Адрианов Н.М.
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 ₽, который должен покрыть будущие перемещения. Сколько? ₽3 достаточно!

Метод потенциалов

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

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

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

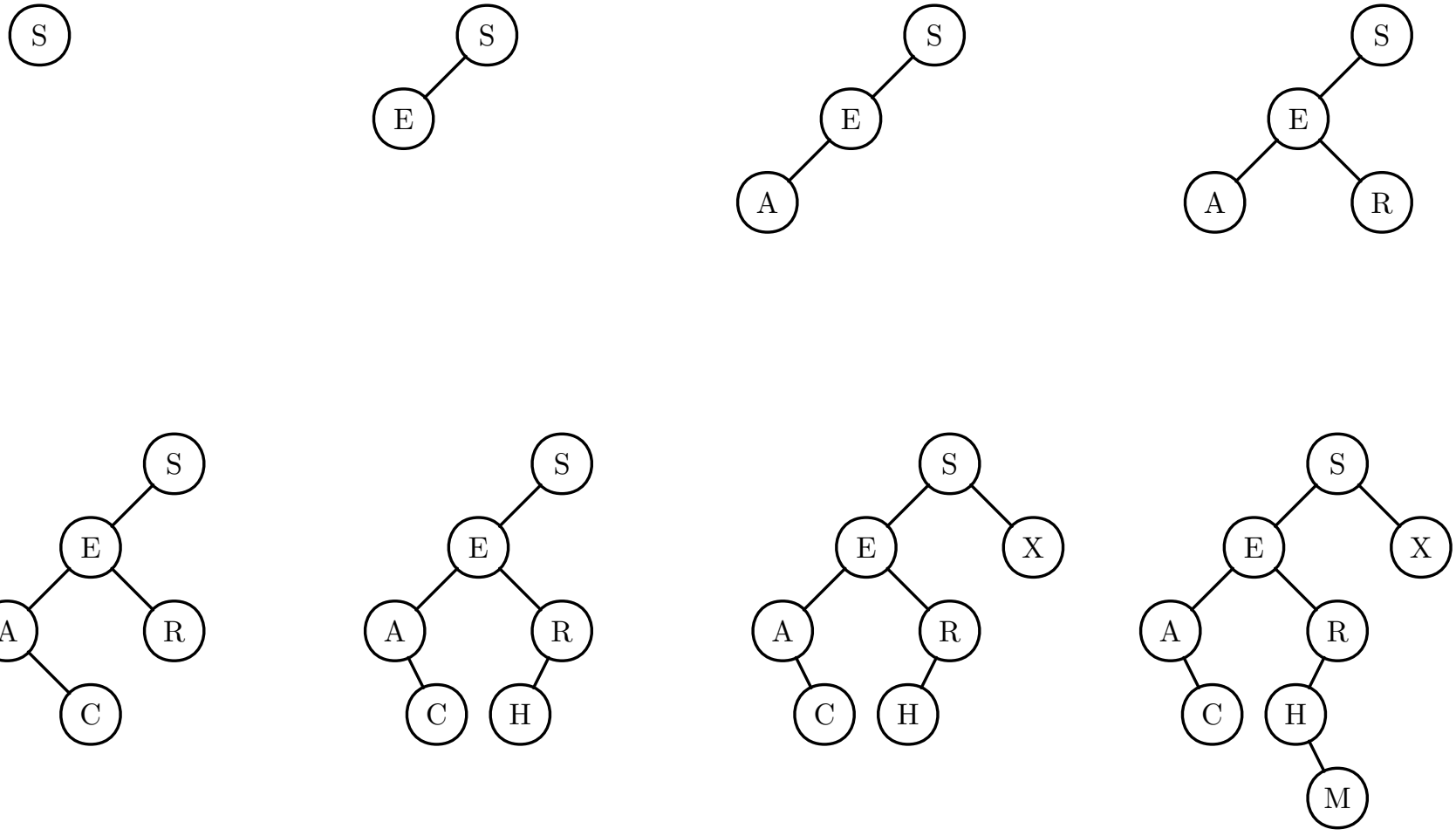
- Базовая структура в программировании (мемоизация)
- Сети (DNS, маршрутизация)
- Алгоритмы (2SUM)
- ...

