

ИВАНОВ А.Б.

# АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

АБСТРАКТНЫЕ ТИПЫ ДАННЫХ  
СТРУКТУРЫ ДАННЫХ  
АССОЦИАТИВНЫЕ МАССИВЫ

# АБСТРАКТНЫЕ ТИПЫ ДАННЫХ (АТД, ADT)

Стек, очередь, дек, список...

Абстрактный тип данных = интерфейс

Стек:

- добавление элемента / проталкивание (push)
- удаление элемента (pop)
- чтение головного элемента (peek)

# РЕАЛИЗАЦИЯ СТЕКА (СВЯЗНЫЙ СПИСОК)

Связанный список

Сложность в худшем случае:

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

# РЕАЛИЗАЦИЯ СТЕКА (ДИНАМИЧЕСКИЙ МАССИВ)

На основе динамического массива

Сложность в худшем случае:

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

# АМОРТИЗАЦИОННЫЙ АНАЛИЗ

На основе динамического массива

Амортизированная сложность - средняя сложность операции в последовательности  $c_i, i = 1 \dots n, :$

$$\frac{\sum_{i=1}^n c_i}{n}$$

- push -  $O^*(1)$
- pop -  $O^*(1)$
- peek -  $O(1)$

## МЕТОД ПРЕДОПЛАТЫ (БУХГАЛТЕРСКИЙ)

Амортизированная сложность  $\hat{c}_i, i = 1 \dots n, :$

$$\sum_{i=1}^n c_i \leq \sum_{i=1}^n \hat{c}_i$$

Свяжем с каждым добавлением и удалением "бюджет" в несколько  $\text{Р}$ , который должен покрыть будущие перемещения. Сколько?  $\text{Р}3$  достаточно!

# МЕТОД ПОТЕНЦИАЛОВ

$D_i$  - состояние структуры данных после  $i$  - операций.  
Тогда потенциал (зависит только от состояния):

$$\Phi : \mathbb{N} \rightarrow \mathbb{R}_0^+$$

Амортизированная сложность

$$\hat{c}_i = c_i + \Phi(i) - \Phi(i - 1)$$

Тогда

$$\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n c_i + \Phi(n) - \Phi(0)$$

## МЕТОД ПОТЕНЦИАЛОВ

$$\Phi(n) = \begin{cases} 2n - N, & \text{если } \frac{N}{2} \leq n \leq N \\ -n + \frac{N}{2}, & \text{если } \frac{N}{4} \leq n < \frac{N}{2} \end{cases}$$

$n$  - текущий размер структуры,  $N$  - размер массива

# АССОЦИАТИВНЫЙ МАССИВ (MAP, DICTIONARY)

Абстрактный тип данных:

набор пар <ключ, значение> (ключ уникальный).

Операции (интерфейс):

- Вставка
- Поиск по ключу
- Удаление по ключу

## РЕАЛИЗАЦИЯ - МАССИВ

- Вставка -  $O^*(1)$
- Поиск по ключу -  $O(n)$
- Удаление по ключу -  $O(1)$

## РЕАЛИЗАЦИЯ - ОТСОРТИРОВАННЫЙ МАССИВ

- Вставка -  $O(n)$
- Поиск по ключу -  $O(\log n)$
- Удаление по ключу -  $O(n)$

# РЕАЛИЗАЦИЯ АССОЦИАТИВНОГО МАССИВА

- Массив
- Отсортированный массив
- Бинарные деревья поиска
- Хеш-таблицы

# ПРИМЕНЕНИЯ АССОЦИАТИВНОГО МАССИВА

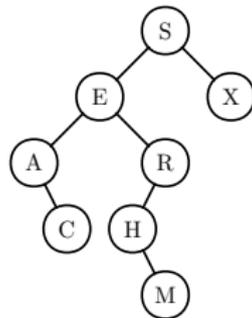
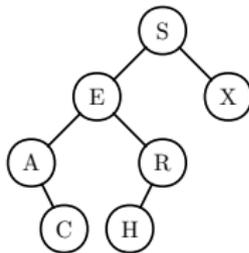
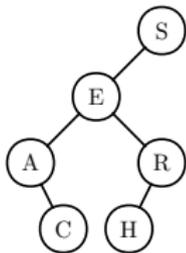
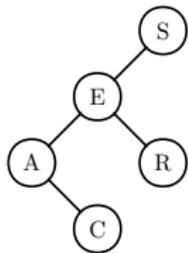
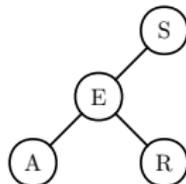
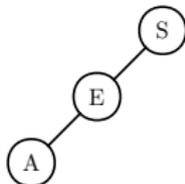
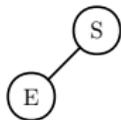
- Базовая структура в программировании ( мемоизация )
- Сети (DNS, маршрутизация)
- Алгоритмы (2SUM)
- ...

