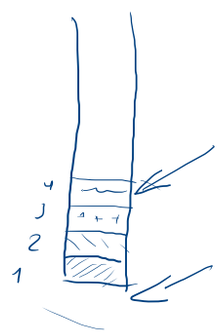
Элементарные структуры данных

Динамическое множество - поддерживает операции добавления и удаления элементов

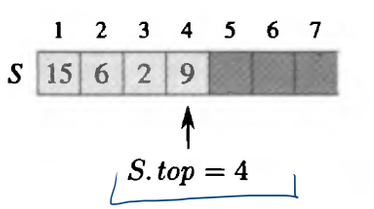
Стек

LIFO - last in, first out: последний добавленный при удалении будет первым



Реализация: массив S

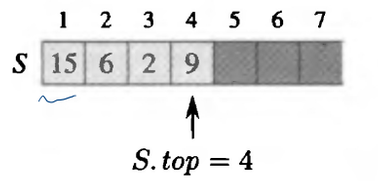
и индекс последнего добавленного элемента S.top



Операции:

проверка на пустоту стека

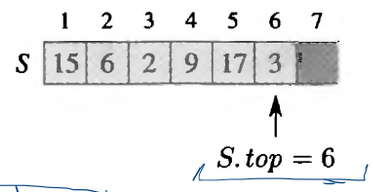
```
STACK-EMPTY(S)
1  if S.top == 0
2       return TRUE
3  else return FALSE
```

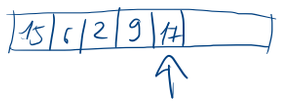


O(1)

добавление элемента

```
PUSH(S, x)
1  S.top = S.top + 1
2  S[S.top] = x
```

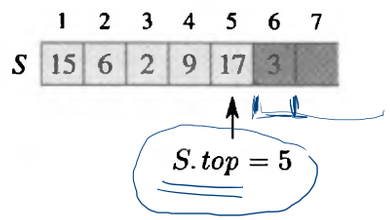




O(1)

снять элемент со стека

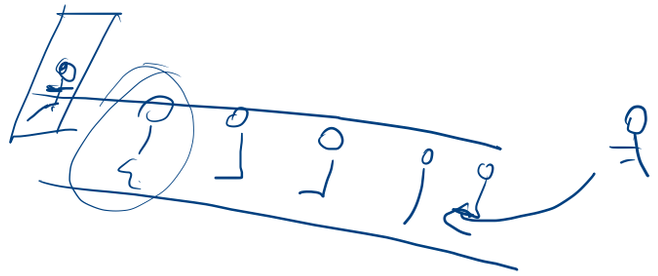
```
POP(S)
1  if STACK-EMPTY(S)
2       error "underflow"
3  else S.top = S.top − 1
4       return S[S.top + 1]
```



O(1)

Очередь

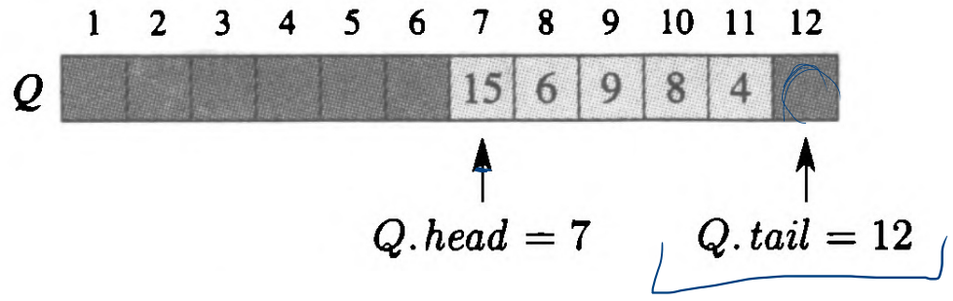
FIFO - first in, first out: первый добавленный при удалении будет первым



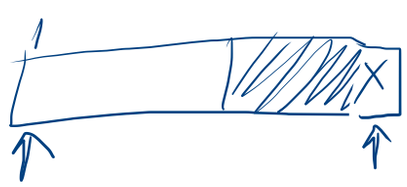
Реализация: массив

и индекс первого элемента очереди

индекс хвоста очереди



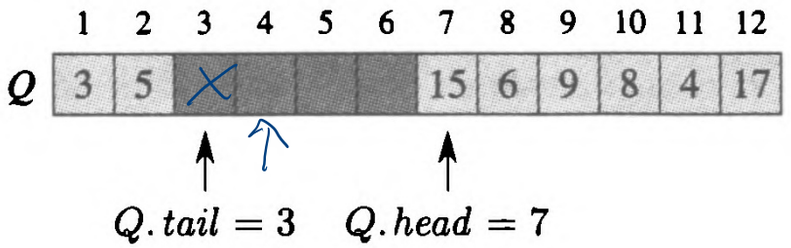
Операции:



добавление элемента в очередь

```
ENQUEUE(Q, x)
1  Q[Q.tail] = x
2  if Q.tail == Q.length
3      Q.tail = 1
4  else Q.tail = Q.tail + 1
```

