

Массивы

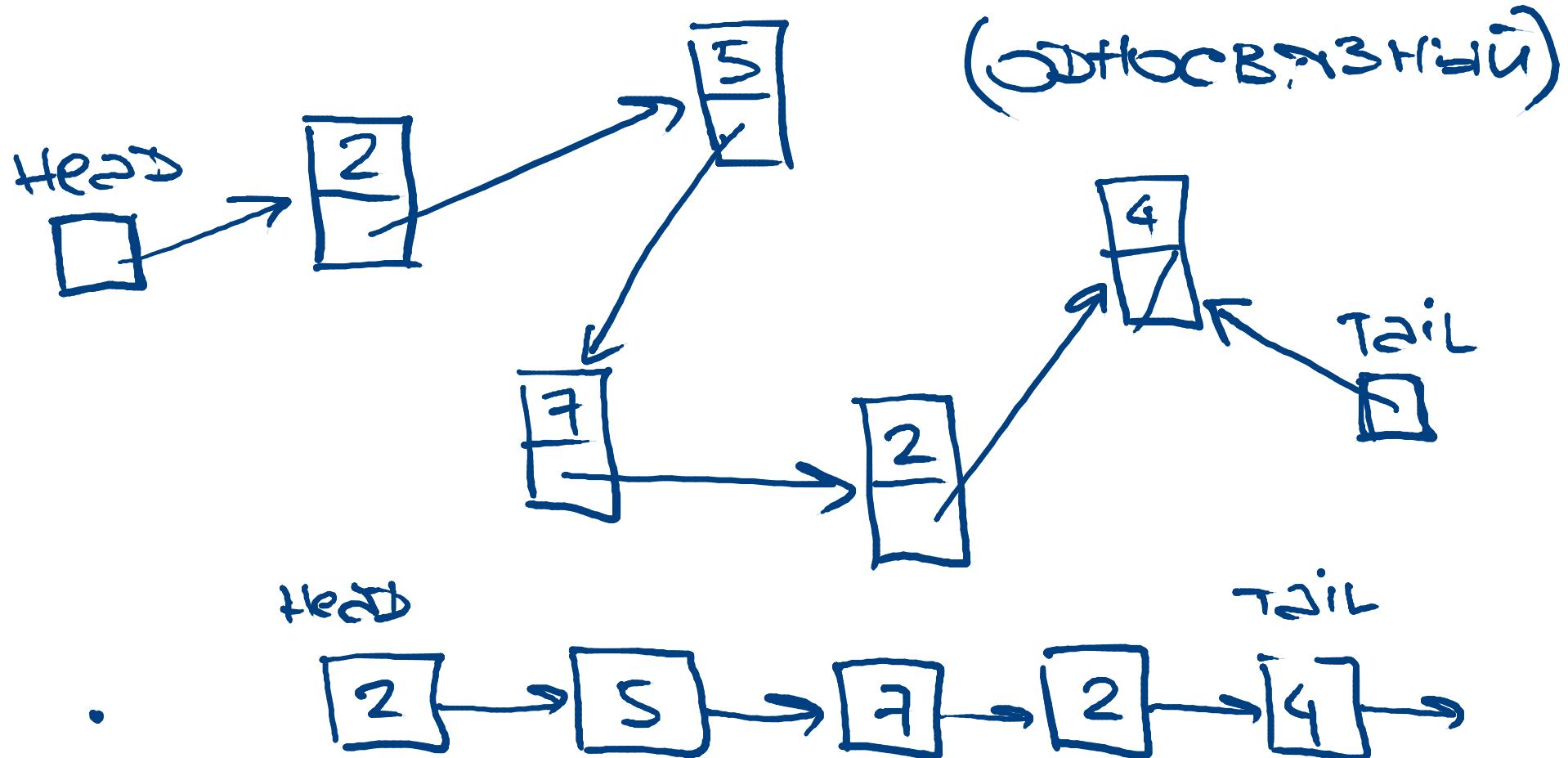
7	8	1	5	4	7				
0	1	2	3	4	5	6	7	8	9

- Константный доступ по индексу
- непрерывный кусок памяти
- фиксированый размер

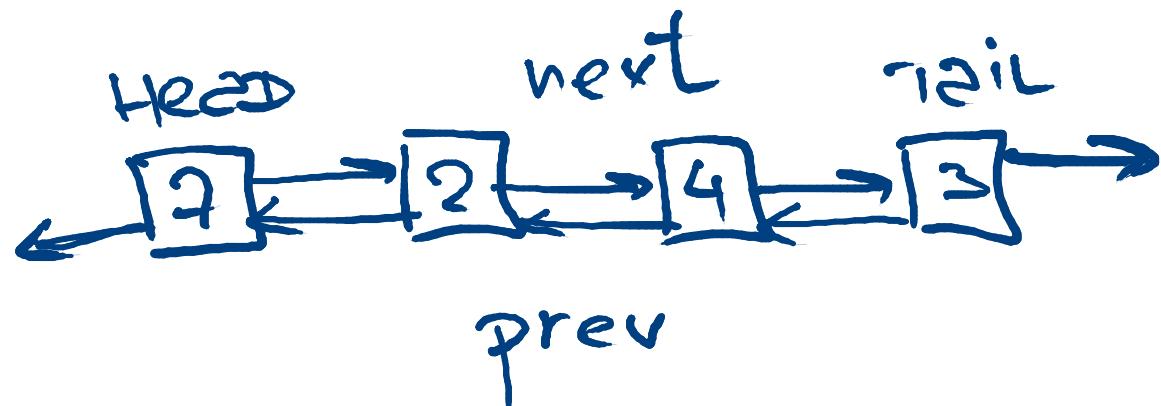
	Быстрая	Ранжирование
таблица	$O(n)$	$O(n)$
котейн	$O(1)$	$O(1)$
средства	$O(n)$	$O(n)$



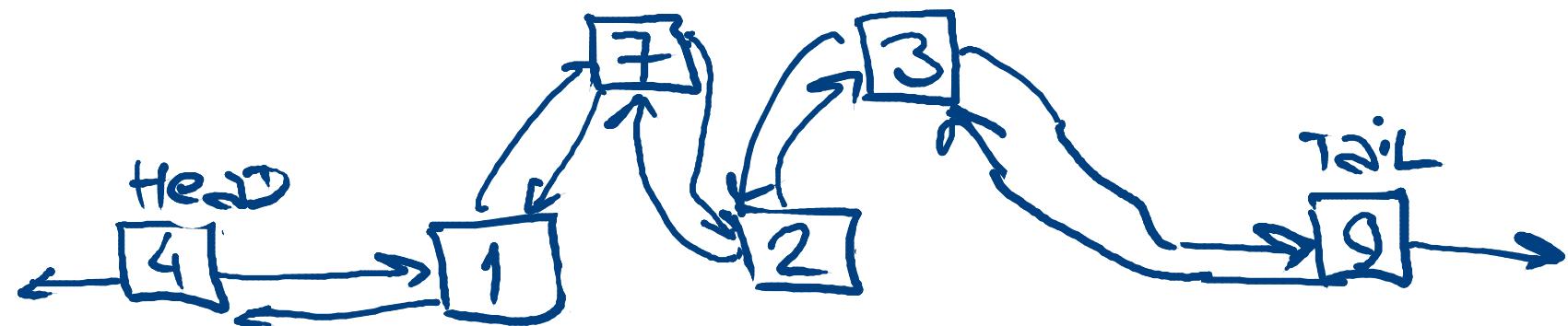
Спинки



ДВУСЫЛІЙСТІНІН СМУСОК:



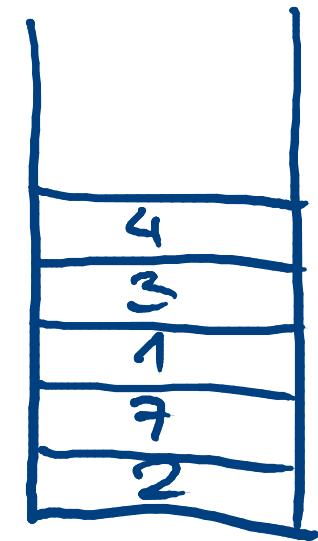
	Вставка	Удаление
Начало	$O(1)$	$O(1)$
Конец	$O(1)$	$O(1)$ (извлечь)
середине	$O(1)$ (извлечь)	$O(1)$ (избрать)



Стеки

Абстрактная структура данных:

- Push(Key)
- Key Top()
- Key Pop()
- Bool Empty()



LIFO (last in
first out)

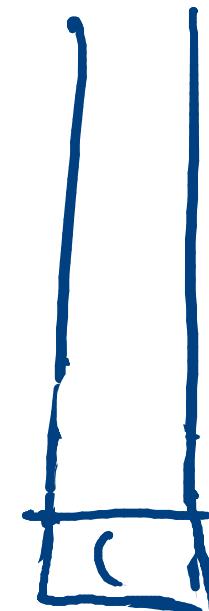
Пример: скобочная последовательность

Правильные:

- ()[]([])
- ((([]())[([)])))[]

Неправильные:

-][
- ([)]()
- ([]



IsBalanced(str):