## BINARY SEARCH TREE: ПРИМЕР

S, E, A, R, C, H, X, M

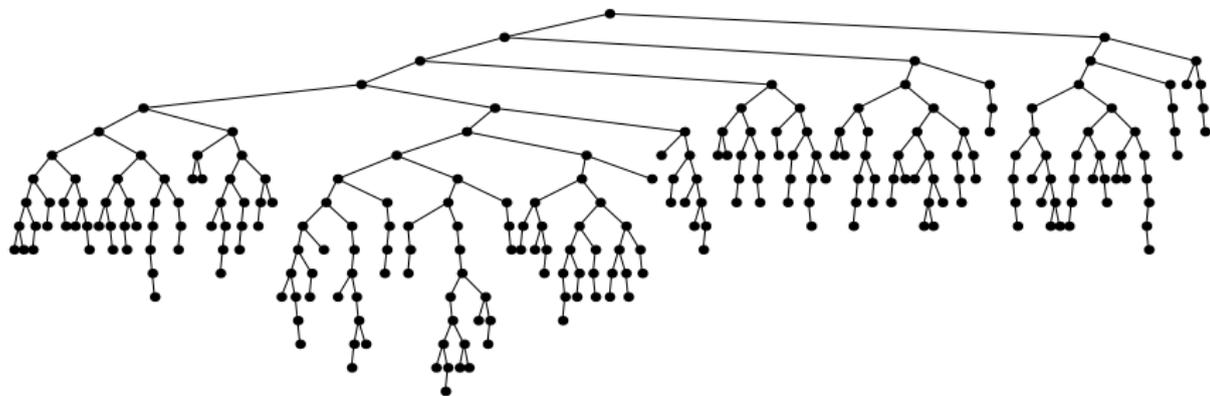
# BINARY SEARCH TREE: ПРИМЕР

S, E, A, R, C, H, X, M



# Симуляция 1 ( $N = 255$ )

max = 18      avg = 9.31



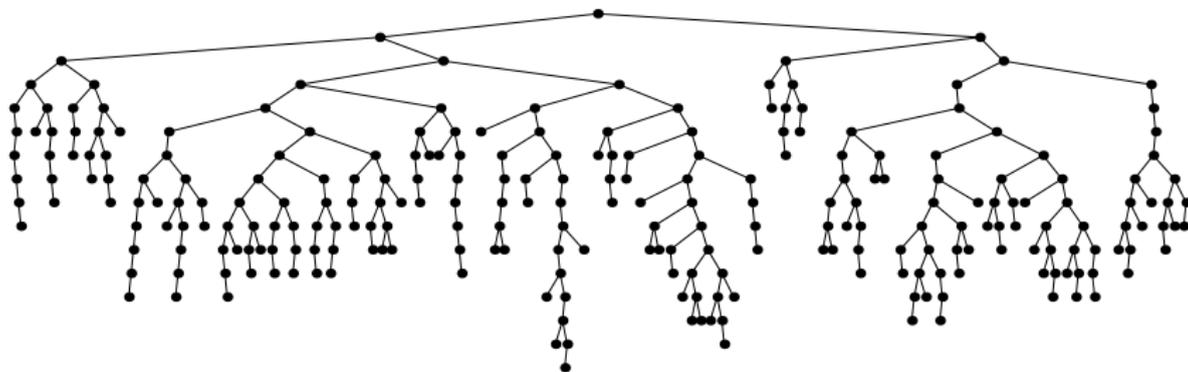
Полностью сбалансированное дерево:

max = 8      avg = 7.03

# Симуляция 2 ( $N = 255$ )

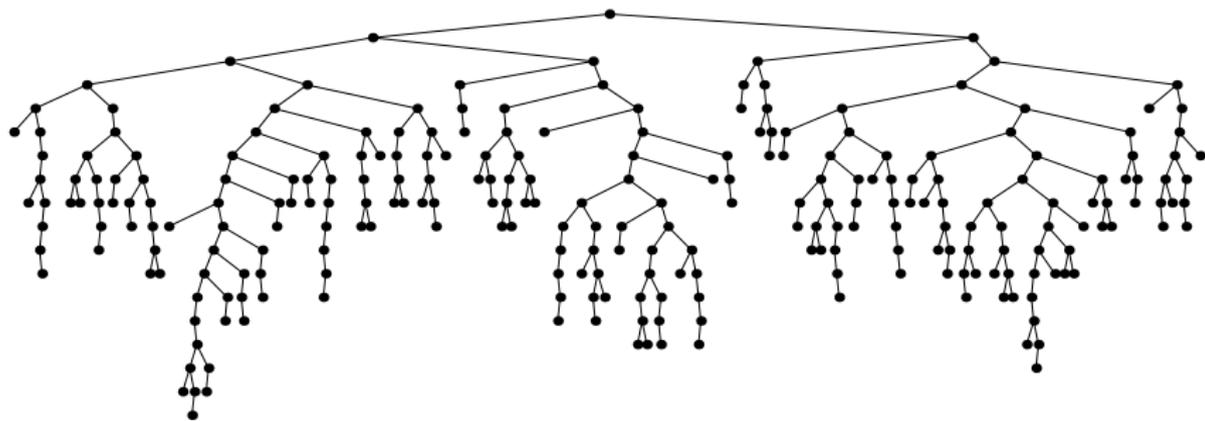
max = 17

avg = 9.13



# Симуляция 3 ( $N = 255$ )

max = 16      avg = 9.07



## СЛОЖНОСТЬ В СЛУЧАЙНОМ BST

**УТВ.** Search hit в случайном бинарном дереве поиска, содержащем  $N$  ключей требует в среднем  $\sim 2 \ln N$  ( $\approx 1.39 \log N$ ) сравнений.

Пусть  $C_N$  – сумма длин путей до всех узлов в дереве с  $N$  узлами (тогда успешный поиск требует в среднем  $1 + C_N/N$ ).

$$C_N = N-1 + (C_0 + C_{N-1})/N + (C_1 + C_{N-2})/N + \dots + (C_{N-1} + C_0)/N$$

$$C_0 = C_1 = 0$$

(рекуррентное соотношение аналогично тому, которое возникало при анализе Quicksort, см. лекцию 3).

$$C_N \sim 2N \ln N$$

## СЛОЖНОСТЬ В СЛУЧАЙНОМ BST

**УТВ.** Вставка и search miss в случайном бинарном дереве поиска, содержащем  $N$  ключей требует в среднем  $\sim 2 \ln N$  ( $\approx 1.39 \log N$ ) сравнений.

Вставка и search miss требует на 1 сравнение больше, чем search hit (задача).