Priority queue: Пример я: 100, 200, 50, 300, 250, 500, 40, 81

Что такое кучи:

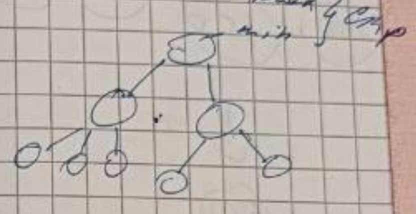
Скажем, что у нас есть набор эл-тов $K_1, K_2, \dots, K_n$ — будет наз. кучей, если будет выполняться след. условие $K_{[j/2]} \geq K_j$

([..] - округление вниз) for $1 \leq [j/2] < j \leq n$

=> $K_1 > K_2$, $K_1 > K_3$, $K_1 > K_4$

$K_2 > K_4$

$K_1 = max(K_1, K_2, K_3 \dots K_n)$
min
стр



1. Binary Heap (приоритетная очередь)
(бинарная куча)
2. Left-ist Heap - косая куча (левая - left)
3. Skew Heap
4. Binomial Heap
5. Fibonacci Heap
6. Treap

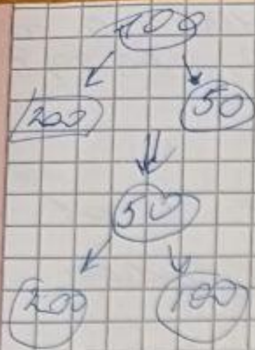
49

① Binary Heap

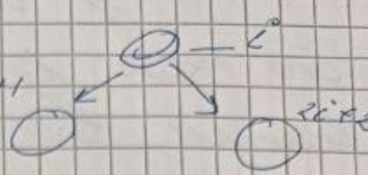
Правила:
(insert)
1. Новый эл-т в конец массива -> просеивается
2. get Priority - root ee array[0]
3. Remove - удаляем array[0] на его место ставим last

