Зимняя сессия 2024

4) Турнирная сортировка. Оценка сложности

Идея: алг-м использует подход "турнира": элем-ы сравниваются попарно, победитель (min/max) проходит дальше. Строится бинарное дерево сравнений, в котором на вершине находится min/max элем-т. После этого победитель исключается и алг-м повторяется для оставшихся элементов.

Сложность: на 1ом уровне сравниваются n элементов, на 2ом - уже $\frac{n}{2}$ и т.д. $\Rightarrow$ глубина - $\log_2 n$, на каждом ~~повторении~~ в худшем случае n сравнений. Построение турнира - $O(n)$. Алг-м должен извлечь всех победителей $\Rightarrow n$ повторений. Общая сложность - $O(n \log n)$.

Худший случай: массив из всех/всех, кроме одного, одинаковых эл-в - $O(n^2)$.

Лучший случай: массив сразу отсортир. - $O(n \log n)$.

Средний случай: рандомный массив — $O(n \log n)$. Одинаково, т.к. структура та же, только не знаем заранее, кто будет победителем, доп. сравнений не требуется.

5) Понятие пирамиды. Построение пирамиды. Процедура просеивания и оценка сложности.

Пирамида — структура данных, представляющая бинарным деревом, которое удовлетворяет следующим свойствам:

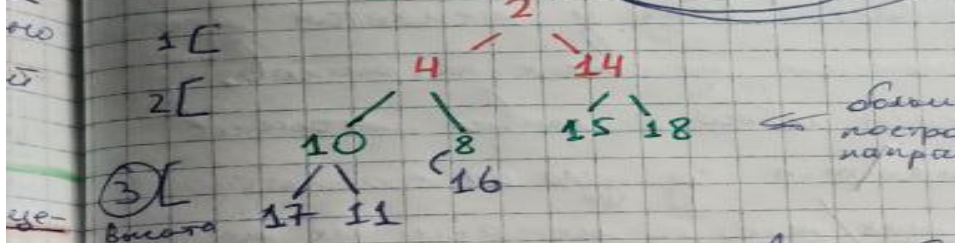
1) у каждого узла не более 2 потомков;
2) значение в каждом узле $\geq$ / $\leq$ значениями его потомков.

Корень дерева находится в 1й ячейке массива: A[0] — корень. Для узла с индексом i:

- левый потомок — $2i+1$
- правый потомок — $2i+2$
- родитель: $\frac{i-1}{2}$.

Потому что логика построения. Пример:

A = [2, 4, 14, 10, 8, 15, 18, 17, 11, 16]



Высота пирамиды – кол-во рёбер от вершины до самого дальнего листа.

Для построения пирамиды мы просеиваем эл-ты сверху вниз, начиная с 1го внут-го узла, т.е. первого узла с потомками. Первый внут. узел на поз-ции $\frac{n}{2}-1$, где n – длина исходного массива (узел внут-й, если его левый потомок $2i+1 \leq n-1 \Rightarrow i \leq \frac{n}{2}-1$). Двигаемся от $\frac{n}{2}-1$ до 0, т.к. эл-ты от $\frac{n}{2}$ до $n-1$ – листья, применяя Sift-Down.

↑ индекс i-го эл-та массива

Используется для восстан-я св-в пирамиды. Если

значение в i не удовлет-т св-вам пирамиды, то оно "проседается" вниз и меняется местами с большим/меньшим из своих потомков.

Для узла i:
- вычисляем индексы левого и правого потомков
- определяем max\min из узла и его потомков
- если узел не max/min, то меняем их местами
- рекурсивно вызываем для потомка пока пирамида не будет восстановлена

Т.к. на каждом ур-не максимум 1 обмен, то время работы — $O(\log n)$.
высота дерева

6) Сортировка

1) Массив → max\min пирамида
2) n раз:
- меняем местами max/min корень с *последним эл-м массива (наг-я куча всегда имеет max/min эл-т в корне)
- размер пирамиды −1
- восстан. св-ва пирамиды через Sift-Down

Оценка: для каждого из n эл-в время восстанов-

*ПОСЛЕДНИМ НЕ БУКВАЛЬНО, А ПОСЛЕДНИМ НЕ ТРОНУТЫМ (желтым)

ПРИМЕР: [1,7,3,4,6,9] → [1,6,3,4,7,9]

ления пирамидой $\log n$ — $O(n \log n)$.

