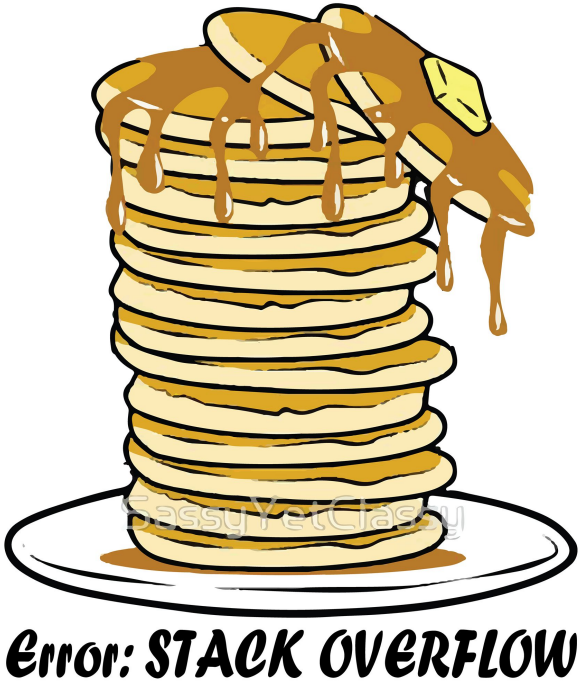
# Замена системной рекурсии стеком. Разбор формул при помощи стека - без приоритета и с приоритетом. Бинарные деревья. Бинарные деревья поиска.

ФББ. Осень 2021

# Рекурсия и ее проблемы

- Рекурсия проста и понятна для человека
- Машине приходится хранить кучу информации кроме полезных данных (порядок вызова функций, их локальные данные)
- Если рекурсия глубока -- у компьютера начинаются трудности



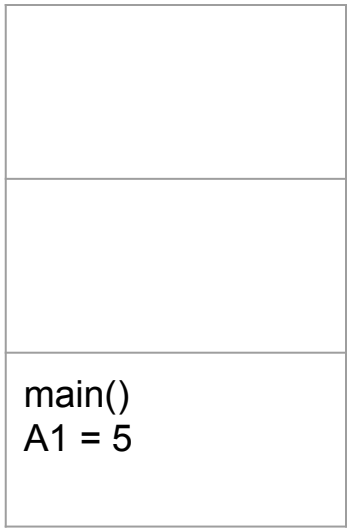
# Как хранятся вызовы функций? Стек вызовов

Вычисляется: main(5)

```
def main(A1):
    r1 = submain(A1) + 1
    return r1

def submain(A2):
    r2 = subsubmain(A2) + 1
    return r2

def subsubmain(A3):
    return 2 + A3
```



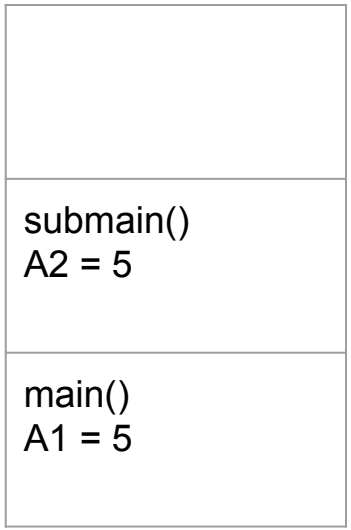
# Как хранятся вызовы функций? Стек вызовов

Вычисляется: main(5)

```
def main(A1):
    r1 = submain(A1) + 1
    return r1

def submain(A2):
    r2 = subsubmain(A2) + 1
    return r2

def subsubmain(A3):
    return 2 + A3
```



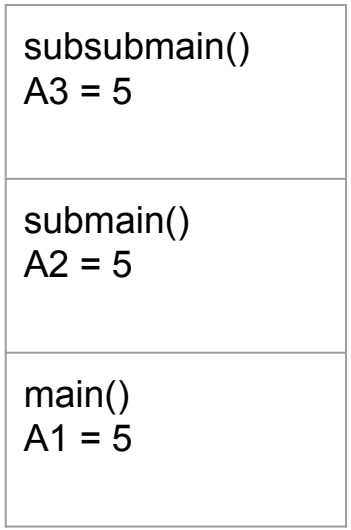
# Как хранятся вызовы функций? Стек вызовов

Вычисляется: main(5)

```
def main(A1):
    r1 = submain(A1) + 1
    return r1

def submain(A2):
    r2 = subsubmain(A2) + 1
    return r2

def subsubmain(A3):
    return 2 + A3
```



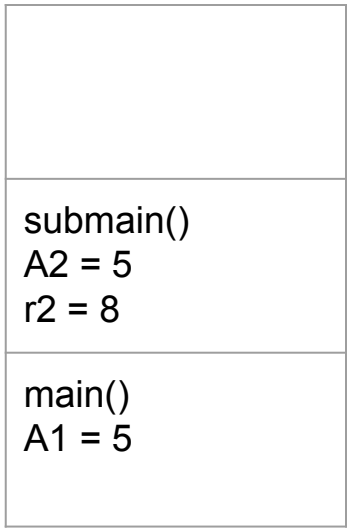
# Как хранятся вызовы функций? Стек вызовов

Вычисляется: main(5)

```
def main(A1):
    r1 = submain(A1) + 1
    return r1

def submain(A2):
    r2 = subsubmain(A2) + 1
    return r2

def subsubmain(A3):
    return 2 + A3
```



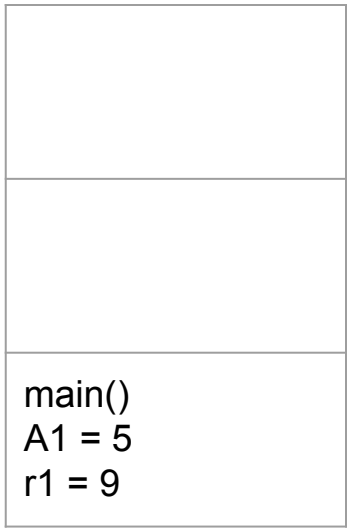
# Как хранятся вызовы функций? Стек вызовов

Вычисляется: main(5)

```
def main(A1):
    r1 = submain(A1) + 1
    return r1

def submain(A2):
    r2 = subsubmain(A2) + 1
    return r2

def subsubmain(A3):
    return 2 + A3
```