Stack stack

for char in str:

 if char ∈ {'(', '['}:

 stack.Push(char)

 else:

 if stack.Empty(): return False

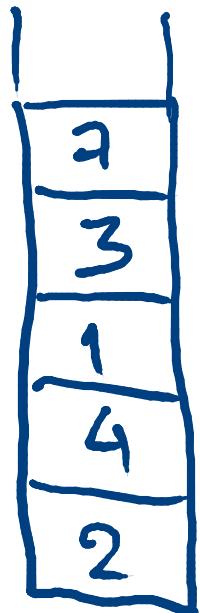
 top ← stack.Pop()

 if (top = '(' and char ≠ ')') or
(top = '[' and char ≠ ']'):

 return False

return stack.Empty()

Реализации



- с помощью массива



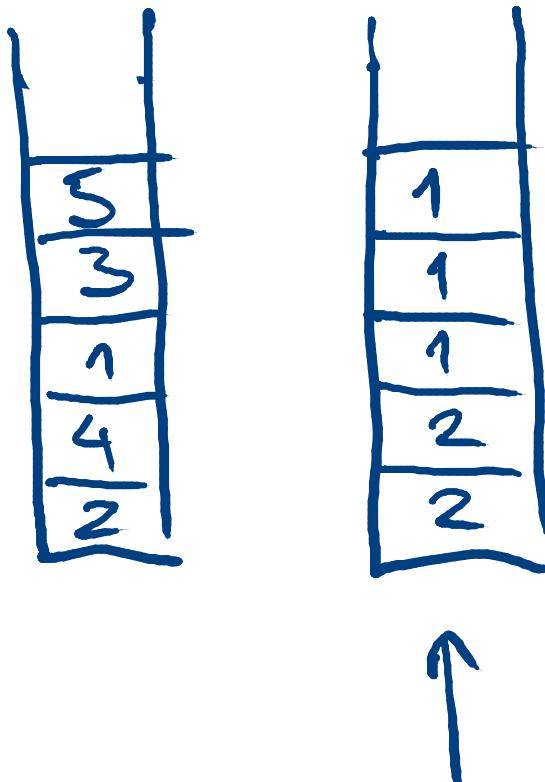
- с помощью списка

head



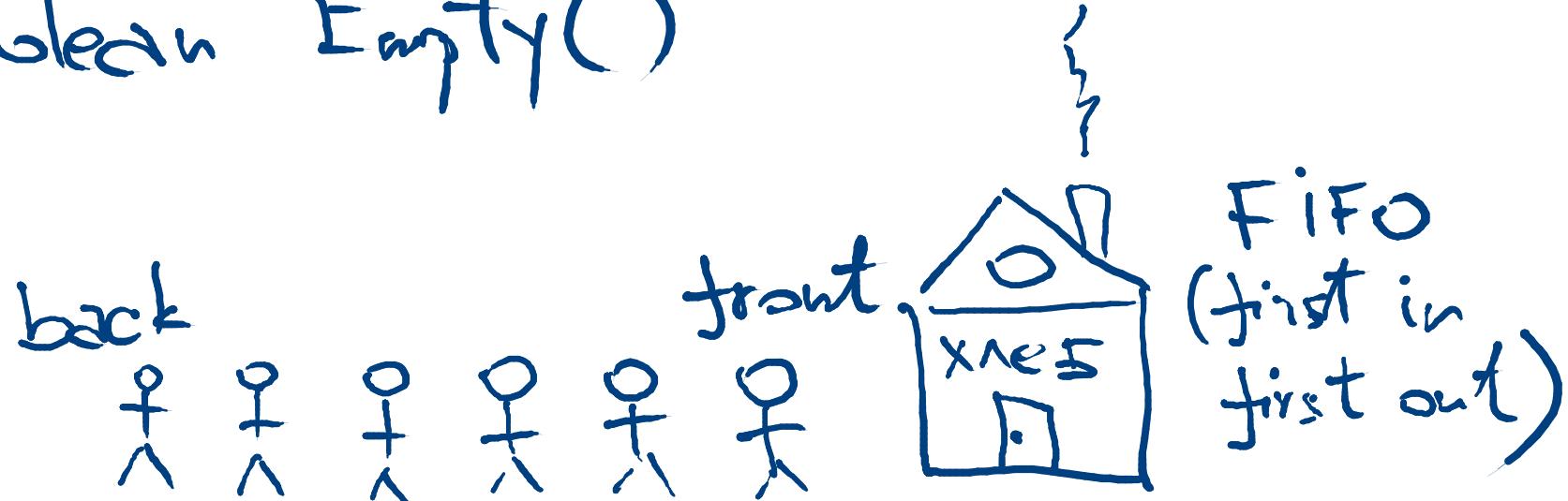
Все операции: $O(1)$

Стек с поддержкой минимума



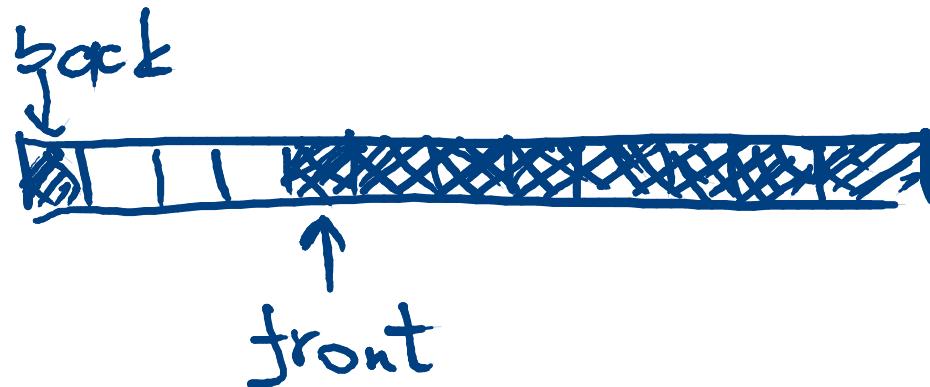
Queues

- Enqueue(Key) (PushBack)
- Key Dequeue() (PopFront)
- Boolean Empty()

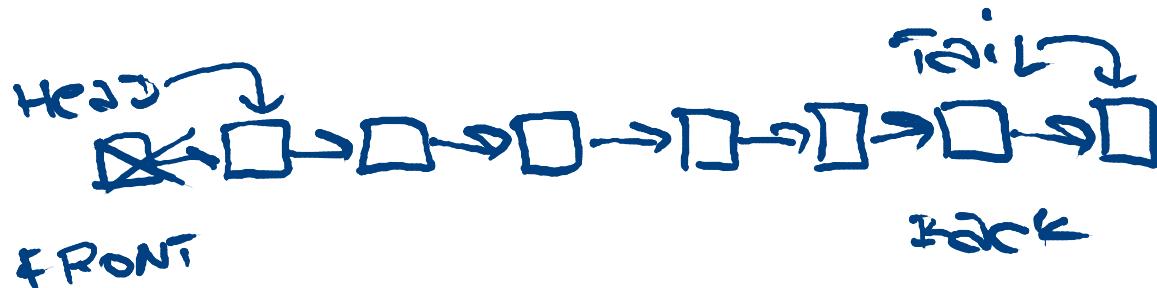


Рекурсивные:

- с помощью массива:



- с помощью структ:



Все
одномерные:

$O(1)$

- С помощью двух стеков:

к наст
надём

вытачиваем

$\text{PushBack}(i)$ где $i = 1, 2, 3, 4$:



$\text{PopFront}()$:



$\text{PushBack}(5)$:



$\text{PopFront}()$:



ОМОНИУЗ.
ВРЕМЯ

$\Theta(1)$

Задача:

Вход: последовательность чисел
 $a_1, a_2, \dots, a_n,$

число m .

Выход: пройтиесь по последнику
окном размера m и вывести
минимум в каждом из них.

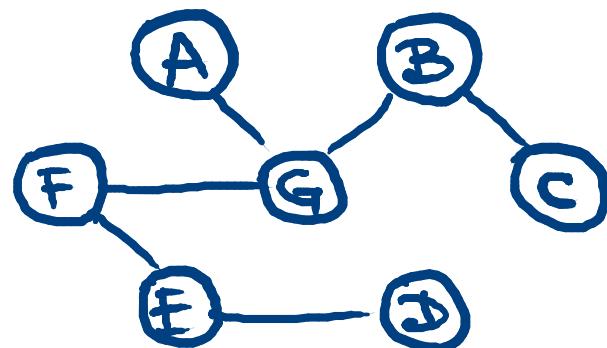
Пример:

5 1 3 2 4 6 1 7 3 2 8	$m = 3$
1 1 2 2 1 1 1 2 3	

Наивное решение: $O(n \cdot m)$. Торпрос: $O(m \cdot n)$?

Деревья

Дерево:

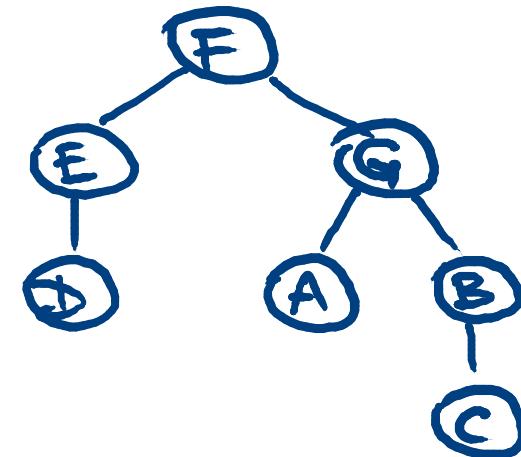


ОПР: связный
граф без
циклов

Свойства:

- 1) n вершин, $n-1$ ребер
- 2) ровно один путь
для любых двух вершин

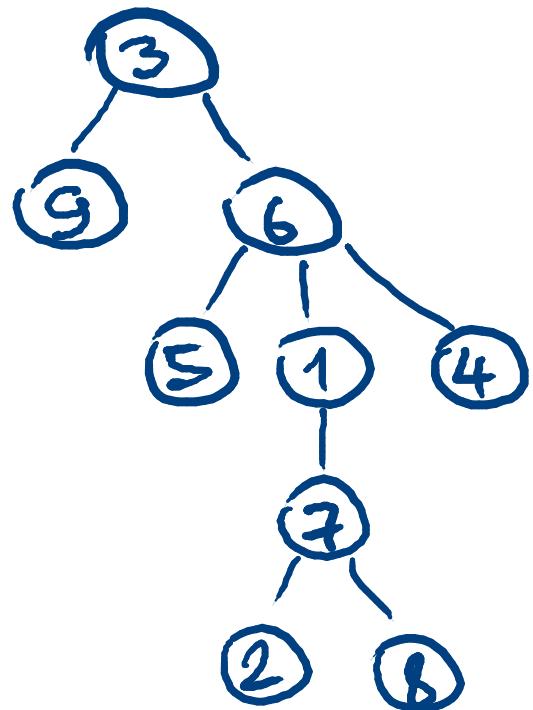
Корневое дерево



F - корень
A, B - дети G
E - родитель D
A, B, C - потомки G
F, G - предки B

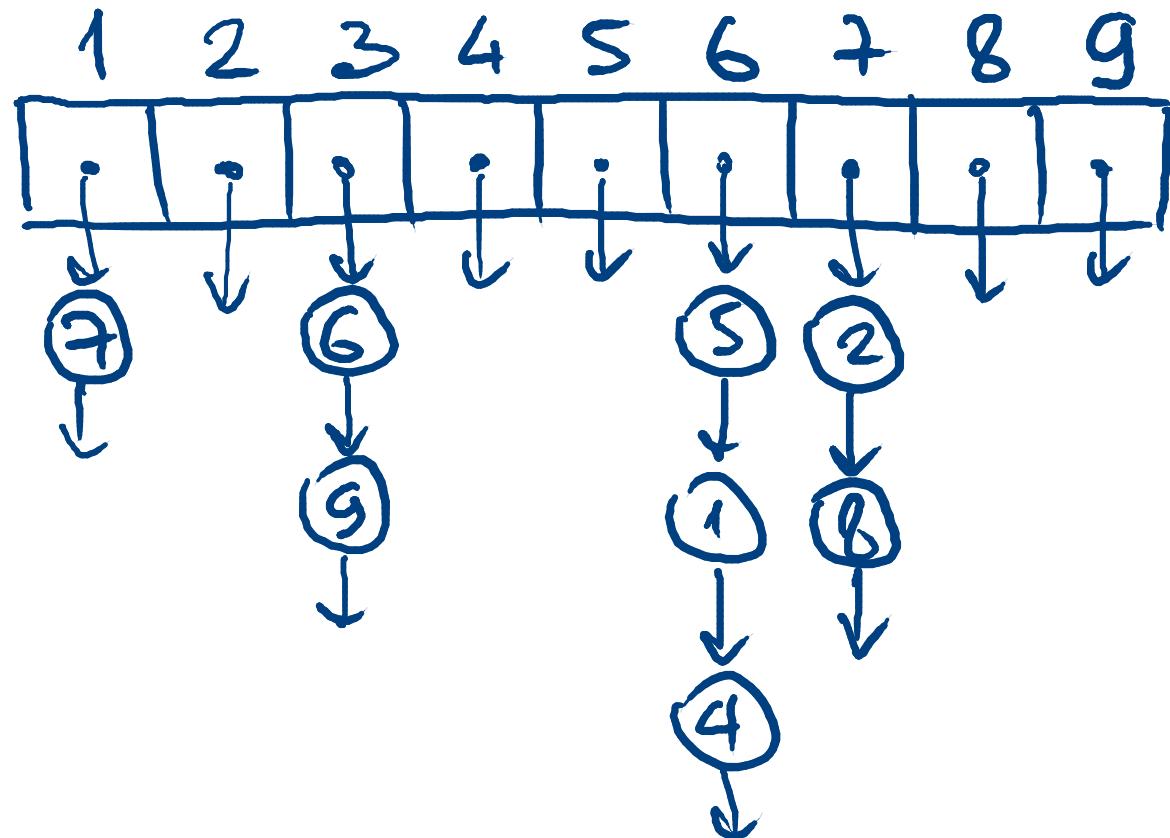
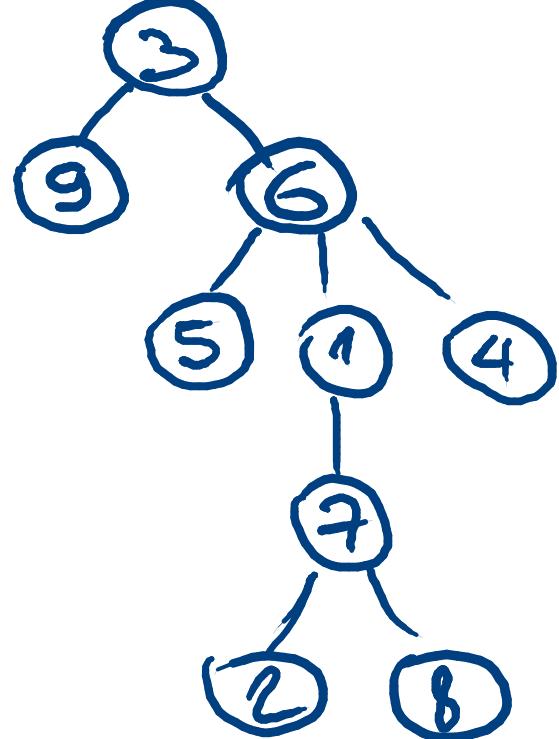
Способы представления деревьев