①

|100|200|50|

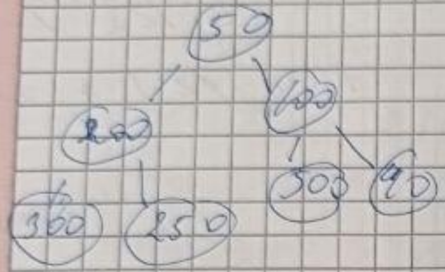


|50|200|100|



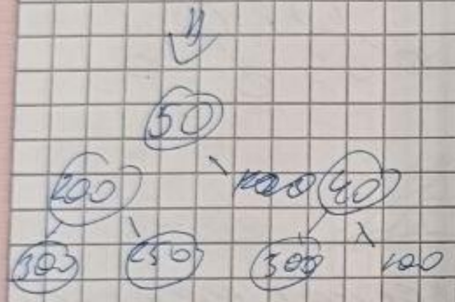
Добавим 300

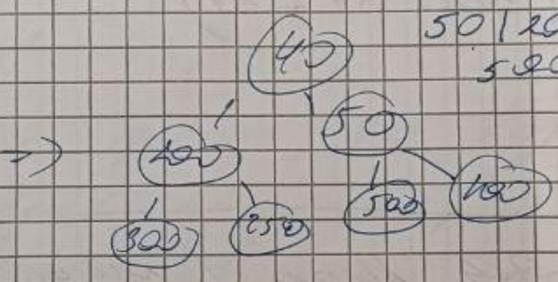
50|200|100|300|250|500|40



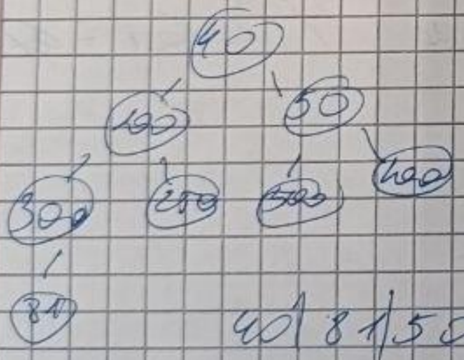
40 приоритетнее 50, меняем

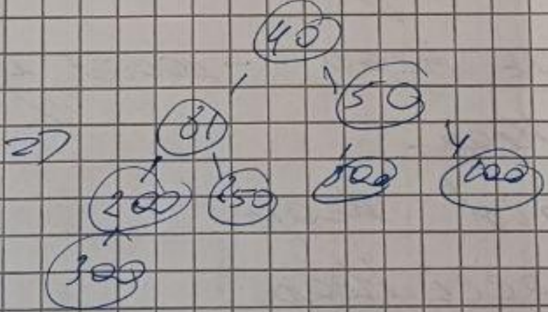
50|200|100|300|250|500|40





Добавим 81

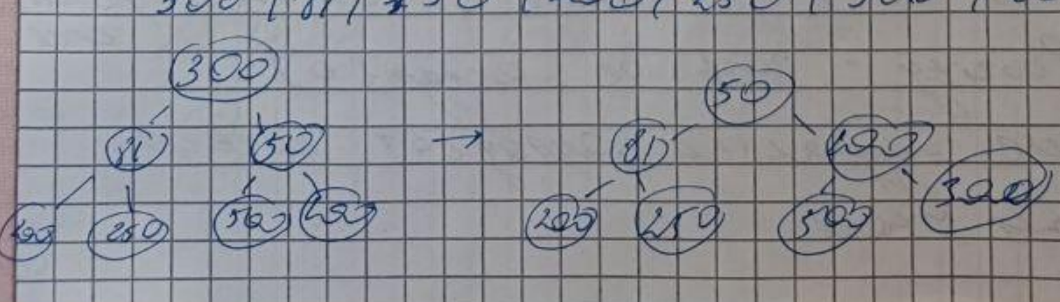




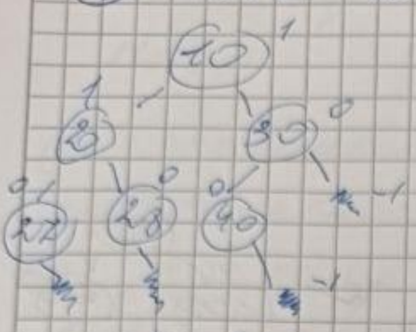
40|81|50|200|250|500|100|300

Удаляем всегда самый приоритетный и на его место ставим самый наименьший

300|81|50|200|250|500|100|~~300~~



② NPL - NULL PATH LENGTH



Св-ва:

① Чем узел выше, тем он более приоритетный

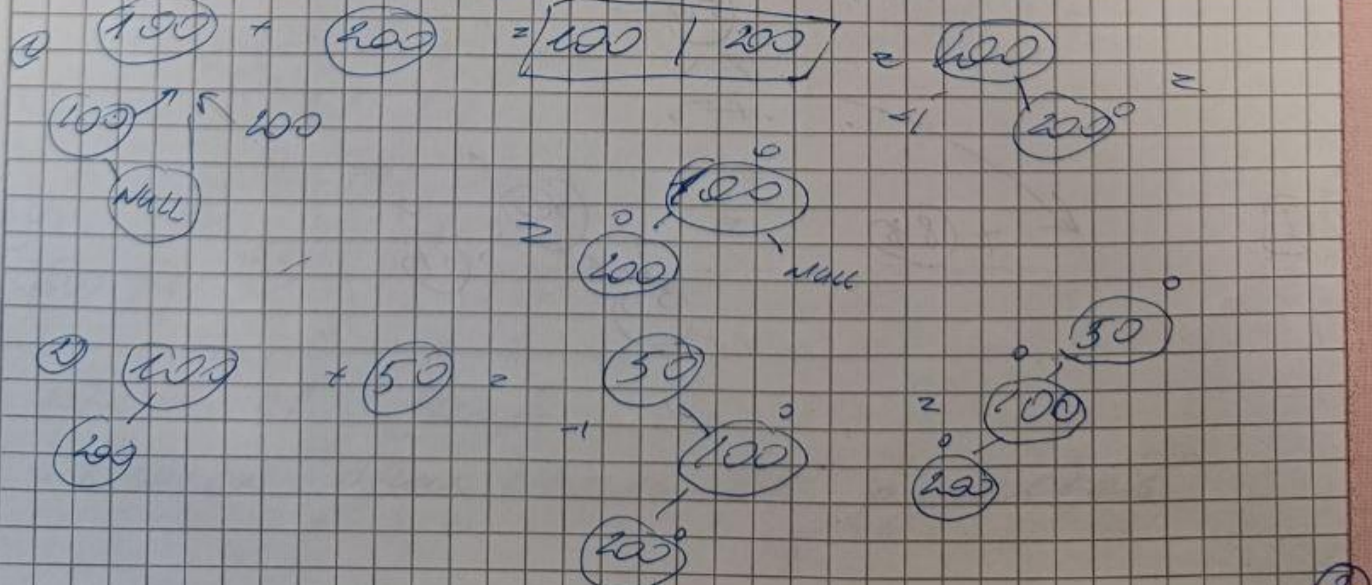
② NPL left child ≥ NPL right child

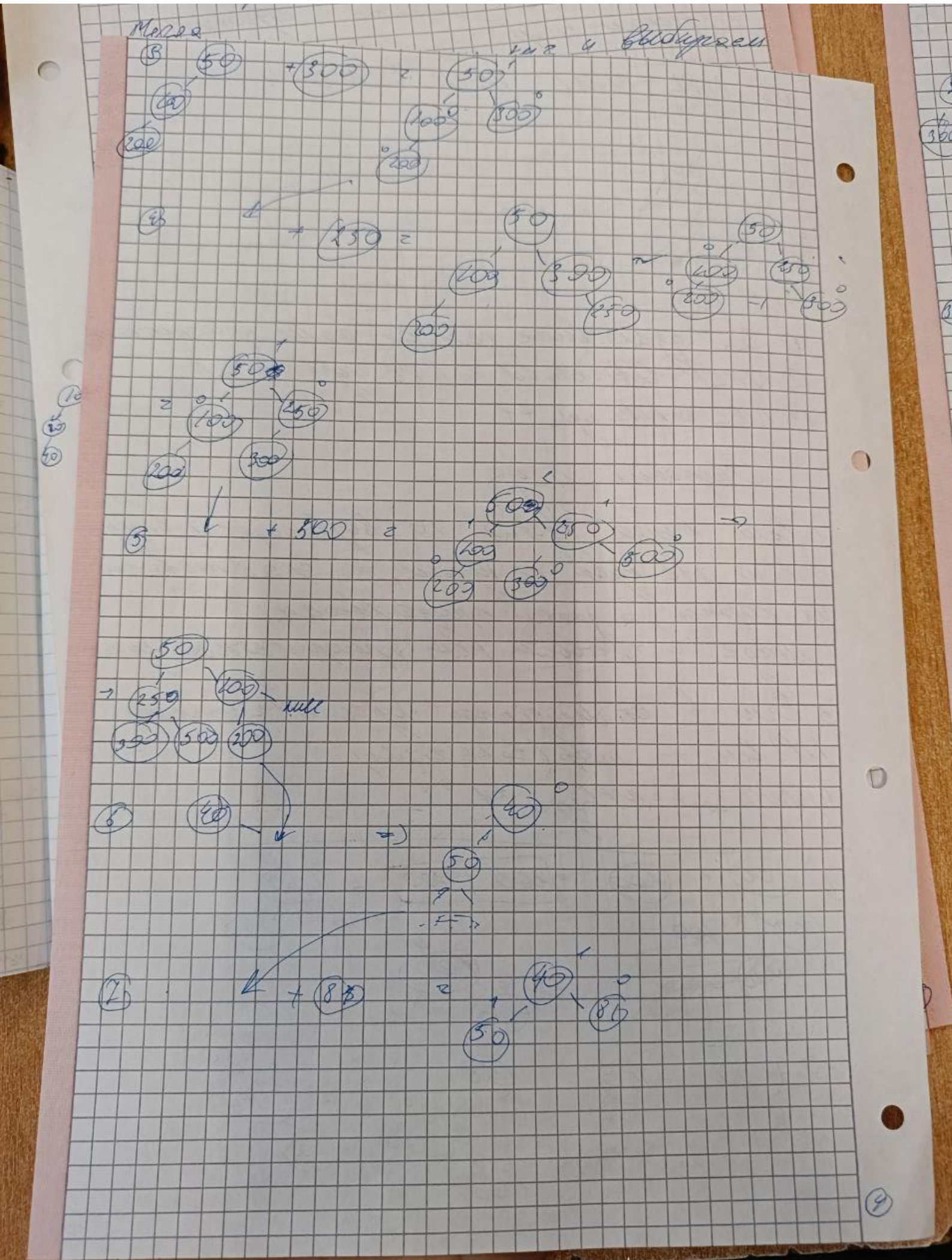
③ $NPL = NPL_{right} + 1$

Операции Merge

1. Если 1 куча пустая, то возвращаем вторую
2. Если обе кучи не пустые, то сравниваем их корни, а затем продолжаем Merge с правым п/д приоритетной кучи
3. Если в ст. Merge-нной куче $NPL_{right} > NPL_{left}$, то swap(left, right)