# Хвостовая рекурсия

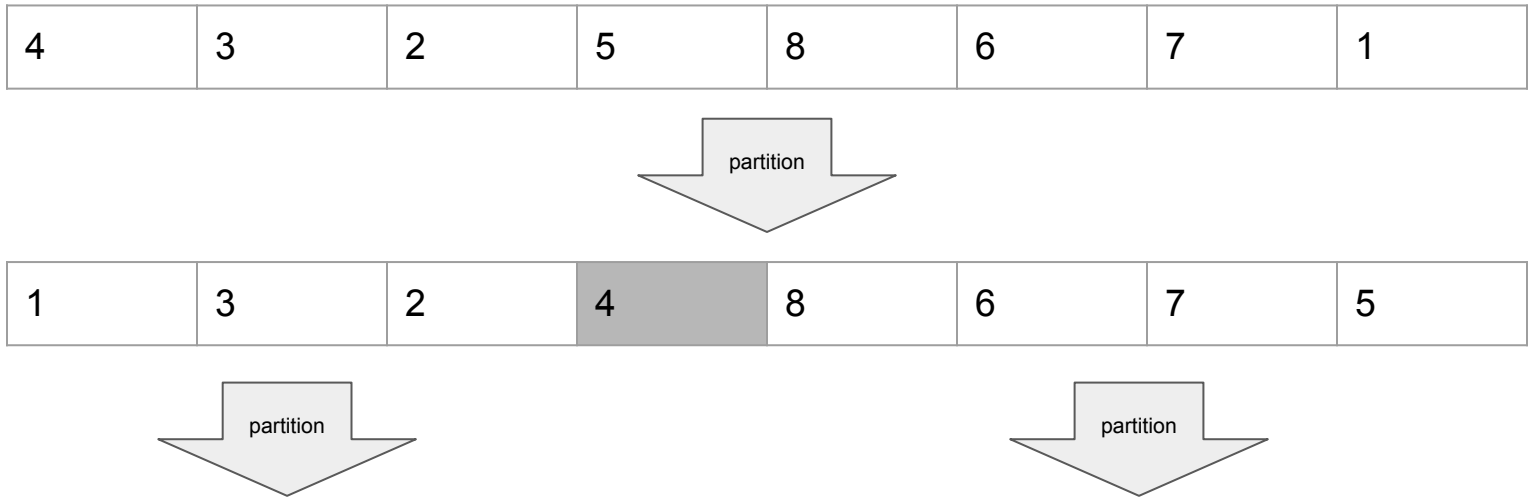
```
def bitshift(arg, shift):
    return arg if shift==0 else bitshift(arg//2, shift-1)
```

```
def bitshift(arg, shift):
    for i in range(shift):
        arg //= 2
    return arg
```

# Хвостовая рекурсия?

```
def factorial(arg):
    return 1 if arg==0 else arg * factorial(arg-1)
```

# QSort.




Какова ожидаемая длина стека вызовов?

# Если стека не хватает -- сделай свой

| 4 | 3 | 2 | 5 | 8 | 6 | 7 | 1 |
|---|---|---|---|---|---|---|---|

Что будем хранить в стеке?

# Инициация

| 4 | 3 | 2 | 5 | 8 | 6 | 7 | 1 |
|---|---|---|---|---|---|---|---|

push( (0, 8) )

| |
|---|
| |
| |
| |
| 0, 8 |

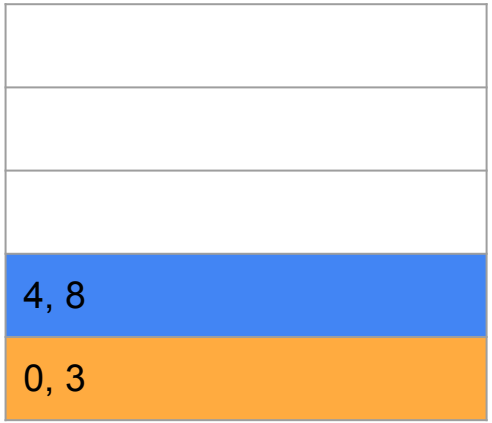
# Ход работы. До начала цикла

| 4 | 3 | 2 | 5 | 8 | 6 | 7 | 1 |
|---|---|---|---|---|---|---|---|

| |
|---|
| |
| |
| |
| 0, 8 |

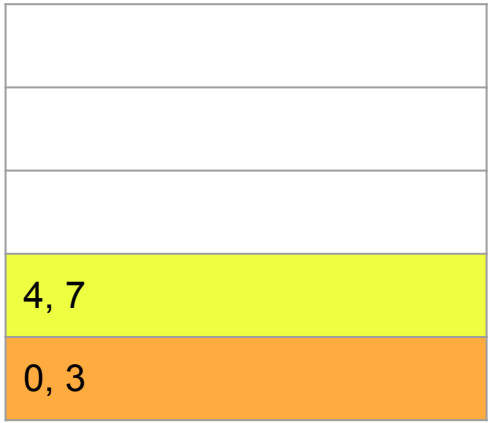
```
while stack is not empty:
        start, end = pop()
        pivot_pos = partition( start, end )
        push( (start, pivot_pos) )
        push( (pivot_pos+1, end) )
```

# Ход работы.После первого цикла

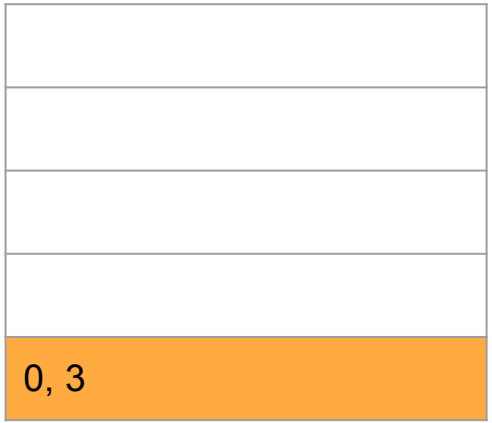

```
while stack is not empty:
        start, end = pop()
        pivot_pos = partition( start, end )
        push( (start, pivot_pos) )
        push( (pivot_pos+1, end) )
```

# Ход работы. После следующего цикла


```
while stack is not empty:
        start, end = pop()
        pivot_pos = partition( start, end )
        push( (start, pivot_pos) )
        push( (pivot_pos+1, end) )
```

# Ход работы. После еще трех циклов


```
while stack is not empty:
        start, end = pop()
        pivot_pos = partition( start, end )
        push( (start, pivot_pos) )
        push( (pivot_pos+1, end) )
```

# Ход работы. После еще пары циклов


Стек пуст. Массив отсортирован.
Конечно не стоит забывать о базовом случае -- массив из одного элемента всегда отсортирован

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | |
| | |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | |
| | 2 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | |
| + | 2 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | 3 |
| + | 2 |

# Разбор формул. Выполнено сложение

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | |
| | 5 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | |
| * | 5 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | 4 |
| * | 5 |

# Разбор формул. Выполнено умножение

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | |
| | 20 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | |
| * | 20 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| ( | |
| * | 20 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| ( | 2 |
| * | 20 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| + | |
| ( | 2 |
| * | 20 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| + | 5 |
| ( | 2 |
| * | 20 |

# Разбор формул. Выполняем сложение

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| ( | 7 |
| * | 20 |

# Разбор формул

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| ) | |
| ( | 7 |
| * | 20 |

# Разбор формул. Закрываем скобки

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

А что если две скобки не были бы рядом в стеке?

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | 7 |
| * | 20 |

# Разбор формул. Последнее умножение

| 2 | + | 3 | * | 4 | * | ( | 2 | + | 5 | ) |
|---|---|---|---|---|---|---|---|---|---|---|

| Стек операторов | Стек чисел |
|---|---|
| | |
| | |
| | |
| | |
| | 140 |

# Приоритет и ассоциативность

- Операции имеют разный приоритет
- Операции могут выполняться в разном порядке, иногда не слева направо
- Просто вычислить это не получится

2 + 2 * 2 != 8

2 ^ 2 ^ 3 = ?

# Нотации

- Обычно мы используем инфиксную нотацию (2 + 2)
- То же самое можно записать в префиксной нотации (+ 2 2) или постфиксной нотации (2 2 +)
- Постфиксная нотация называется *польской обратной нотацией*
- Такая нотация проще разбирается на компьютере

Для разбора выражения с приоритетом на придется сначала перевести выражение в обратную польскую нотацию и только затем вычислить.