1. Список родителей

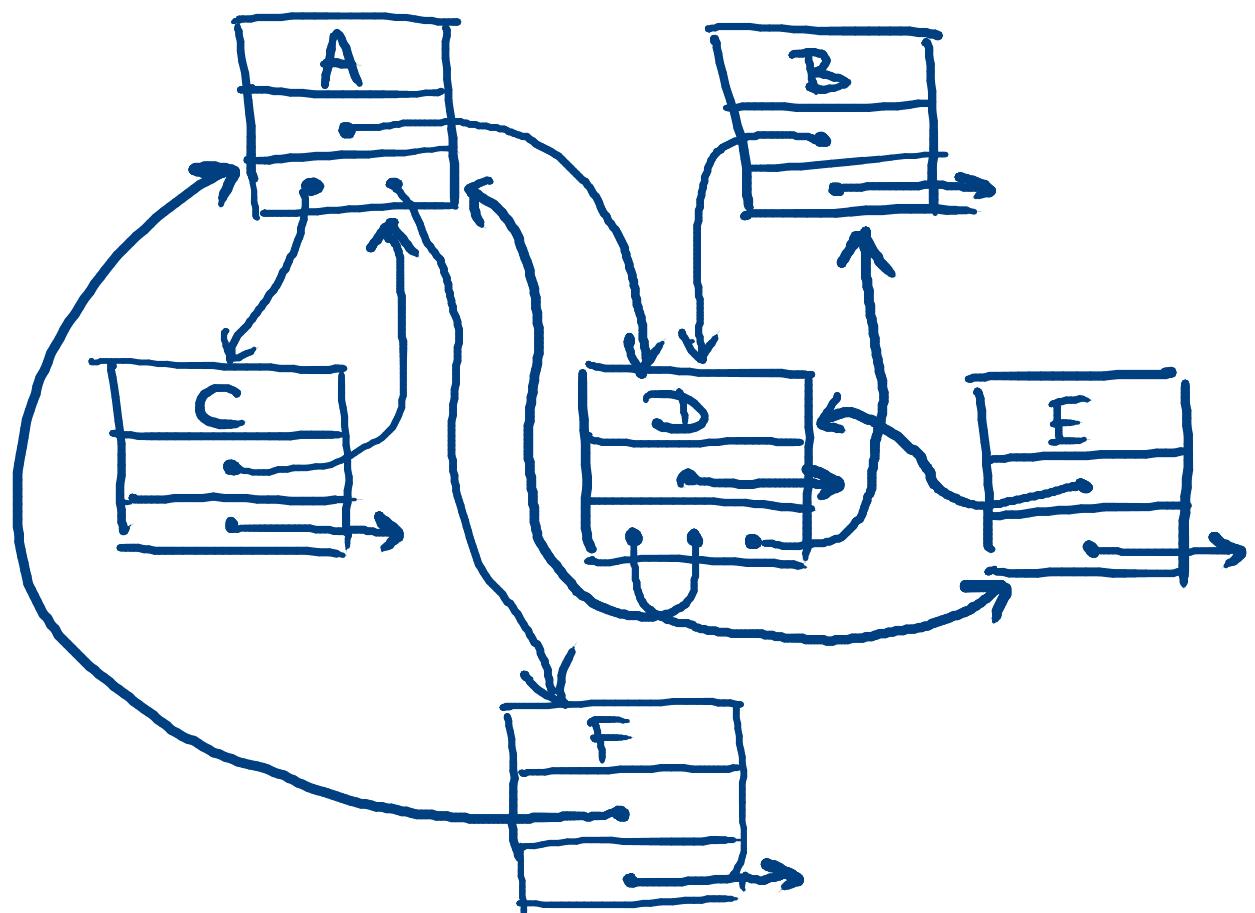
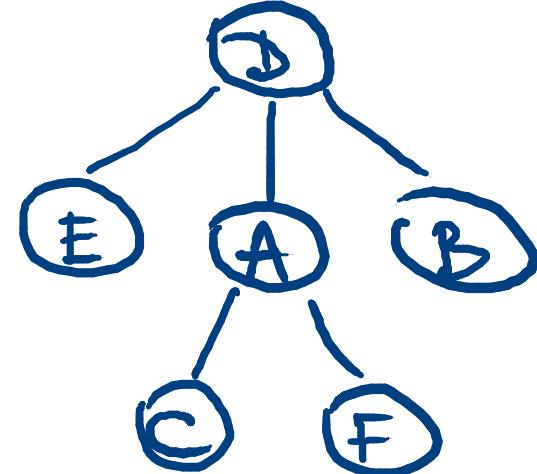


