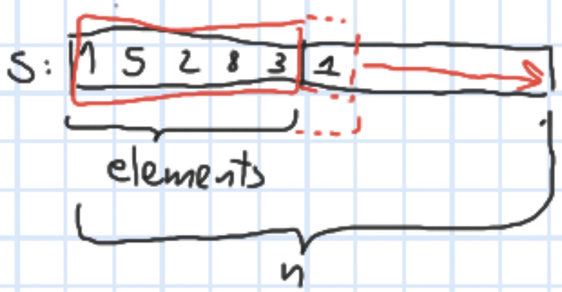
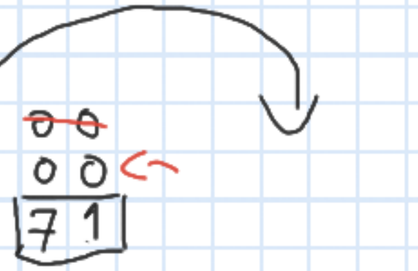
ASD - Wykład 5

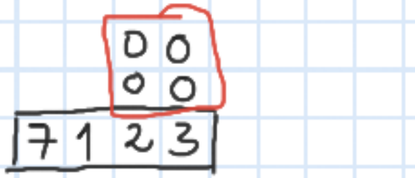
Implementacja tablicowa stosu

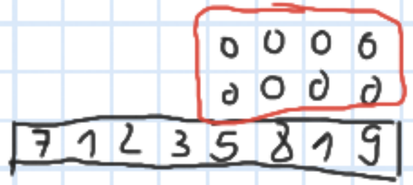


Co zrobić gdy stos się przepełni?

- zaalokować nowy, 2 razy większy i skopiować tam zawartość starego stosu







Analiza pesymistyczna:

- pop $O(1)$
- push $O(n)$

Analiza zamortyzowana:

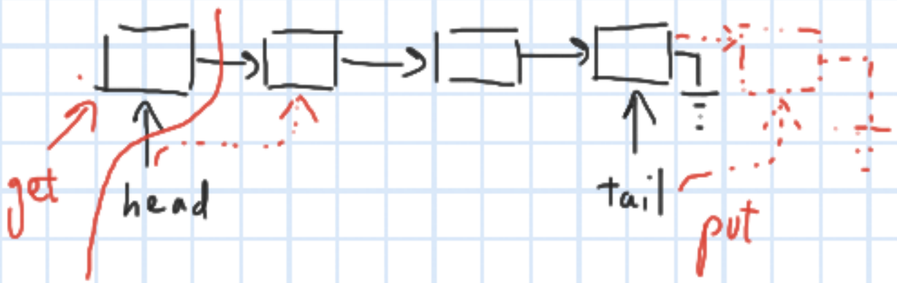
- push 3 zł
- pop 1 zł.

Kolejka (queue)

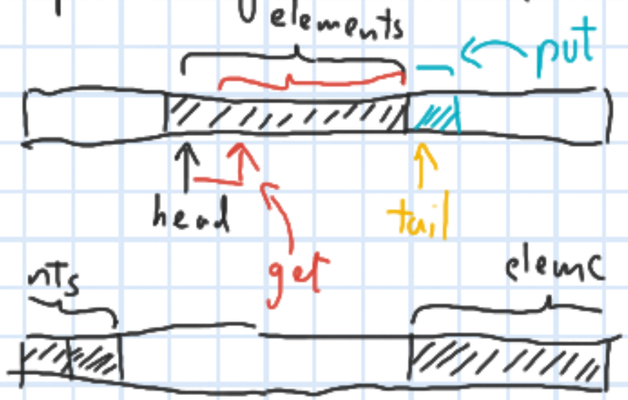
Operacje
- put
- get

"Coś co pozwala ustawiać na koniec i pobierać z początku"

Implementacja listowa



Implementacja tablicowa



Klasyczne zadanie

W jaki sposób zaimplementować kolejkę, jeśli do dyspozycji mamy dwa stosy

Chcemy zamortyzowanej złożoności O(1) dla put i get

Kolejka priorytetowa

"coś co pozwala na układanie danych powiązanych z priorytetem i wyciąganie w kolejności malejących/rosnących priorytetów"