# Алгоритм

- Если видим число -- пишем его в выходную очередь
- У операторов необходимо проверить приоритет.
  - Если стек пуст или операции в нем имеют меньший приоритет -- помещаем оператор в стек
  - Если операция в стеке больше или равна по приоритету -- извлекаем ее из стека и переносим в очередь пока выполняется это условие. Добавляем оператор в стек
- При встрече со скобкой -- достаем из стека все до открывающей скобки и переносим в очередь

# Пример

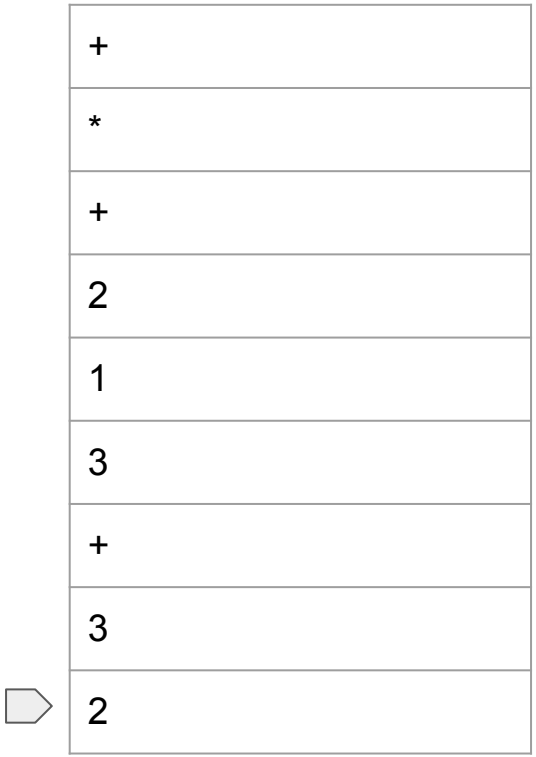
2 + 3 + 3 * (1 + 2)

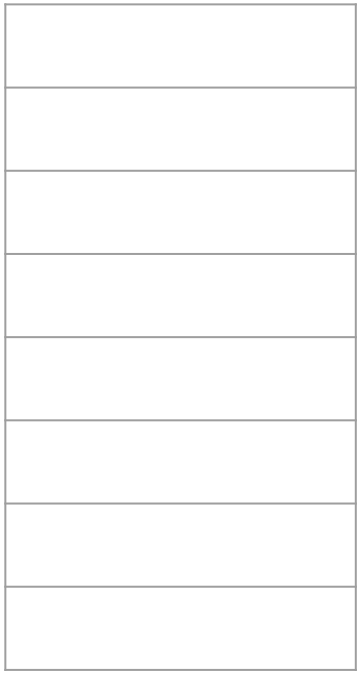
| Символ | Операция | Стек | Выходная строка |
|---|---|---|---|
| 2 | добавить к выходной строке | | 2 |
| + | поместить в стек | + | 2 |
| 3 | добавить к выходной строке | + | 2, 3 |
| + | 1) присоединить стек в обратном порядке к выходной строке; 2) поместить новую операцию в стек. | + | 2, 3, + |
| 3 | добавить к выходной строке | + | 2, 3, +, 3 |
| * | поместить в стек | +, * | 2, 3, +, 3 |
| ( | поместить в стек | +, *, ( | 2, 3, +, 3 |
| 1 | добавить к выходной строке | +, *, ( | 2, 3, +, 3, 1 |
| + | поместить в стек | +, *, (, + | 2, 3, +, 3, 1 |
| 2 | добавить к выходной строке | +, *, (, + | 2, 3, +, 3, 1, 2 |
| ) | 1) присоединить содержимое стека до скобки в обратном порядке к выходной строке; 2) удалить скобку из стека. | +, * | 2, 3, +, 3, 1, 2, + |