1	2	3	4	5	6	7	8	9
6	7	3	6	6	3	1	7	3

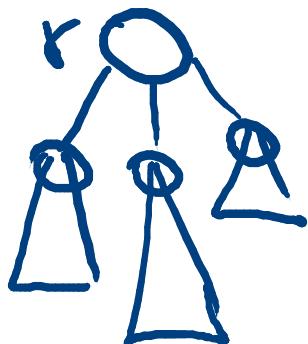
2. Способы детерминации



3. Зеркала хранят данные, ссылку
на родителя, ссылки на детей



Рекурсивное определение деревьев и рекурсивные алгоритмы



height(r):

height $\leftarrow 1$

for all children c of r:

height $\leftarrow \max(\text{height}, 1 + \text{height}(c))$

return height

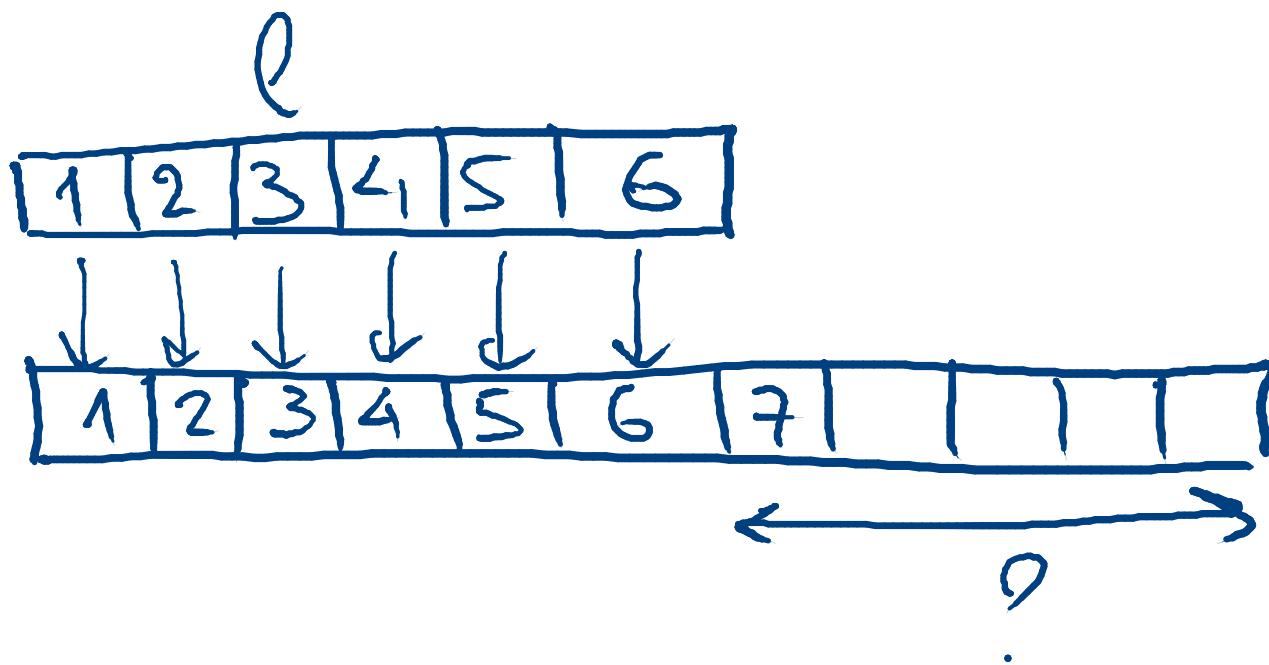
PrintTree(r):

print(r)

for all children c of r:

PrintTree(c)

MacCubby's replacement
Polymer



Схемы перевыделения:

$$\frac{\text{Деление}}{l \rightarrow l + \Delta}$$

$$\begin{aligned}
 & \Delta + 2\Delta + 3\Delta + \dots + k\Delta = \\
 &= \Delta(1+2+\dots+k) = \\
 &= \Delta \cdot \frac{1}{2} \cdot k \cdot (k+1) = \\
 &= \Theta(n^2)
 \end{aligned}$$

✗

$$\text{Множение}$$

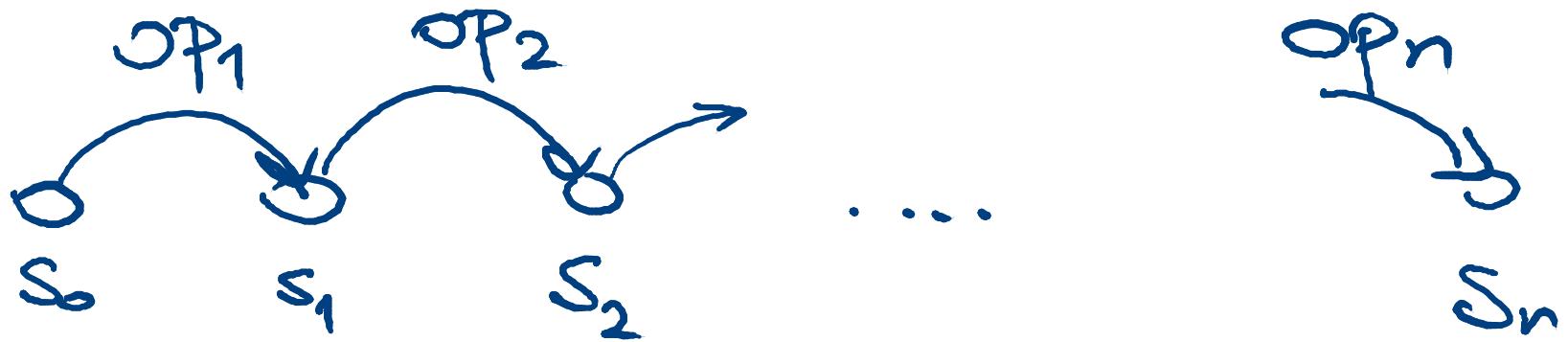
$$l \rightarrow d \cdot l$$

$$1 + d + d^2 + \dots + d^{k-1} \approx n$$

$$= \frac{d^{k+1} - 1}{d - 1} = \Theta(n)$$

✓

МЕТОД ПОТЕНЦИАЛОВ



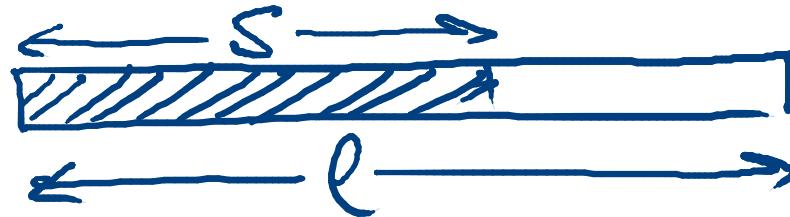
c_i - стоимость операции OP_i

φ_i - потенциал состояния s_i

$c'_i = c_i + (\varphi_i - \varphi_{i-1})$ - чистая стоимость

$$\sum_{i=1}^n c'_i = \sum_{i=1}^n c_i + (\varphi_0 - \varphi_n)$$

$$l \rightarrow 2 \cdot l$$



$$\varphi_i = 2s - l$$

1) Javasabtuve iels tepebbedeletuvi:

$$s \rightarrow s+1, l \rightarrow l, c_i = 1$$

$$c_i = 1 + \underbrace{(2(s+1) - l)}_{(2s+2-l)} - \underbrace{(2s - l)}_{(2s-l)} = 3$$

2) Javasabtuve o tepebbedeletuviem:

$$s = l, s \rightarrow s+1, l \rightarrow 2l, c_i = l+1$$

$$c_i = \underbrace{l+1}_{l+1} + \underbrace{(2(l+1) - 2l)}_{(2l+2-2l)} - \underbrace{(2l - l)}_{(2l-l)} = 3$$

$$\sum_{i=1}^n c_i = \sum_{i=1}^n c_i + (\varphi_0 - \varphi_n) = 3n + (\varphi_0 - \varphi_n) = \Theta(n)$$