Binary Search Tree: пример

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

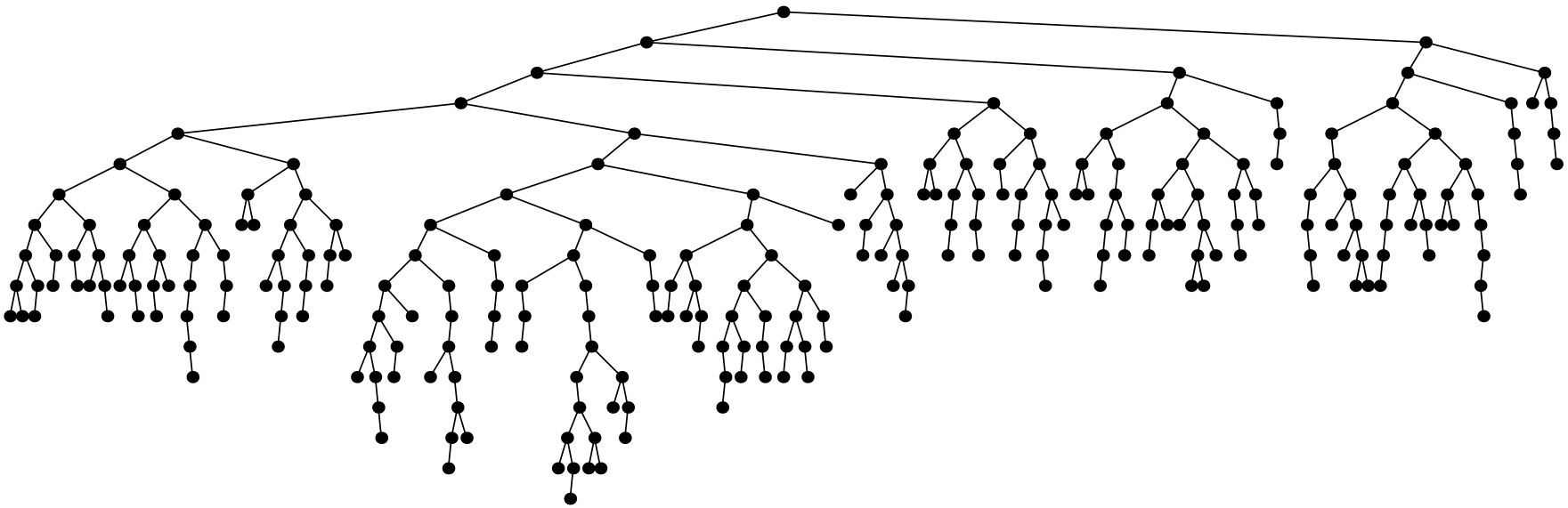
Binary Search Tree: пример

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



Симуляция 1 ($N = 255$)