2 3 + 3 1 2 + * +

# Вычисление

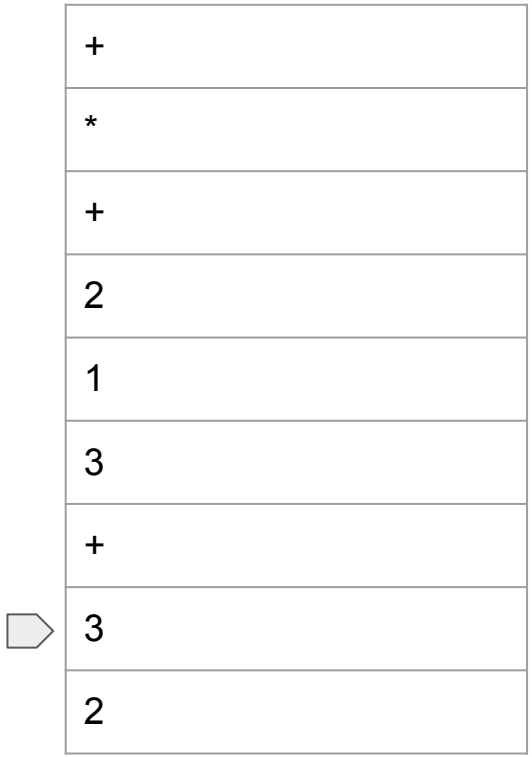
2 3 + 3 1 2 + * +

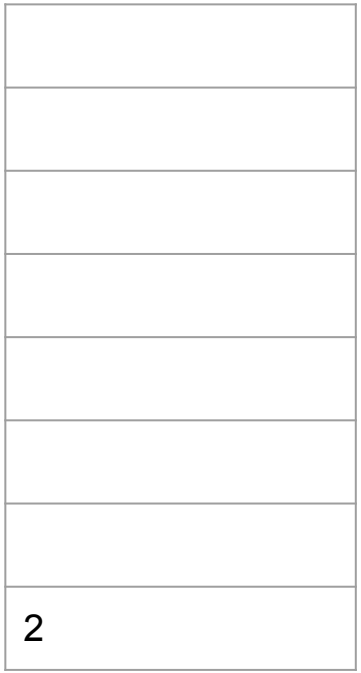


# Вычисление

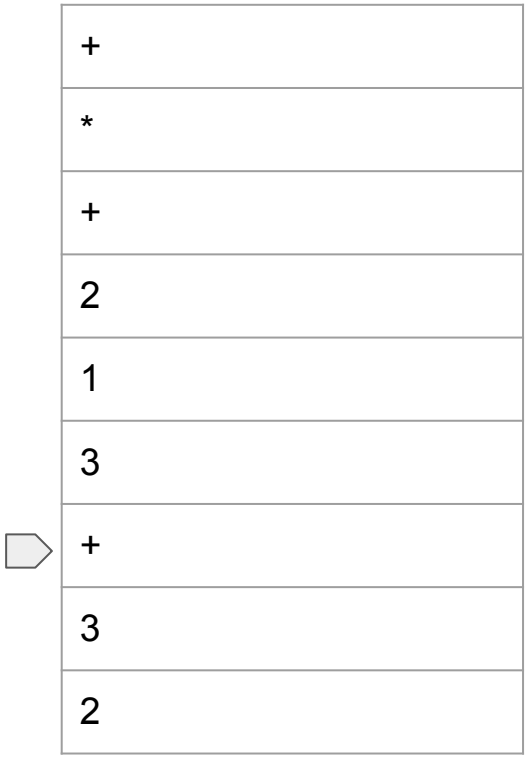
2 3 + 3 1 2 + * +

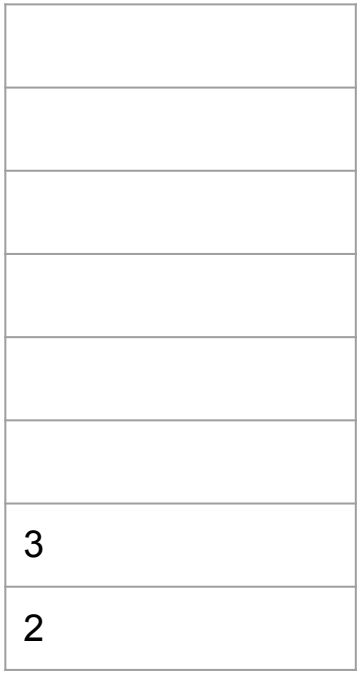


# Вычисление

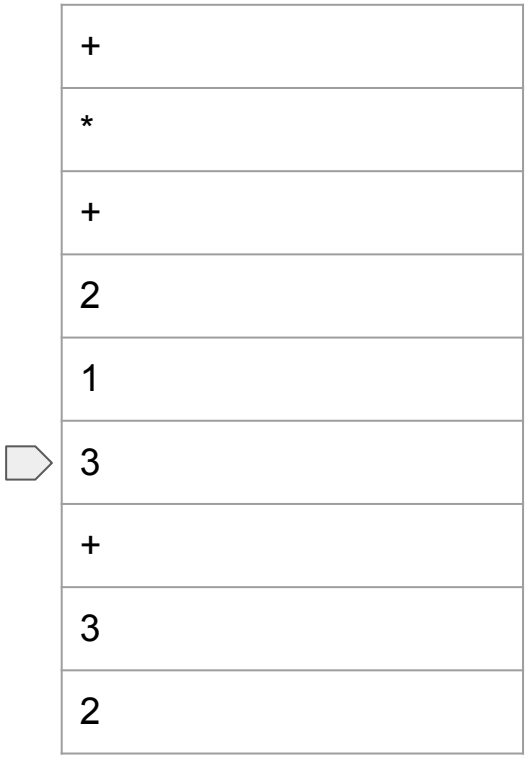
2 3 + 3 1 2 + * +

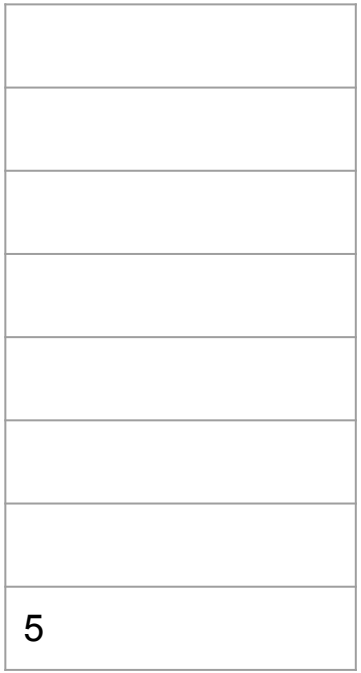


# Вычисление

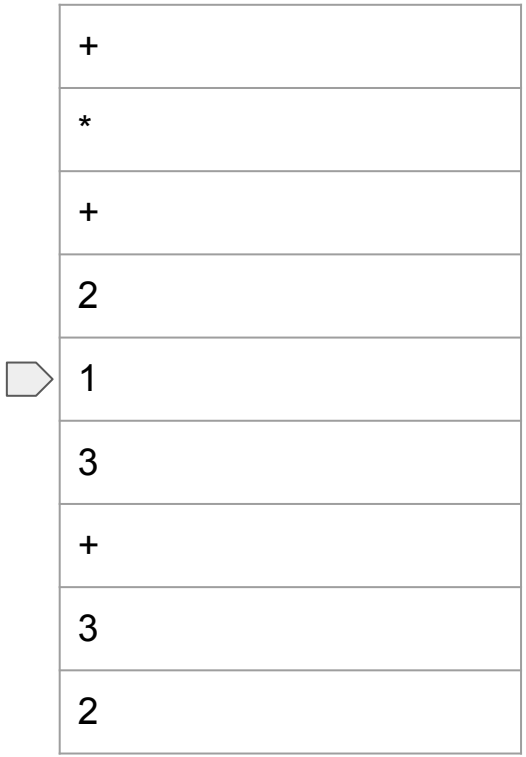
