БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СБОРНИК ЗАДАЧ

ПО ТЕОРИИ АЛГОРИТМОВ.

Структуры данных.
Часть. Специальные структуры данных.
Бинарная куча.

МИНСК
2020

УДК 510.51(075.8)
ББК 22.12я73-1
С23

А в т о р ы :
С. А. Соболь, К. Ю. Вильчевский, В. М. Котов,
Е. П. Соболевская

Р е ц е н з е н т ы :
кафедра информатики и методики преподавания информатики
физико-математического факультета Белорусского государственного
педагогического университета им. М. Танка
(заведующий кафедрой, кандидат педагогических наук,
доцент *С. В. Вабищевич*);
профессор кафедры информационных технологий в культуре
Белорусского государственного университета культуры и искусства,
кандидат физико-математических наук, доцент *П. В. Гляков*

ISBN 978-985-518-412-4

Часть 1

СПЕЦИАЛЬНЫЕ СТРУКТУРЫ ДАННЫХ

1.1. БИНАРНАЯ КУЧА

Куча (англ. *heap*) — специализированная древовидная структура данных, которая удовлетворяет *свойству кучи*. В вершинах древовидной структуры хранятся ключи. Различают два варианта куч: *min-heap* и *max-heap*. Для min-heap свойство кучи формулируется следующим образом: если вершина с ключом y является потомком вершины с ключом x, то $x \leqslant y$. Когда рассматривается max-heap, знак в неравенстве меняется на противоположный. Для определённости в этом разделе будем работать с первым вариантом — min-heap.

Для куч определён следующий базовый набор операций:

1) GetMin() — поиск минимального ключа;
2) ExtractMin() — удаление минимального ключа;
3) Insert(x) — добавление ключа x.

Расширенный набор операций включает также следующие:

4) IncreaseKey и DecreaseKey — модификация ключа вершины на заданную величину (предполагается, что известна позиция вершины внутри структуры данных);
5) Heapify — построение кучи для последовательности из n ключей.

Существует много способов реализации структуры данных «куча» с помощью корневых деревьев. Наиболее простым способом является реализация с помощью *полного бинарного дерева*. Такая куча называется *бинарной кучей* (англ. *binary heap*), или *пирамидой*.

Напомним, что полное бинарное дерево — это такое корневое дерево, в котором каждая вершина имеет не более двух сыновей, а заполнение вершин осуществляется в порядке от верхних уровней к нижним,

причём на одном уровне заполнение вершинами производится слева направо. Пока уровень полностью не заполнен, к следующему уровню не переходят. Последний уровень может быть заполнен не полностью. Высота полного бинарного дерева, содержащего n вершин, — $\mathrm{O}(\log n)$.

1.1.1. Представление в памяти

В памяти компьютера полное бинарное дерево легко реализуется с помощью массива (индексы массива начинаются с единицы). Для элемента с индексом i сыновьями являются элементы с индексами $2i$ и $2i + 1$, а родителем является элемент массива по индексу $\lfloor i/2 \rfloor$.

Пример 1.1. На рис. 1.1 приведён пример полного бинарного дерева из десяти вершин.

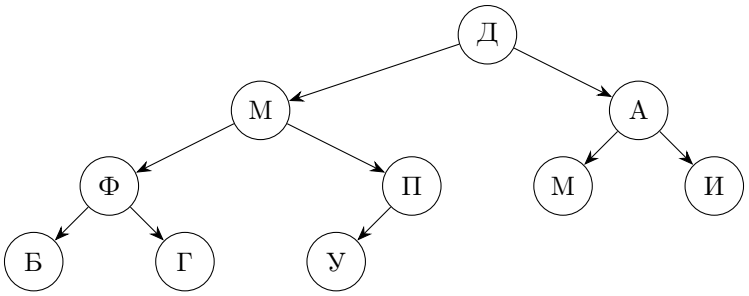

Рис. 1.1. Полное бинарное дерево

В памяти компьютера указанное полное бинарное дерево будет храниться в массиве следующим образом:

1	2	3	4	5	6	7	8	9	10
Д	М	А	Ф	П	М	И	Б	Г	У

Пример 1.2. На рис. 1.2 приведён пример бинарной кучи.