Плюсы:
1) Лучший и средний случаи — $O(n \log n)$
2) Не требует доп. памяти

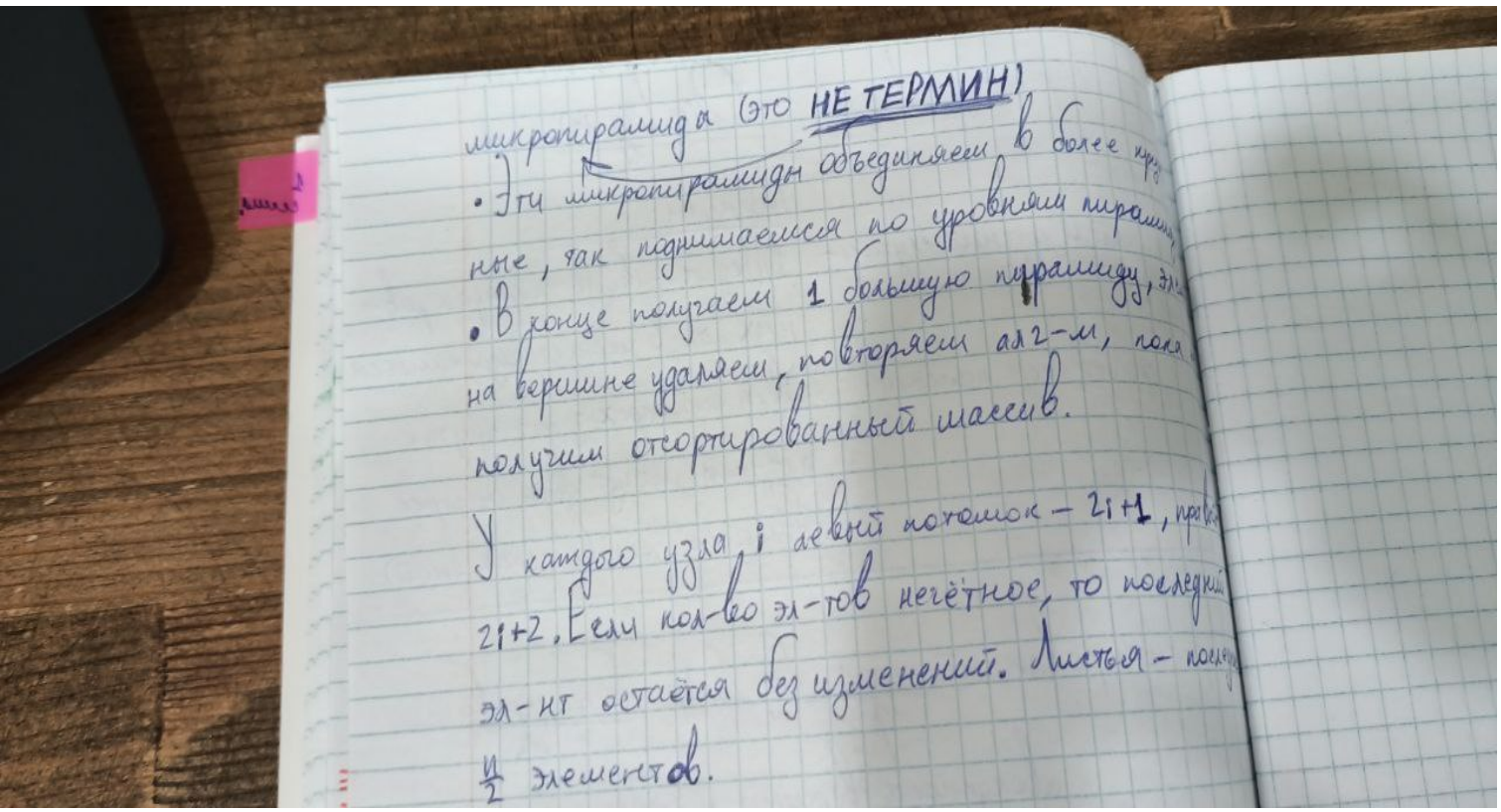
Минусы:
1) Неустойчивость — порядок равных эл-в может измениться
2) Для некоторых задач другие алг-мы могут быть быстрее (большое кол-во сравнений)

Неориентированный граф без циклов — бин. дерево
совокупность вершин и рёбер.
Пирамида — разновидность

Снизу вверх

Начинаем с листьев дерева, т.е. с элем-в массива, и постепенно объединяем их в пирамидину.

Идея:
- Массив делится на пары эл-в и из этих пар создаются



микропирамида (это НЕ ТЕРМИН)

• Эти микропирамиды объединяем в более кр
ные, так поднимаемся по уровням пирами
• В конце получаем 1 большую пирамиду, эл
на вершине удаляем, повторяем алг-м, пока
получим отсортированный массив.

У каждого узла i левый потомок – $2i+1$, пра
$2i+2$. Если кол-во эл-тов нечётное, то последний
эл-нт остаётся без изменений. Листья – послед
$\frac{n}{2}$ элементов.