2 3 + 3 1 2 + * +

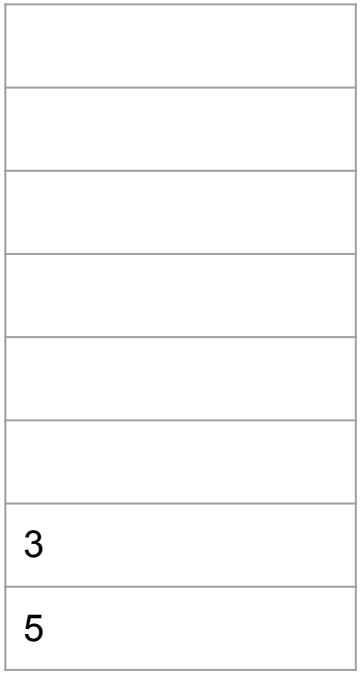


# Вычисление

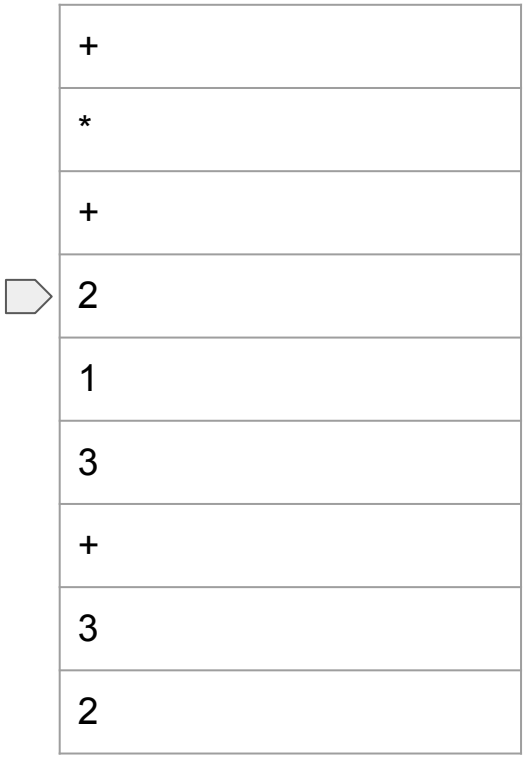
2 3 + 3 1 2 + * +

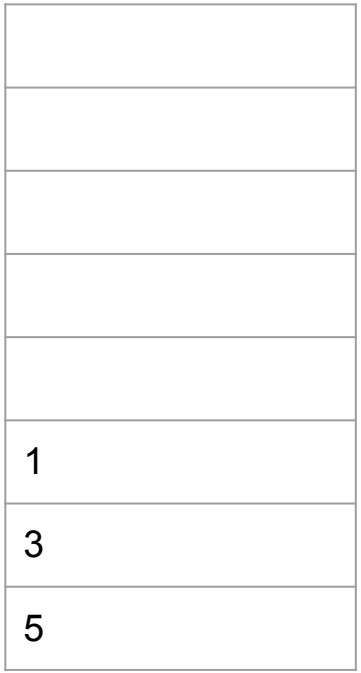


# Вычисление

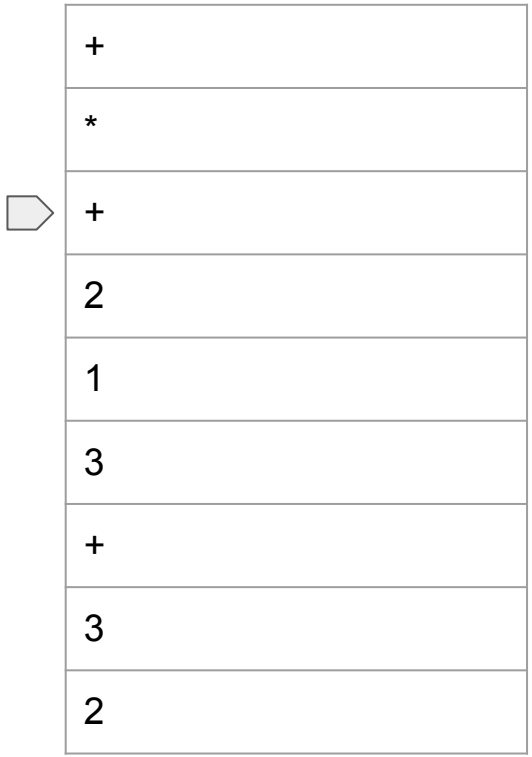
2 3 + 3 1 2 + * +

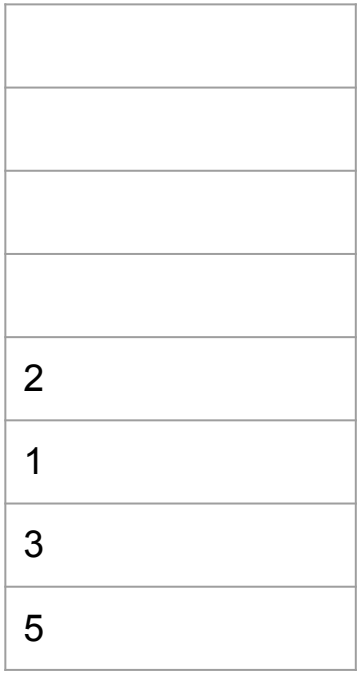


# Вычисление

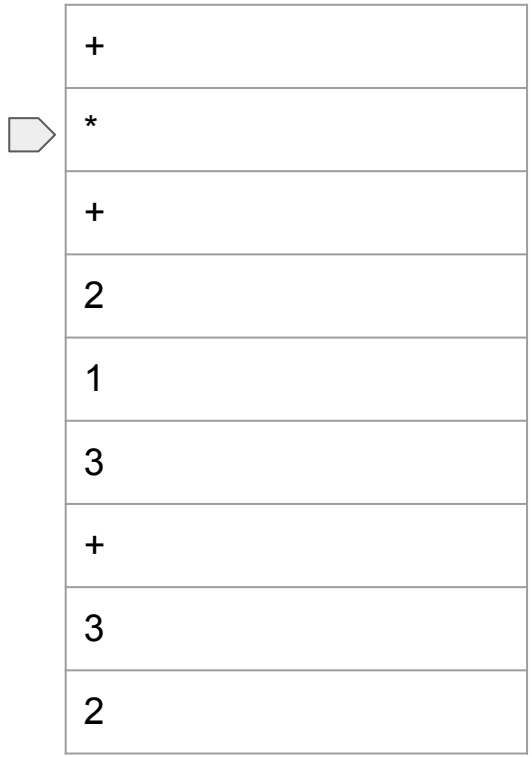
2 3 + 3 1 2 + * +

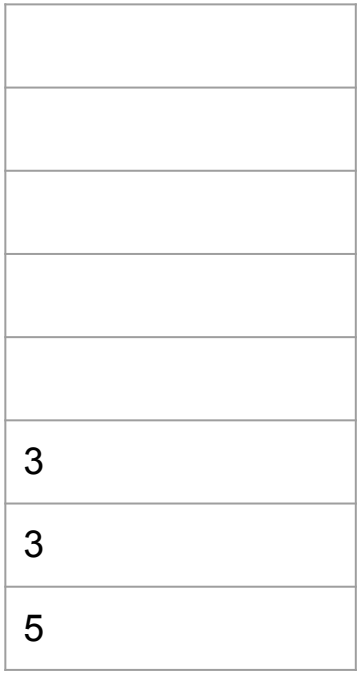