В памяти компьютера эта бинарная куча будет храниться в массиве следующим образом:

1	2	3	4	5	6	7	8	9	10
1	5	2	13	7	4	3	14	18	7

Кроме массива, который используется для хранения ключей бинарной кучи, вводится переменная целого типа, определяющая количество элементов в куче.

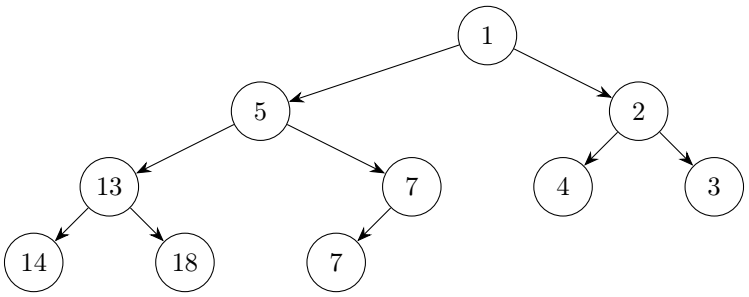

Рис. 1.2. Бинарная куча (min-heap)

1.1.2. Основные операции

Время выполнения базовых операций для бинарной кучи, содержащей n вершин, следующее:

- GetMin() — $O(1)$;
- ExtractMin() — $O(\log n)$;
- Insert(x) — $O(\log n)$.
- IncreaseKey и DecreaseKey — $O(\log n)$;
- Heapify — $O(n)$.

Более подробно с описанием процедур, реализующих интерфейс структуры данных «бинарная куча», можно ознакомиться, например, в [3].

1.1.3. Псевдокод

Опишем на псевдокоде основные операции, которые можно выполнять с бинарной кучей (для определённости, min-heap). Массивы в псевдокоде индексируются с нуля, как во всех распространённых языках программирования, поэтому требуется определить, как отношения между элементами связаны с их индексами. Для перехода от 1-индексации к 0-индексации в формулы, указанные в начале раздела 1.1.1, вместо i подставим $i' = i + 1$, затем из результата вычтем 1.

- Сыновьями элемента i являются элементы с индексами

$$2(i+1) - 1 = 2i + 1, \quad 2(i+1) + 1 - 1 = 2i + 2.$$

- Родителем элемента i является элемент

$$\lfloor (i+1)/2 \rfloor - 1 = \lfloor (i-1)/2 \rfloor.$$

Пусть в коде `a` — динамический массив, который используется для хранения кучи (индексы начинаются с 0). Тогда `len(a)` — число элементов в этом массиве. Изначально массив пуст.

Операция Insert(x) работает следующим образом. Ключ добавляется в конец массива. Возможно, при этом свойство кучи нарушится, поэтому нужно восстановить это свойство, выполнив проталкивание элемента вверх. Каждый раз ключ текущего элемента с индексом i сравнивается с ключом его родителя с индексом j, при необходимости выполняется обмен ключей.

```
def Insert(a, x):
    a.append(x)

    i = len(a) - 1
    while i > 0:
        j = (i - 1) // 2  # a[j] is the parent of a[i]
        if a[j] <= a[i]:
            break
        a[i], a[j] = a[j], a[i]  # swap
        i = j
```

Операция получения минимума GetMin() тривиальна: минимум находится в корне дерева, нужно просто вернуть его.

```
def GetMin(a):
    return a[0]
```

Операция извлечения минимума ExtractMin() работает так. Берём последний элемент (крайний правый элемент на нижнем уровне дерева) и ставим его в корень. Возможно, при этом свойство кучи нарушится, поэтому нужно восстановить это свойство, выполнив проталкивание элемента вниз по дереву. Каждый раз для элемента с ключом i определяется индекс потомка j, ключ которого наименьший, и при необходимости выполняется обмен.

```
def ExtractMin(a):
    a[0] = a[len(a) - 1]
    a.pop()

    i = 0
    while 2 * i + 1 < len(a):
        if (2 * i + 2 == len(a)) or (a[2 * i + 1] < a[2 * i + 2]):
            j = 2 * i + 1  # left child
        else:
            j = 2 * i + 2  # right child
        if a[i] <= a[j]:
            break
        a[i], a[j] = a[j], a[i]  # swap
        i = j
```

На практике бинарную кучу редко приходится реализовывать самостоятельно, поскольку готовые решения есть в стандартных библиотеках многих языков программирования. Однако важно понимать, как именно устроена эта структура данных.

