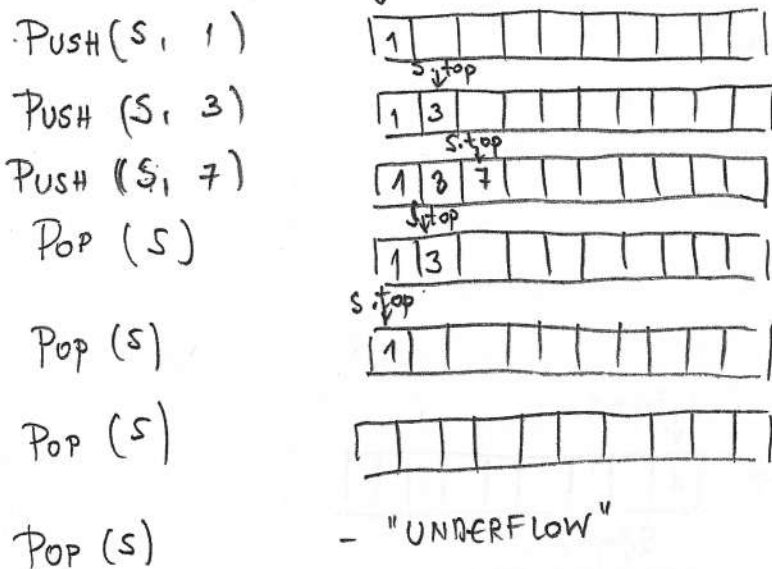
* gdje S duljine n može spremiti stog od najviše n elemenata

* S.top je indeks vrha stoga u polju S

- Stog S sastoji se od elemenata $S[1], \dots, S[\text{top}.S]$
- n je kapacitet polja S , koji ima ulogu spremnika elemenata stoga (zamislite kao da tanjure slažete u policu u koju može stati najviše n elemenata)
- veličinu stoga možemo doznati na osnovu indeksa vrha stoga $S.\text{top}$.
- ukoliko je $S.\text{top} = 0$, stog je prazan i svaka Pop operacija na praznom stogu dovest će do greške "underflow"
- ukoliko je $S.\text{top} = n$, stog je pun i svaka Push operacija na punom stogu dovest će do greške "overflow"

Primeri

S  - polje od 9 elemenata



- implementacija sljedećih operacija:
 - * $\text{STACK-EMPTY}(S)$
 - * $\text{STACK-FULL}(S)$

• STACK-EMPTY (S)

```
if S.top = 0
|
| return TRUE;
else return FALSE;
end if
```

STACK-FULL (S)

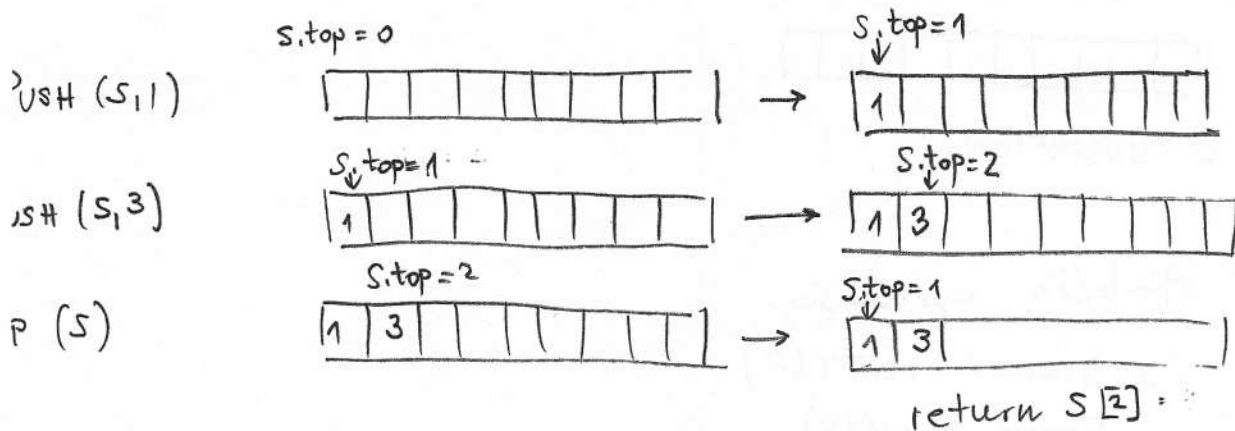
```
if S.top = n
|
| return TRUE;
else return FALSE;
end if
```

PUSH (S, x)

```
if STACK-FULL (S)
|
| error: "Overflow"
else
|
| S.top ← S.top + 1;
|
| S[S.top] = x;
end if
```

POP (S)

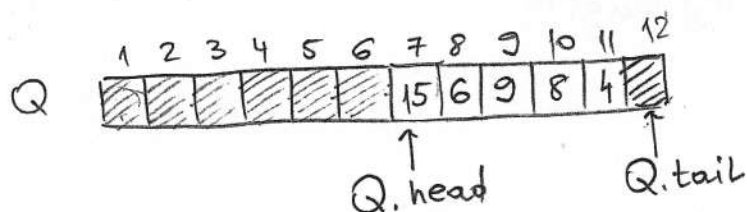
```
if STACK-EMPTY (S)
|
| error: "Underflow"
else
|
| S.top ← S.top - 1
|
| return S[S.top + 1]
end if
```