- kopiec binarny
- posortowana tablica
- nieposortowana tablica

} lub lista

Metody konstrukcji algorytmów

- dziel i zwyciężaj ← Quick Sort, Merge Sort
- algorytmy zachłanne
- programowanie dynamiczne

  metoda zamiany wykładniczego algorytmu rekurencyjnego na wielomianowy algorytm iteracyjny przez spamiętywanie wyników

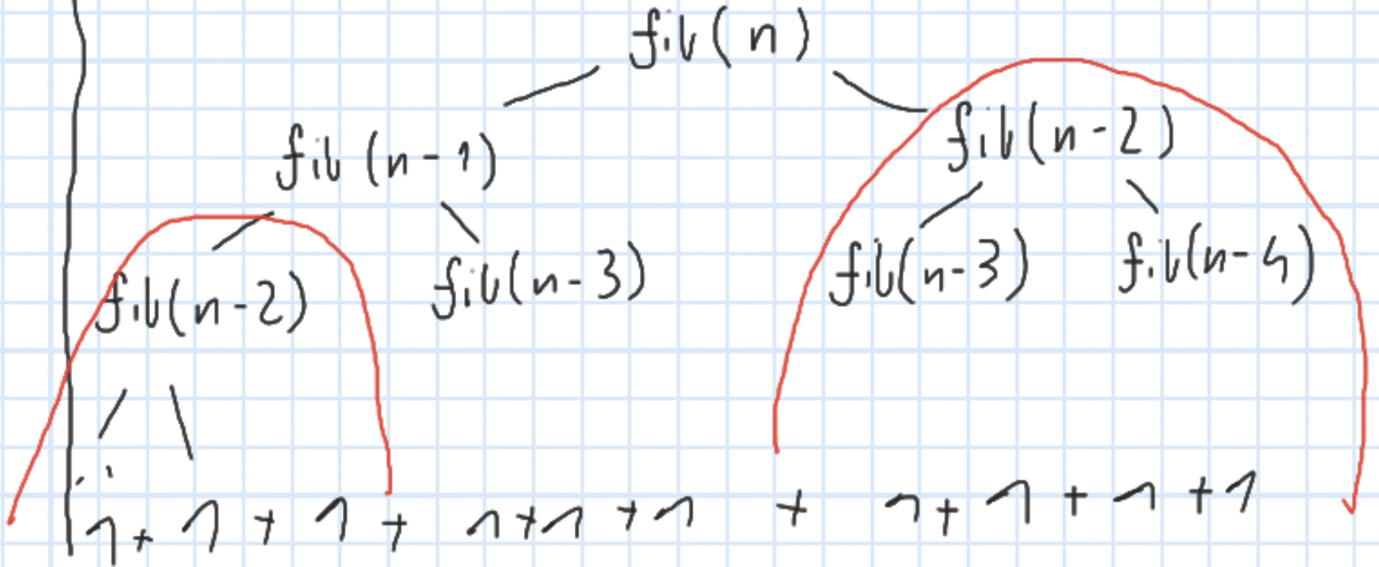
Przykład elementarny

$F_1 = 1$

$F_2 = 1$

$F_n = F_{n-1} + F_{n-2}$

```
def fib(n):
    if n ≤ 2: return 1
    return fib(n-1) + fib(n-2)
```



Implementacja dynamiczna

```
def fib_dyn( n ):
    F = [1] * ( n + 2 )
    for i in range(2, n+1):
        F[i] = F[i-1] + F[i-2]
    return F[n]
```

```
def fib_dyn2( n ):
    if n ≤ 1 : return 1
    Fi1 = 1
    Fi2 = 1
    for i in range(2, n+1):
        Fi = Fi1 + Fi2
        Fi2 = Fi1
        Fi1 = Fi
    return Fi
```

A co by tu zrobić lepiej?