1) Решение за $O(n\log n)$

Алгоритм:

1. вставляем новый элемент в конец массива
2. сравниваем его с родителем
3. если элемент < родителя — меняем их местами
4. повторяем пока элемент не станет $\geq$ родителя или не дойдем до корня

Почему это гарантирует min heap?

- до вставки все элементы в куче уже удовлетворяли свойствам
- вставка элемента влияет только на путь от листа к корню
- просеивание вверх изменяет только этот путь, остальные поддеревья не трогаем

Сложность:

вставка $i$-го элемента:

- наихудший случай — нужно подняться от листа к корню
- высота кучи с $i$ элементами $\approx \log_2 i$

значит вставка $= O(\log i)$

$$O(\log 1 + \log 2 + \ldots + \log n) = O(\log(n!)) =$$

$$= O(n\log n - n) = \underline{O(n\log n)}$$

$O(n)$

Алгоритм:

1. Определяем последний внутренний узел.

2. Обрабатываем все внутренние узлы в обратном порядке:
   начинаем с последнего и идём к корню
   для каждого узла вызываем sift_down.

3. Операция sift-down:
   - сравниваем текущий узел с его детьми
   - если один из детей меньше, меняем узел с минимальным ребёнком
   - продолжаем процесс, пока узел не станет < детей или не достигнем листа

Сложность $O(n)$

$$\sum_{k=0}^{k} (\text{кол-во узлов на уровне } k)\ O(h-k)$$

$$= O\left(\sum_{k=0}^{k} 2^k (h-k)\right)$$

$h-k=i$

$$O\left(\sum 2^{h-i} i\right) = O\left(2^h \sum \frac{i}{2^i}\right)$$

$$S = \sum x^i = \frac{1}{1-x}$$

$$S' = \sum i x^{i-1} = \frac{1}{(1-x)^2} \Rightarrow \sum i x^i = \frac{x}{(1-x)^2}$$

$$\Rightarrow \sum \frac{i}{2^i} = \frac{\frac{1}{2}}{(1-\frac{1}{2})^2} = 2$$

$$T = O(2^h \cdot 2) = O(2^h) =$$

$$\left[ \begin{array}{l} h = \lceil \log_2 n \rceil \\ 2^h \leq 2^{\log_2 n} = n \end{array} \right] = \underline{O(n)}$$