4) $NPL_{root} = NPL_{right} + 1$





Skew Heap

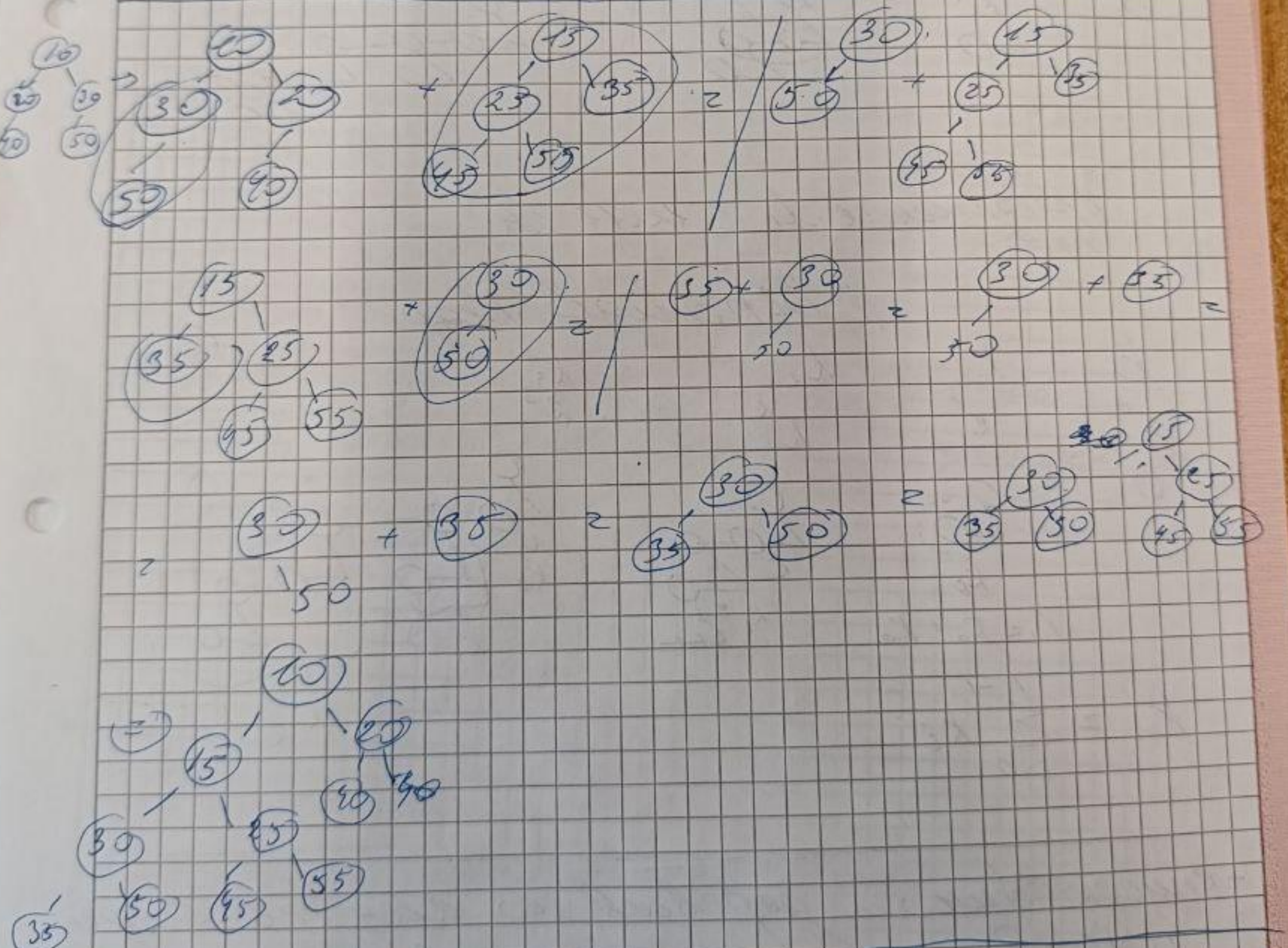
Merge

① Сравниваем корни куч и выбираем приоритетный

② => Swap куч таким образом, чтобы слева оказалась приоритетная (+)

③ Продолжаем то же самое для МПД левой кучи

④ внутри левой кучи swap (left, right)



Binomial Heap

- список биномиальных деревьев

(в-ва)

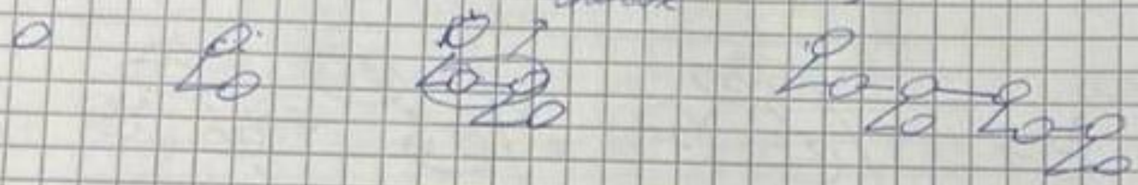
① Каждое биним. дерево строится с учетом сумм

