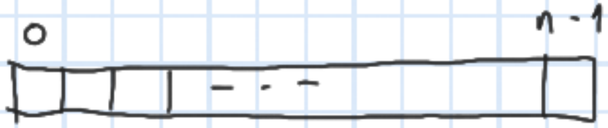
Algorytmy i Struktury Danych

Wykład 5

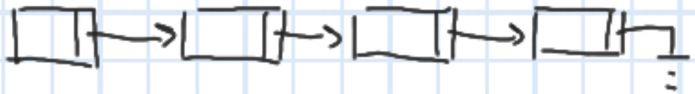
Abstrakcyjne struktury danych
- "kontrakt" co do działania
- zestaw operacji
- fizyczna realizacja

Tablica



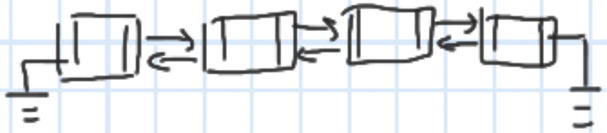
"Coś co ma ponumerowane komórki, do których można się bezpośrednio odwoływać"

Lista jednokierunkowa



"Coś co pozwala wędrować od początku do końca i wpinać/usuwać elementy"

Lista dwukierunkowa



"Coś co pozwala wędrować w obie strony i wpinać/usuwać elementy"

```
class DNode:
    def __init__(self):
        self.prev = None
        self.next = None
        self.val = None
```

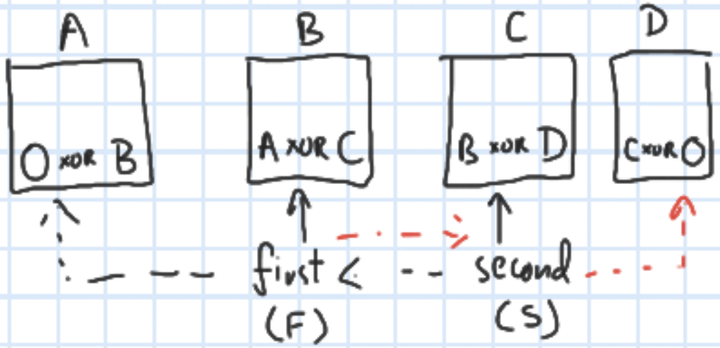
Lista dwukierunkowa z jednym wskaźnikiem

```
  11010101
XOR 01110011
------------
  10100110
```

(A XOR B) XOR B = A

(A XOR B) = (B XOR A)

(A XOR B) XOR A = B



(A XOR C) XOR S.ptr =

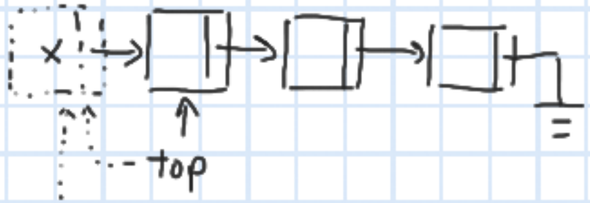
(A XOR C) XOR C = A

Stos (ang. stack)

"Coś, co pozwala odkładać elementy na szczyt, lub z niego zdejmować"

- push(x) <— włóż x na stos
- pop() <— zdejmij element ze stosu
- isempty() <— czy stos jest pusty

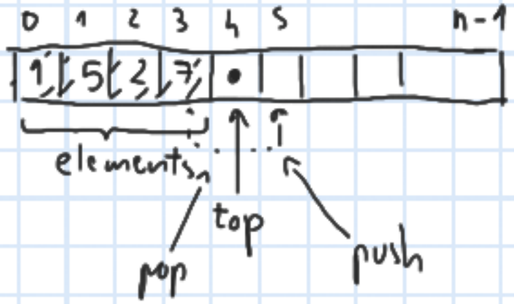
Implementacja na liście jednokierunkowej



push(x)

pop()

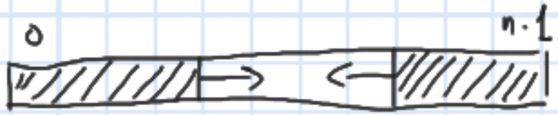
Implementacja tablicowa



Co jeśli skończy nam się miejsce w zaalokowanej tablicy?