# Вычисление

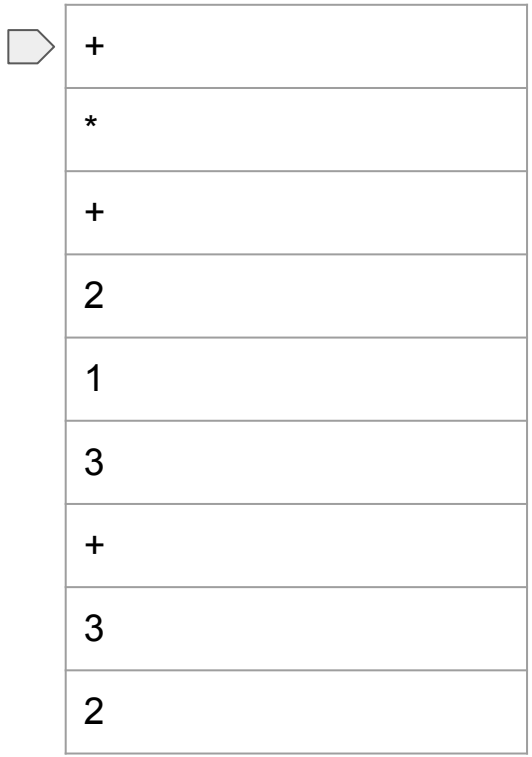
2 3 + 3 1 2 + * +

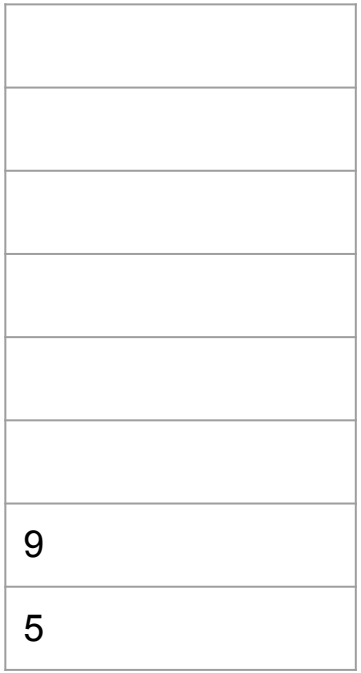


# Вычисление

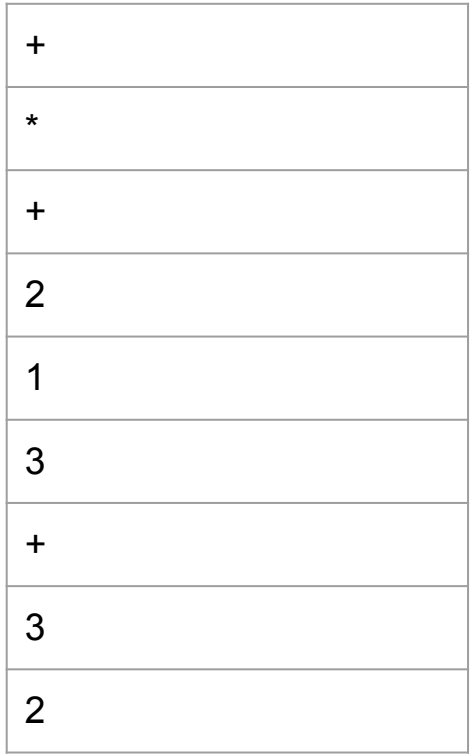
2 3 + 3 1 2 + * +

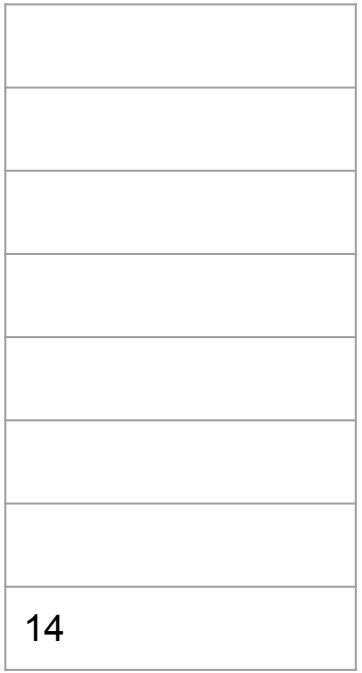


# Вычисление

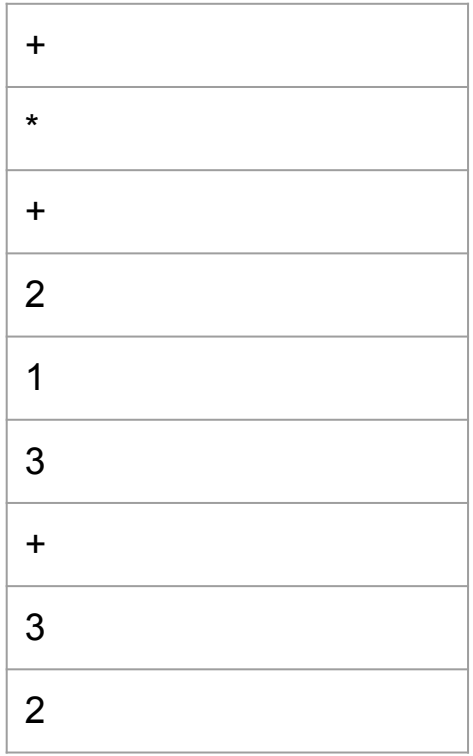
2 3 + 3 1 2 + * +

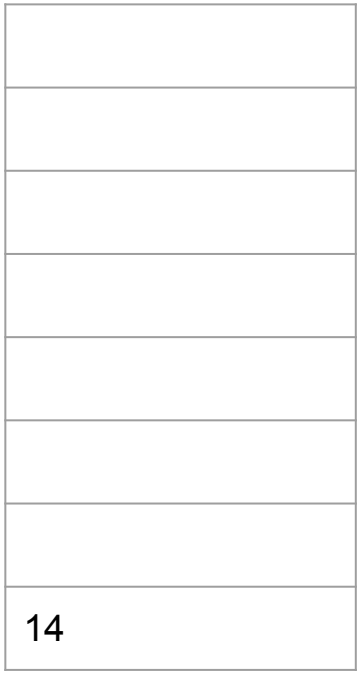


# Вычисление

2 3 + 3 1 2 + * +



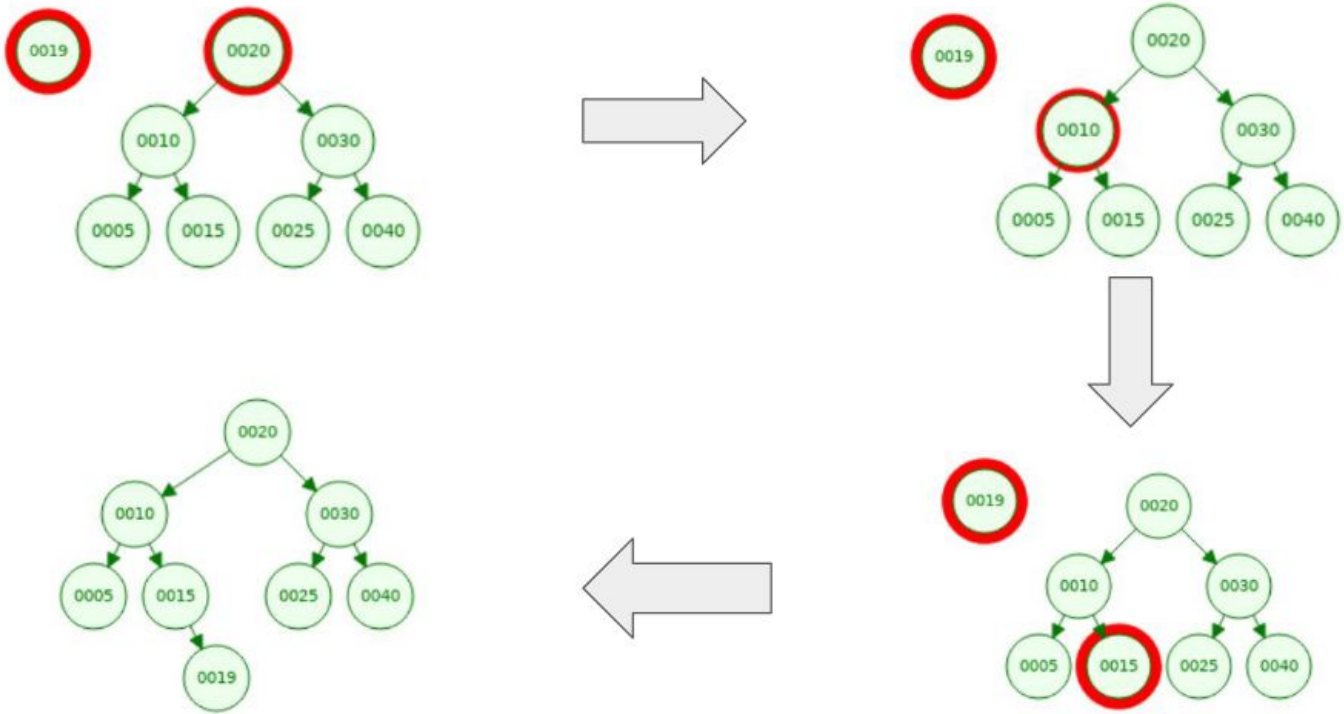