② Нельзя встретить 2 одинаковых по порядку в биним. дереве

Биним. дерево квантового порядка

2^k эл-ов

K_0 K_1 K_2 список K_3

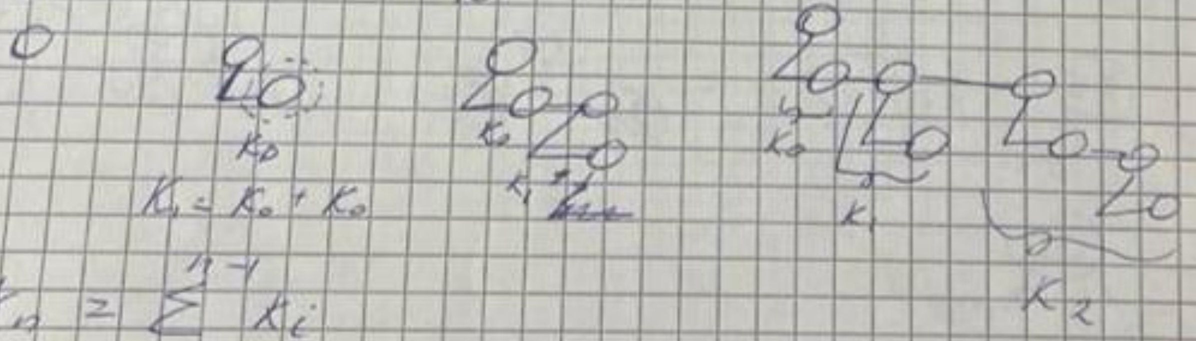


Упражнение лк. 12.11.24

Binomial Heap

K_0 K_1 K_2 K_3

2^0 2^1 2^2 2^3



$K_1 = K_0 + K_0$

$$K_n = \sum_{i=0}^{n-1} K_i$$

- список биним. деревьев со след св-вами

1) ∀ биним. дерево соответствует конкр. порядку

2) Мы не можем встретить в данном списке двух одинаковых по порядку биним. дерево ⑥

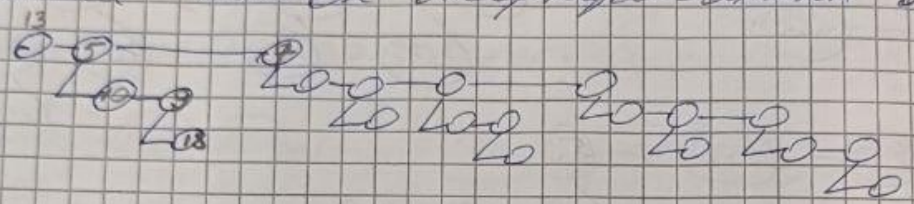
$K_0, K_1 \ldots K_n$ $K_0 - K_1 - K_2$

только по ним

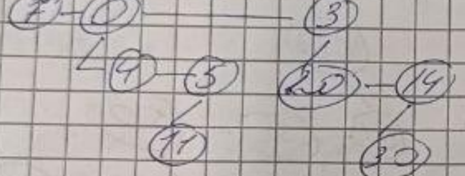
Правило для кучи:

1. Все биномиальные деревья находятся в отсортированном списке по степеням этих деревьев

2. Внутри каждого бином дерева тоже находится список отсортированных бином деревьев



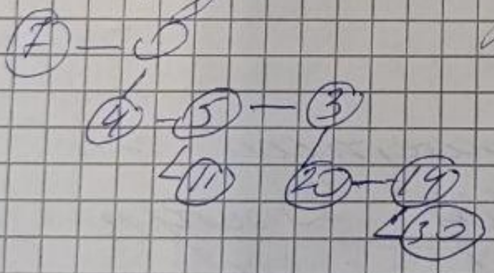
best priority



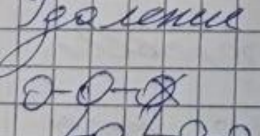
Должны перейти к другому списку чтобы не допустить одинаков. дерев одинаков.

Просто складываем два дерева сравниваем корни этих деревьев

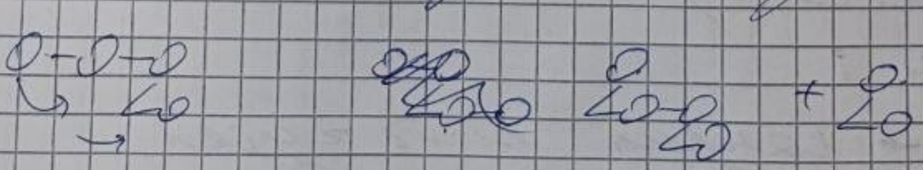
0 < 3 значит к дереву 0 присоед. дер. 3



Удаление 11.