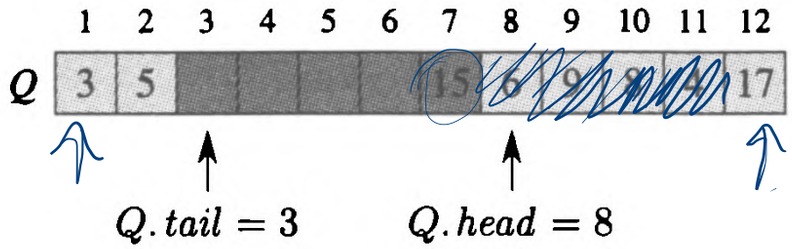
O(1)



извлечь элемент из очереди

```
DEQUEUE(Q)
1  x = Q[Q.head]
2  if Q.head == Q.length
3      Q.head = 1
4  else Q.head = Q.head + 1
5  return x
```

O(1)



Связный список

Динамическое множество, элементы которого расположены в линейном порядке, определенном указателями

Поля: key - ключевое значение

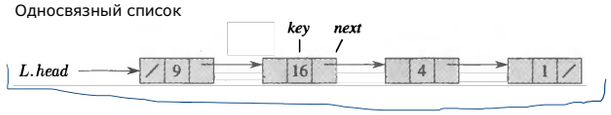
next - указатель на следующий элемент

prev - указатель на предыдущий элемент

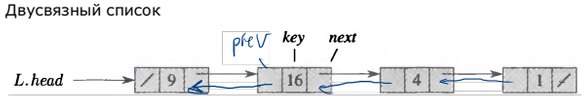
head - указатель на первый элемент списка

Типы списков:

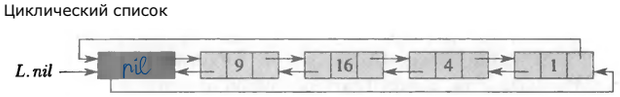
Односвязный список

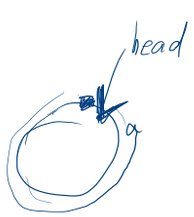


Двусвязный список



Циклический список





Операции в двусвязном списке:

поиск элемента с заданным ключом

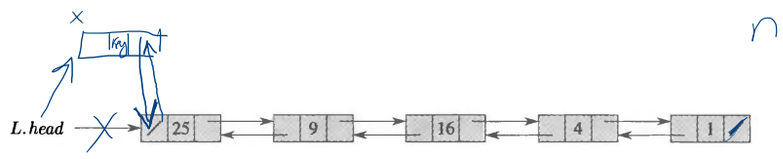
```
LIST-SEARCH(L, k)
1  x = L.head
2  while x ≠ NIL и x.key ≠ k
3      x = x.next
4  return x
```

O(n)

вставка элемента с определенным ключом

```
LIST-INSERT(L, x)
1  x.next = L.head
2  if L.head ≠ NIL
3      L.head.prev = x
4  L.head = x
5  x.prev = NIL
```

O(1)



n

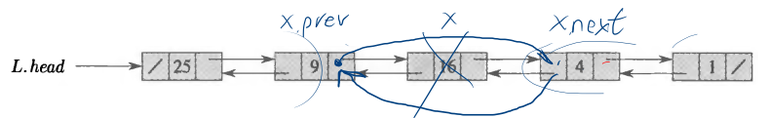
удаление элемента с определенным ключом

```
LIST-DELETE(L, x)
1  if x.prev ≠ NIL
2      x.prev.next = x.next
3  else L.head = x.next
4  if x.next ≠ NIL
5      x.next.prev = x.prev
```

+ поиск

O(n)

O(1)



Двоичный поиск в отсортированном массиве

упорядоченный

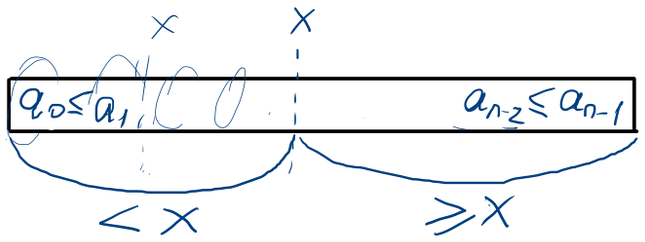
Задача: дан массив из n элементов и ключ x.
Найти: элемент с ключом x или сказать, что такого нет

Простой алгоритм: линейный поиск

```
for i=0 to n-1
    if (a[i]==x)
        return i
return -1
```

O(n).