$\max = 18 \qquad \text{avg} = 9.31$

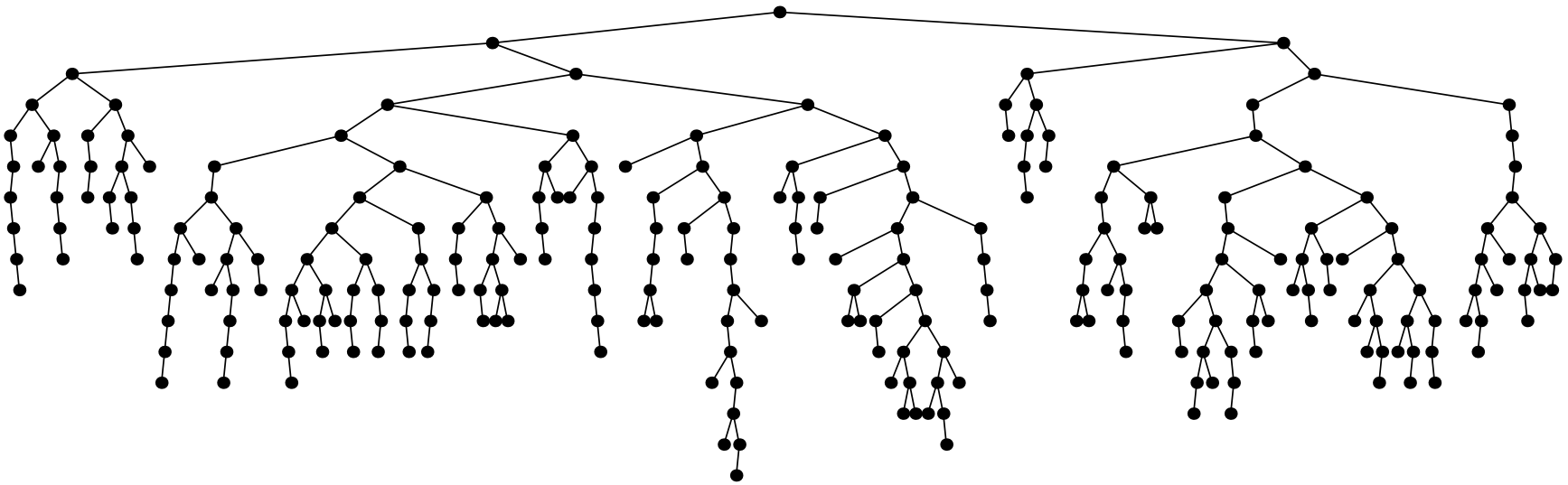


Полностью сбалансированное дерево:

$\max = 8 \qquad \text{avg} = 7.03$

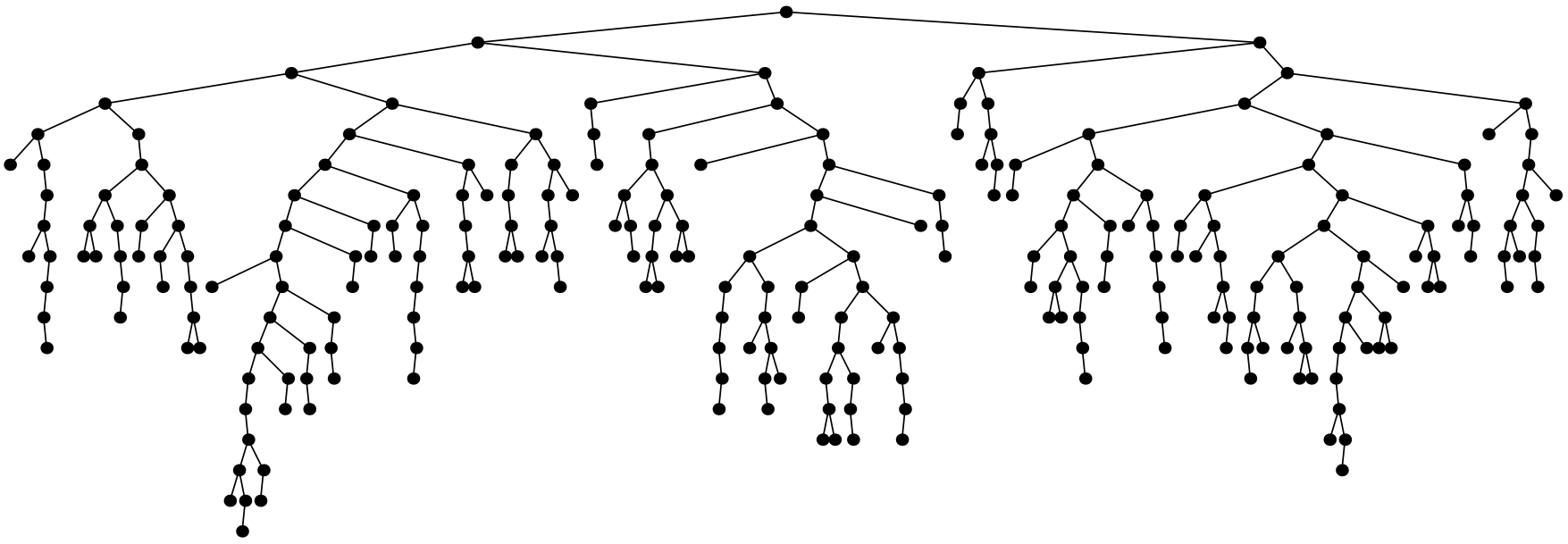
Симуляция 2 ($N = 255$)

$\max = 17 \qquad \text{avg} = 9.13$



Симуляция 3 ($N = 255$)

$\max = 16 \qquad \text{avg} = 9.07$



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 $1 + C_N/N$).

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

$$C_0 = C_1 = 0$$

(рекурентное 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).