# Бинарные деревья.

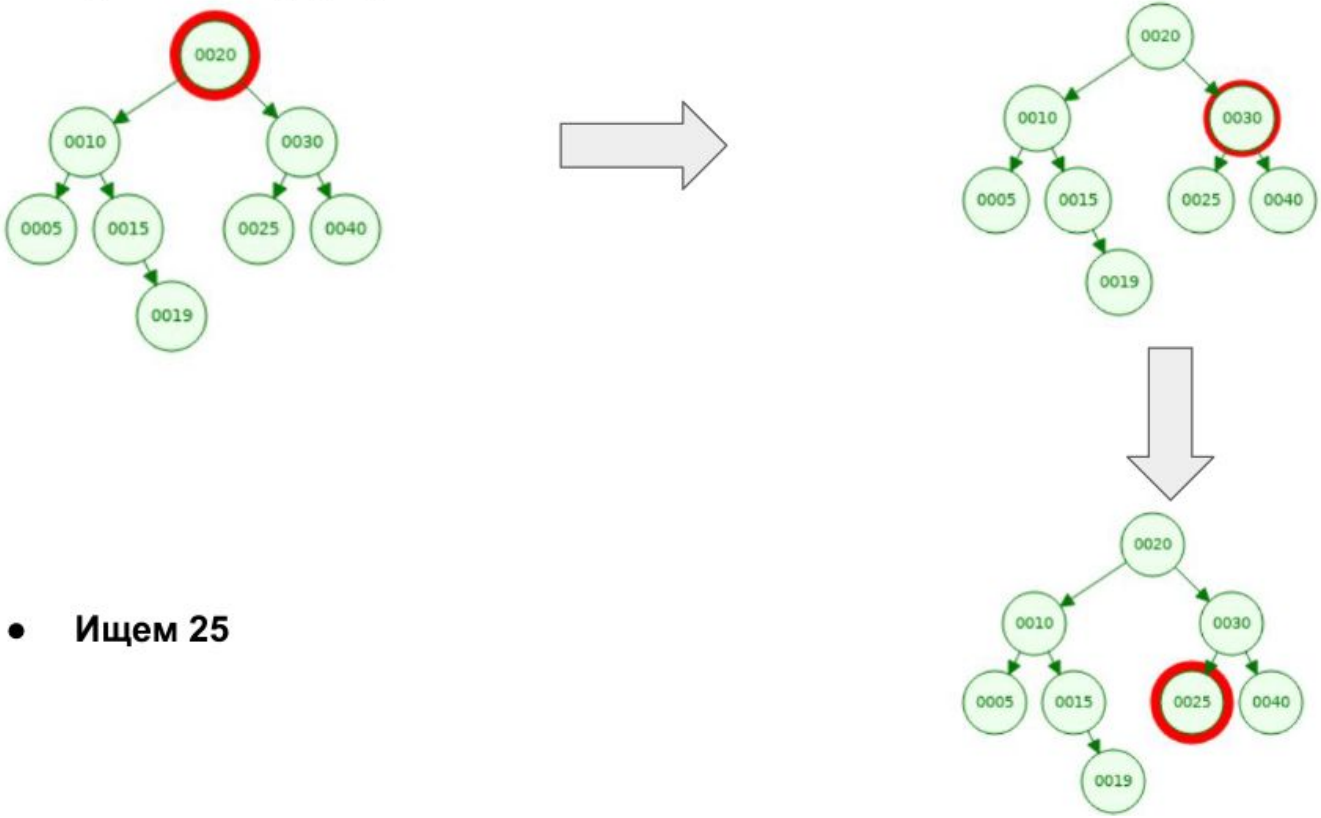
- Состоит из элементов (нод)
- Каждая нода хранит значение и ссылки на некоторые другие нода.
- Этими другими нодами могут быть родительская нода, левый потомок и правый потомок.


*https://www.cs.usfca.edu/~galles/visualization/BST.html*

# Бинарные деревья поиска. Вставка элемента



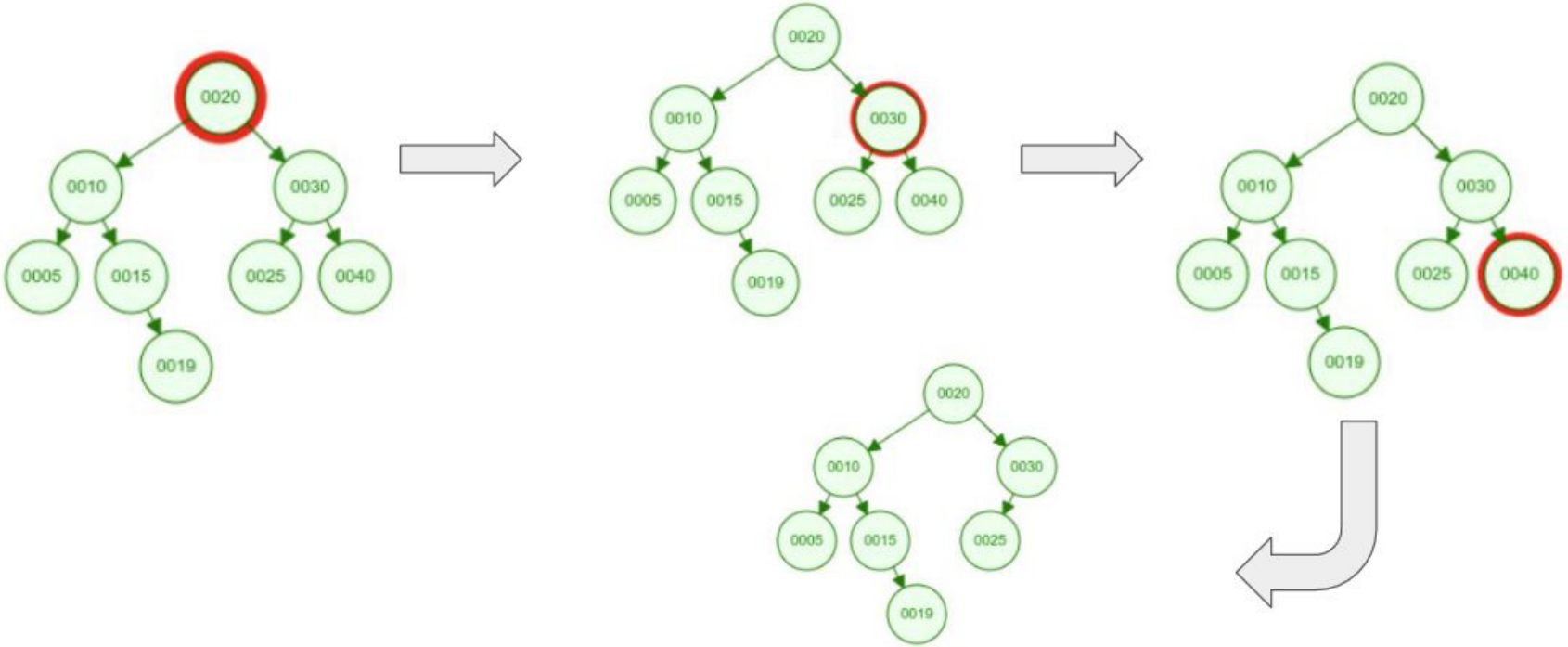
# Бинарные деревья поиска. Поиск элемента




