Strukture podataka - tipovi podataka

dr Davorka R. Jandrlić dr Goran Lazović

Mašinski fakultet, Univerziteta u Beogradu

Apstraktni tip podataka Stek

Stek je specijalna vrsta liste u kojoj se sva umetanja i brisanja obavljaju na jednom kraju koji se zove vrh steka, po tzv. LIFO principu (Last In First Out).

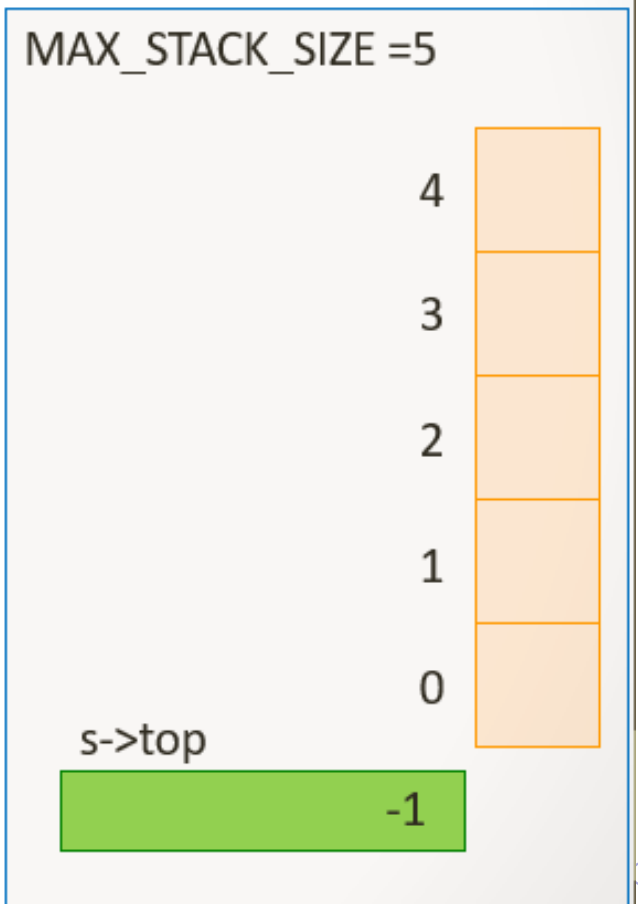
Operacije nad stekom:

- MAKENULL(S)
- TOP(S) vraća element sa vrha steka. Ekvivalento je sa RETRIEVE(FIRST(S), S)
- POP(S) briše element sa vrha steka. Često implementacija operacije POP kao rezultat vraća uklonjen element. Ekvivalentno je sa DELETE(FIRST(S), S).
- PUSH(x, S) umeće element x na vrh steka S. Ekvivalentno je sa INSERT(x, FIRST(S), S).
- EMPTY(S) proverava da li je stek S prazan

Implementacija Steka pomoću niza

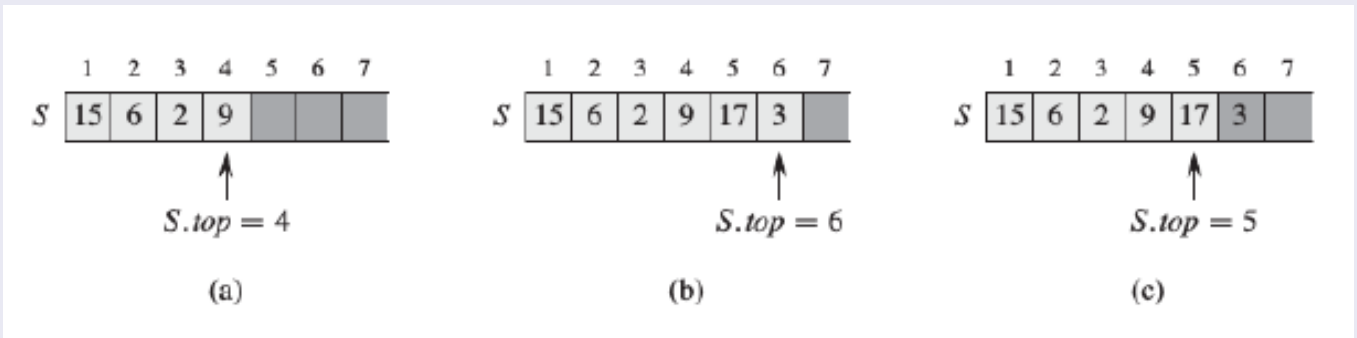
- inicijalizacija praznog steka:
  - postavljanje vrha na početnu vrednost

```
#define MAX_STACK_SIZE 5

#define TOP_POSITION 0

typedef struct _stack {
    position top;
    element_type
      elements[MAX_STACK_SIZE];
} stack;

void initStack(stack *s){
    s->top = TOP_POSITION - 1;
}
```