Бинарный поиск

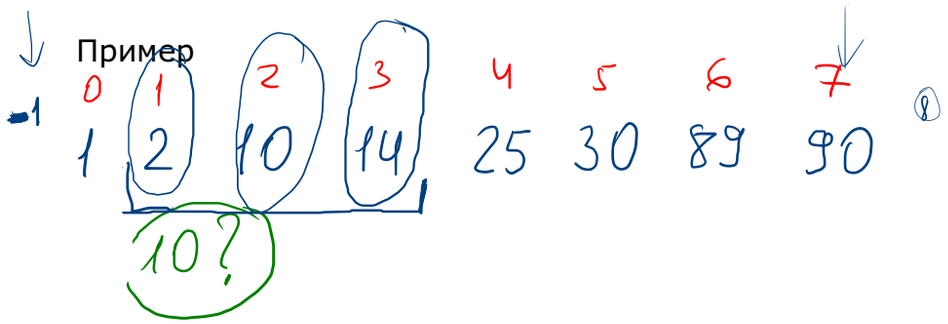


Инвариант: индексы l и r, т.ч. a[l]<x и a[r] >=x

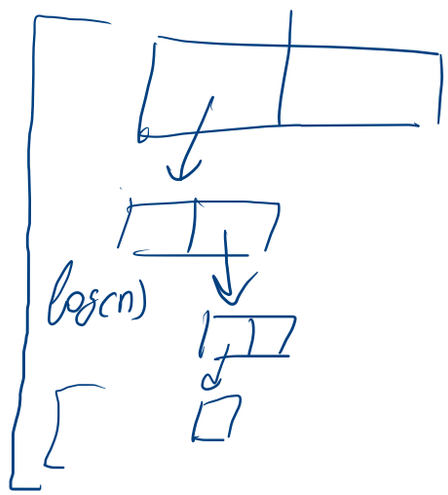
```
BinarySearch (A,x)
l=-1
r=n
while (r>l+1)
    m=⌊(l+r)/2⌋
    if (a[m]<x)
        l=m
    else
        r=m
if (r<n && a[r]==x)
    return r
else
    return -1
```

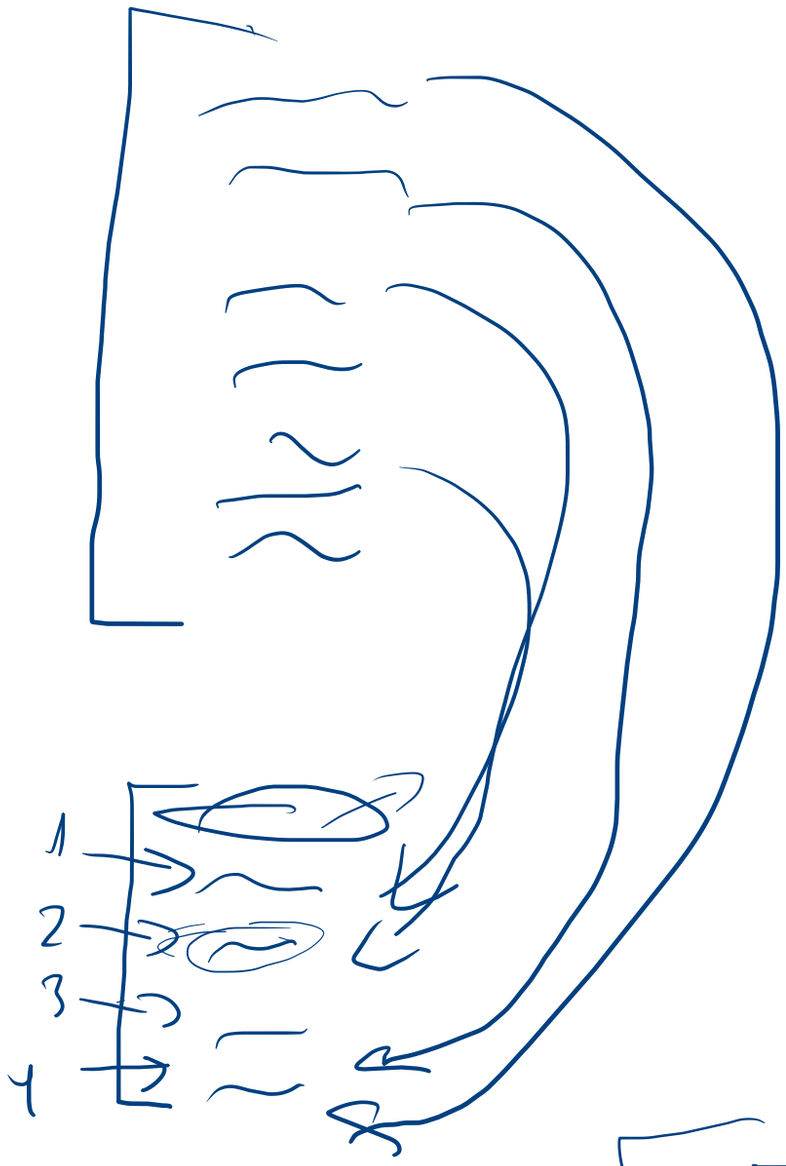
O(logn)

Пример

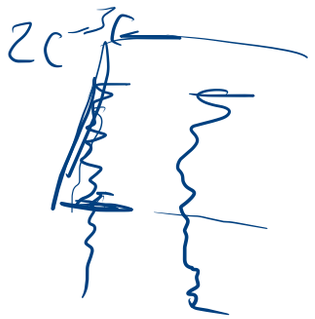


l	r	m
-1	8	3
-1	3	1
1	3	2
1	2	




1
2
3
4


10 cm


20