- **Ищем 25**

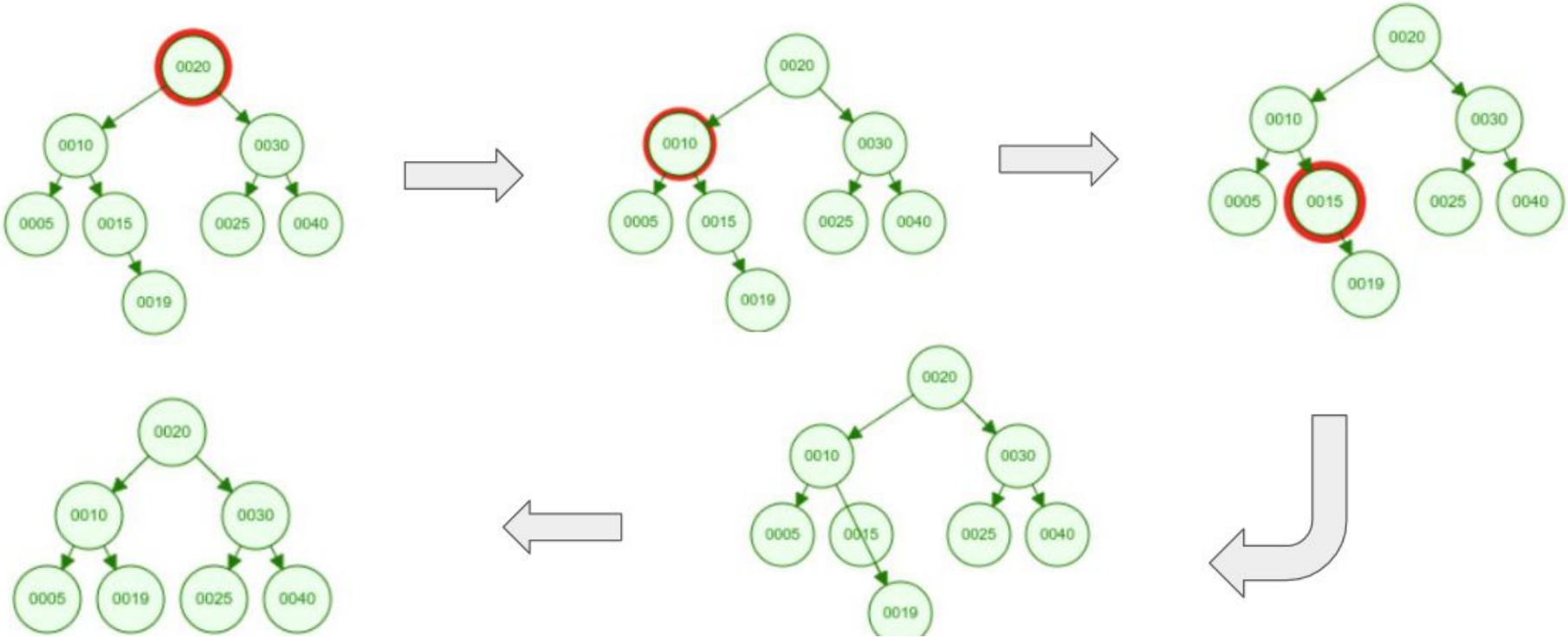
# Бинарные деревья поиска. Удаление элемента.

- Случай первый. Когда потомков нет. Удаляем 40.



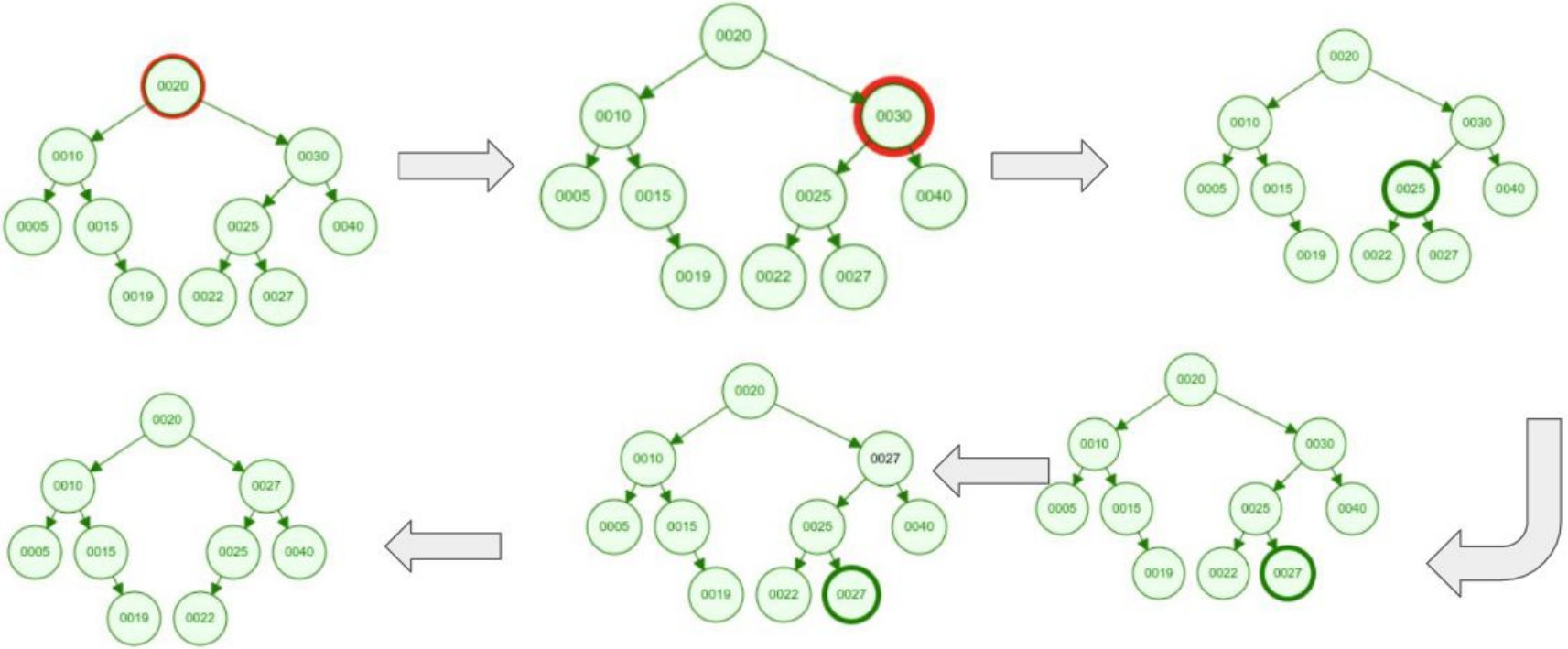
# Бинарные деревья поиска. Удаление элемента.

- Случай второй. Когда когда есть один потомок. Удаляем 15.



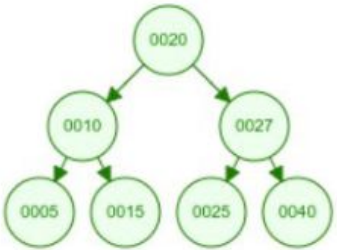
# Бинарные деревья поиска. Удаление элемента.

- Случай третий. Когда потомков два. Удаляем 30.

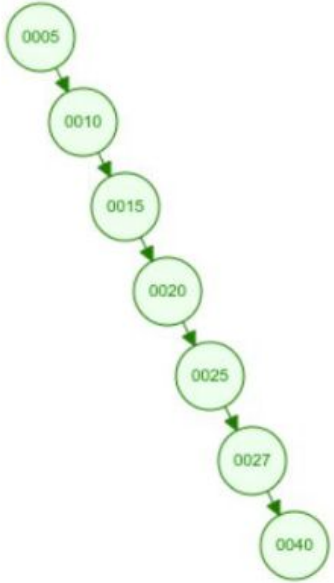


# Время работы Бинарного дерева поиска.

| | В среднем случае | В худшем случае |
|---|---|---|
| Добавление элемента | O(logN) | O(N) |
| Поиск элемента | O(logN) | O(N) |
| Удаление элемента | O(logN) | O(N) |




Лучший случай




Худший случай

*https://www.cs.usfca.edu/~galles/visualization/BST.html*