Push i Pop

```
void push (element_type x, stack *s) {
    assert(s->top < MAX_STACK_SIZE - 1);
    s->top++;
    s->elements[s->top] = x;
}

void pop (stack *s) {
   assert(!empty(s));
   s->top--;
}
```



Implementacija steka preko niza je pogodna jer se sve operacije izvršavaju za konstantno vreme $O(1)$. Mana ove implementacija je što se stek može prepuniti.

Implementacija steka preko pokazivača

Ova se implementacija zasniva na povezanoj listi.

- Struktura se pojadnostavljuje jer nije potrebno pamtiti poziciju.
- Vrh steka je na početku povezane liste.

Kako su operacije ubacivanja i dohvatanja elementa sa steka jednostavnije (uvek se uzimaju sa vrha steka), složenost ovih operacija je $O(1)$.

Red (queue)

Slično kao kod steka, i kod reda je predefenisan redosled pristupa elementima. S time da se u redu poštuje pravilo FIFO (first in first out). Element koji se briše je uvek onaj koji je u skupu najduže. Red se takođe može jednostavno implementirati preko niza.

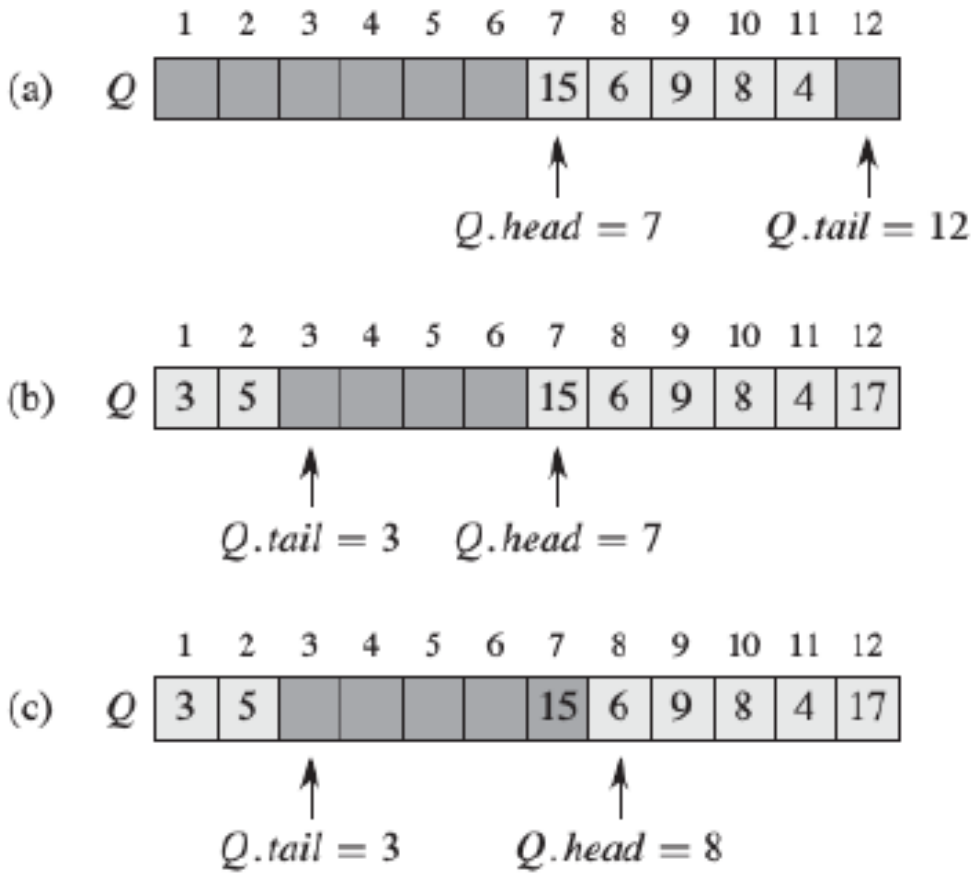
Operacije nad redom

- **Enqueue** - operacija umetanja reda.
- **Dequeue** - operacija izbacivanja iz reda.

Kada se element umeće u red, on zauzima svoje mesto na kraju reda. Element koji se izbaciju iz reda je uvek onaj sa početka reda.

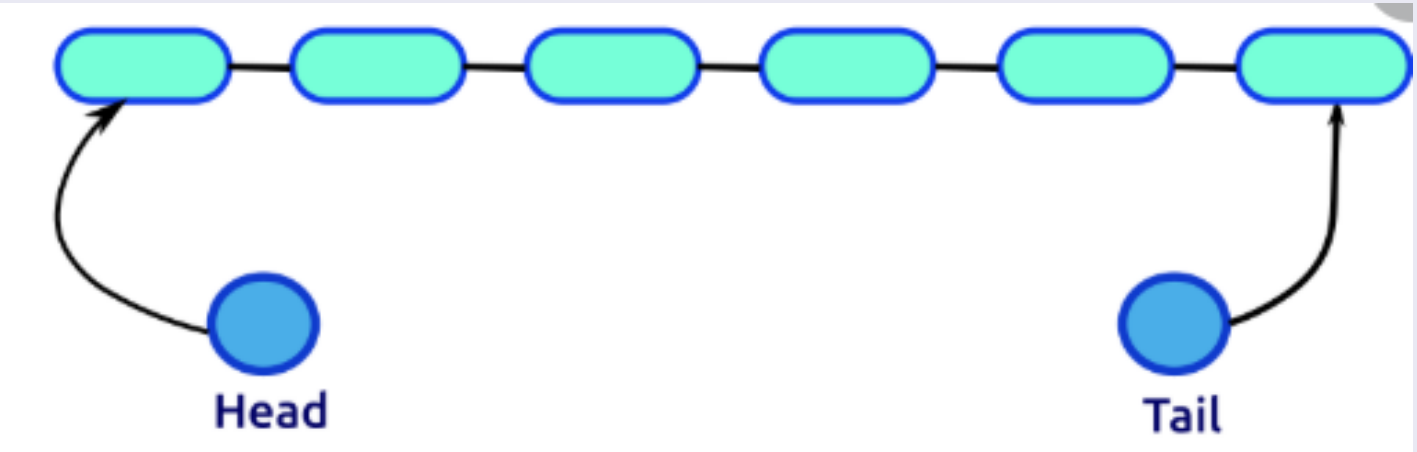
Red sadrži početak (head) i kraj (tail).

Implementacija reda pomoću niza



```c
typedef struct {
   int elements[QUEUE_SIZE];
   int head;
   int tail;
   int size;
} Queue;

void make_empty(Queue *q);
int is_empty(Queue *q);
int is_full(Queue *q);
void enqueue(Queue *q, int el);
int dequeue(Queue *q);
```

Implementacija reda preko povezane liste




```
typedef int element_type;

typedef struct _node {
   element_type element;
   struct _node *next;
} node;

typedef struct {
   node *head;
   node *tail;
```