Redovi

- insert operacija - ENQUEUE
- delete operacija - DEQUEUE
- razlikujemo dvije implementacije redova:
 - * pomoću nizova
 - * pomoću povezanih listi

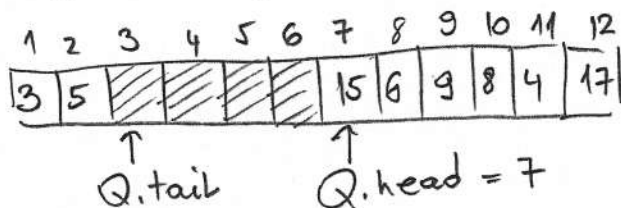
Implementacija reda pomoću polja



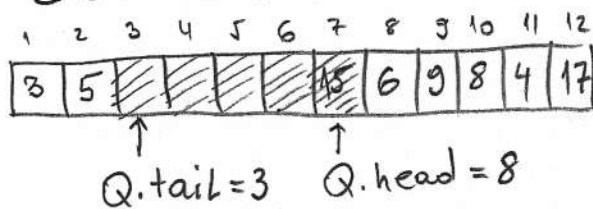
ENQUEUE(Q, 17)

ENQUEUE(Q, 3)

ENQUEUE(Q, 5)



DEQUEUE(Q)



- * polje Q dužine n može spremiti najviše n elemenata
- * Q.head = indeks početka reda
- * Q.tail = indeks kraja reda
- * Q.length = dužina polja Q
- implementacija sljedećih operacija:
 - * QUEUE-EMPTY(Q)
 - * QUEUE-FULL(Q)

QUEUE-EMPTY (Q)

```
if Q.tail = Q.head:  
    return TRUE  
else  
    return FALSE  
end if
```

QUEUE-FULL (Q)

```
if Q.head = (Q.tail + 1) mod Q.length,  
    return TRUE  
else  
    return FALSE  
end if
```

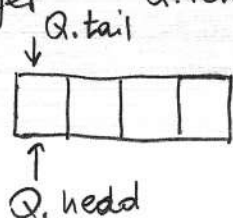
ENQUEUE (Q, x)

```
if QUEUE-FULL(Q)  
    error: "Overflow";  
else  
    Q[Q.tail] ← x  
    Q.tail ← (Q.tail + 1) mod Q.length
```

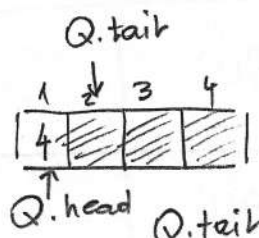
DEQUEUE (Q)

```
if QUEUE-EMPTY(Q)  
    error: "Underflow";  
else  
    x ← Q[Q.head]  
    Q.head ← (Q.head + 1) mod Q.length
```

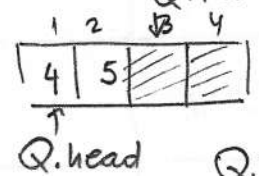
Prinjer Q.length = 4 Q.head = 1 Q.tail = 1 - P



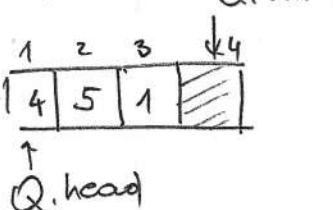
ENQUEUE (Q, 4)



ENQUEUE (Q, 5)



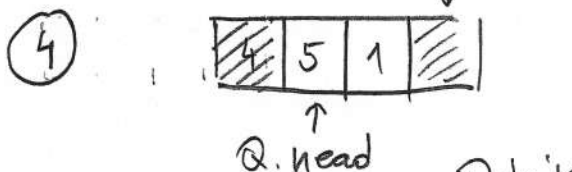
ENQUEUE (Q, 1)



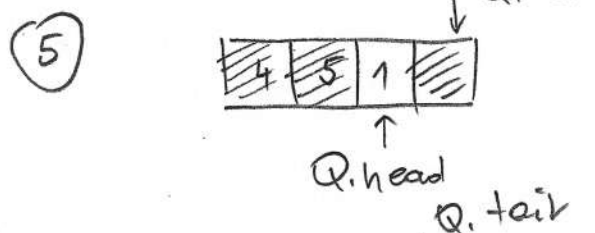
ENQUEUE (Q, 7)

error: "Overflow"

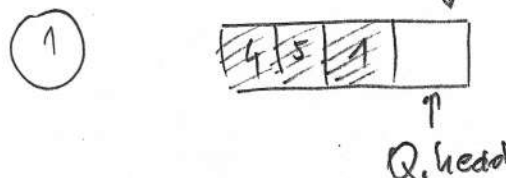
DEQUEUE (Q)



DEQUEUE (Q)



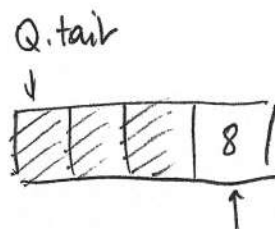
DEQUEUE (Q)



DEQUEUE (Q)

error: "Under-flow"

ENQUEUE (Q, 8)



PREGLED SLOŽENOSTI OPERACIJA NA DO SADAŠNJIH STRUKTURAMA PODATAKA

	SEARCH	INSERT	DELETE
OSTRUKO- VEZANA -ISTA	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$
ITOG	-	$\Theta(1)$	$\Theta(1)$
ED	-	$\Theta(1)$	$\Theta(1)$