- wówczas alokujemy nową, 2× większą tablicę i kopiujemy do niej zawartość stosu

Dwa stosy

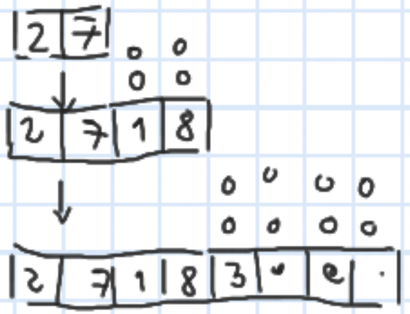


uczone komputery PC XT/AT

Analiza kosztu zamortyzowanego

push - 3 zł

pop - 1 zł

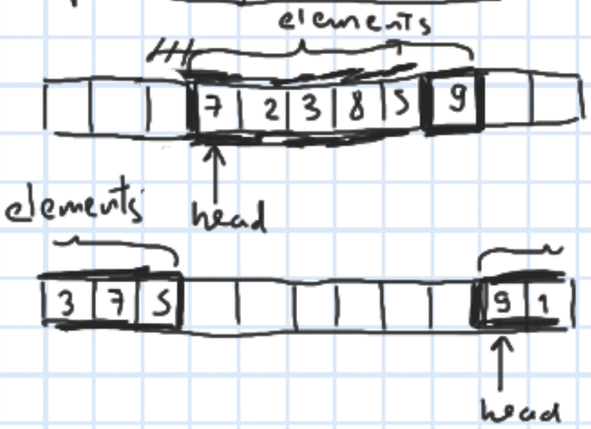


Kolejka (ang. queue)

"coś, co pozwala ustawiać elementy na koniec i wyciągać z początku"

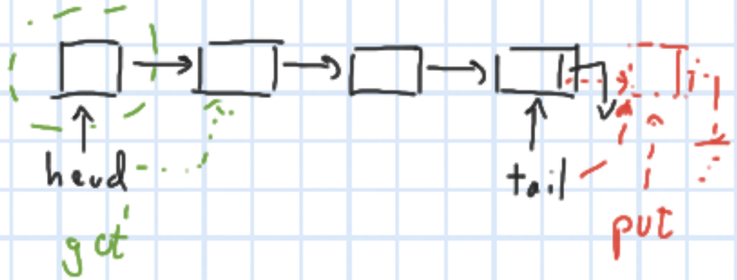
put / get enqueue / dequeue

Implementacja tablicowa



Przepełnienie: ta sama metoda co przy stosie

Implementacja listowa



Standardowa zagadka

Jak zaimplementować kolejkę mając do dyspozycji tylko dwa stosy?

Kolejka priorytetowa (ang. priority queue)

"coś do niego układamy elementy porządane z priorytetem, a wyciągamy w kolejności malejących priorytetów"

- insert
- extract max
- decrease key

Tablica asocjacyjna / słownik

"Coś co można indeksować dowolnymi wartościami"

A["słoń"] = 7

A["mysz"]

A[(1, 19)]

Implementacje

Implementacje	insert	extract
- kopiec binarny	O(log n)	O(log n)
- posortowana tablica	O(n)	O(1)
- nieposortowana tablica	O(1)	O(n)
- j.w. dla listy		
- wiele innych		

Implementacja

- drzewa BST
  - AVL
  - RB / czerwono-czarne
  - B-drzewa
  - drzewa splay

  } O(log n)

- skip lista
- tablica haszująca O(1)

Algorytmy Grafowe

$G = (V, E)$

$V = \{v_1, \ldots, v_n\}$

$E = \{e_1, \ldots, e_m\}$

e_i – złożone dwuelementowy zbiór wierzchołków