Классы из разных языков программирования, внутри которых скрыты бинарные кучи, перечислены в разделе **??**. Кроме того, в C++ STL доступна серия алгоритмов `std::make_heap`, `std::push_heap`, `std::pop_heap` и др. Эти функции позволяют построить кучу на базе любой последовательности элементов.

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1. Алгоритмы: построение и анализ / Т. Кормен [и др.]. — М. : Вильямс, 2005. — 1296 с.

2. *Котов В. М.*, *Мельников О. И.* Информатика. Методы алгоритмизации : учеб. пособие для 10 – 11 кл. общеобразоват. шк. с углубл. изучением информатики. — Минск : Нар. асвета, 2000. — 221 с.

3. *Котов В. М.*, *Соболевская Е. П.*, *Толстиков А. А.* Алгоритмы и структуры данных : учеб. пособие. — Минск : БГУ, 2011. — 267 с. — (Классическое университетское издание).

4. Сборник задач по теории алгоритмов : учеб.-метод. пособие/ В. М. Котов [и др.]. — Минск : БГУ, 2017. — 183 с.

5. Теория алгоритмов : учеб. пособие / П. А. Иржавский [и др.]. — Минск : БГУ, 2013. — 159 с.

6. *Соболь С. А.*, *Котов В. М.*, *Соболевская Е. П.* Опыт использования образовательной платформы Insight Runner на факультете прикладной математики и информатики Белорусского государственного университета // Роль университетского образования и науки в современном обществе : материалы междунар. науч. конф., Минск, 26–27 февр. 2019 г. / Белорус. гос. ун-т ; редкол.: А. Д. Король (пред.) [и др.]. — Минск : БГУ, 2019. — С. 263-267.

7. *Соболь С. А.*, *Котов В. М.*, *Соболевская Е. П.* Методика преподавания дисциплин по теории алгоритмов с использованием образовательной платформы iRunner // Судьбы классического университета: национальный контекст и мировые тренды [Электронный ресурс] : материалы XIII Респ. междисциплинар. науч.-теорет. семинара «Инновационные стратегии в современной социальной философии» и междисциплинар. летней школы молодых ученых «Экология культуры», Минск, 9 апр. 2019 г. / Белорус. гос. ун-т ; сост.: В. В. Анохина, В. С. Сайганова ; редкол.: А. И. Зеленков (отв. ред.) [и др.] — С. 346–355.

СОДЕРЖАНИЕ

Учебное издание

Соболь Сергей Александрович
Вильчевский Константин Юрьевич
Котов Владимир Михайлович и др.

СБОРНИК ЗАДАЧ ПО ТЕОРИИ АЛГОРИТМОВ. СТРУКТУРЫ ДАННЫХ

Учебно-методическое пособие

Редактор *Х. Х. XXXXXXX*
Художник обложки *С. А. Соболь*
Технический редактор *Х. Х. XXXXXXX*
Компьютерная вёрстка *С. А. Соболя*
Корректор *Х. Х. XXXXXX*

Подписано в печать 29.02.2020. Формат 60×84/16. Бумага офсетная.
Печать офсетная. Усл. печ. л. 10,69. Уч.-изд. л. 9,6.
Тираж 150 экз. Заказ

Белорусский государственный университет.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/270 от 03.04.2014.
Пр. Независимости, 4, 220030, Минск.

Республиканское унитарное предприятие
«Издательский центр Белорусского государственного университета».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №2/63 от 19.03.2014.
Ул. Красноармейская, 6, 220030, Минск.