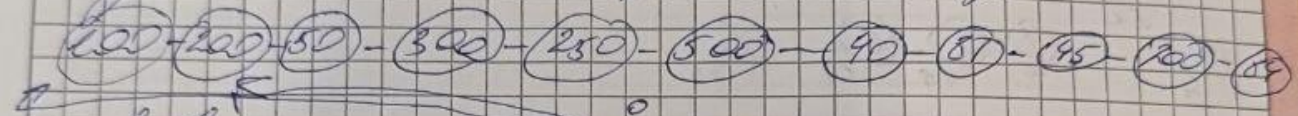


Добавляем каждое дерево из списка вершины



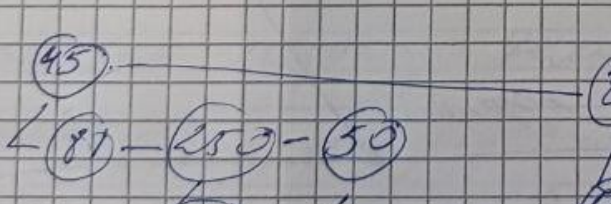
2. Best Priority → аналогичен Binomial

3. Удаление (Remove) - удаляем Heap самую приоритетную вершину, список её детей просто добавляем в основной список. После чего создаём массив порядков биномиальных деревьев и заполняем попарно указателями на деревья соответствующего порядка

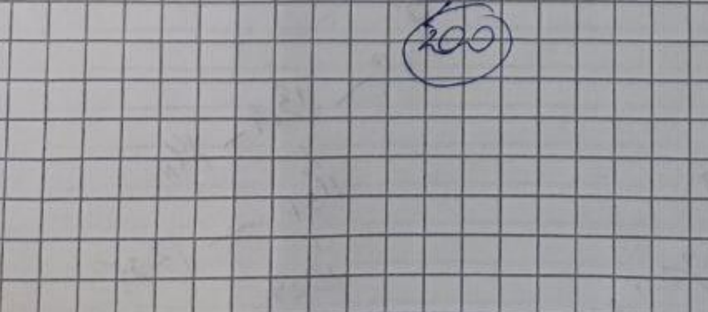


Заводим массив [0 | 1 | 2]
где индекс i позиция массива.

Когда сумма указателей станет равна 2, то делаем merge (100 ∠200)

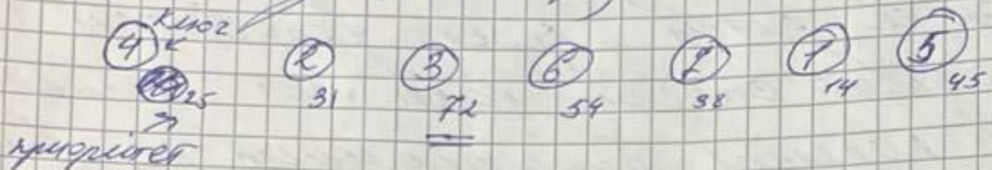


4 ГД.

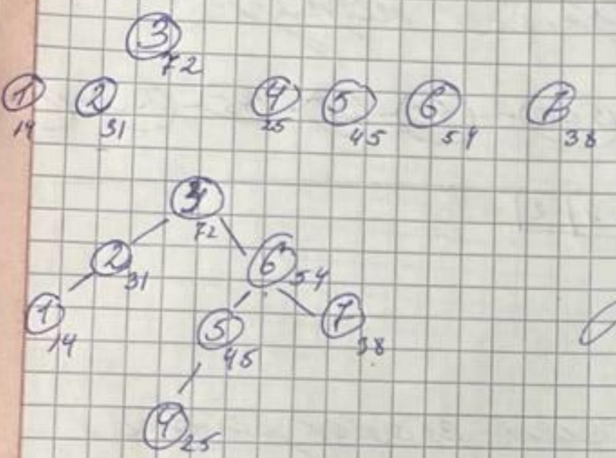


Пиво

декартово дерево (Декартач, Курево, Пиво, Дерамида, Theap)

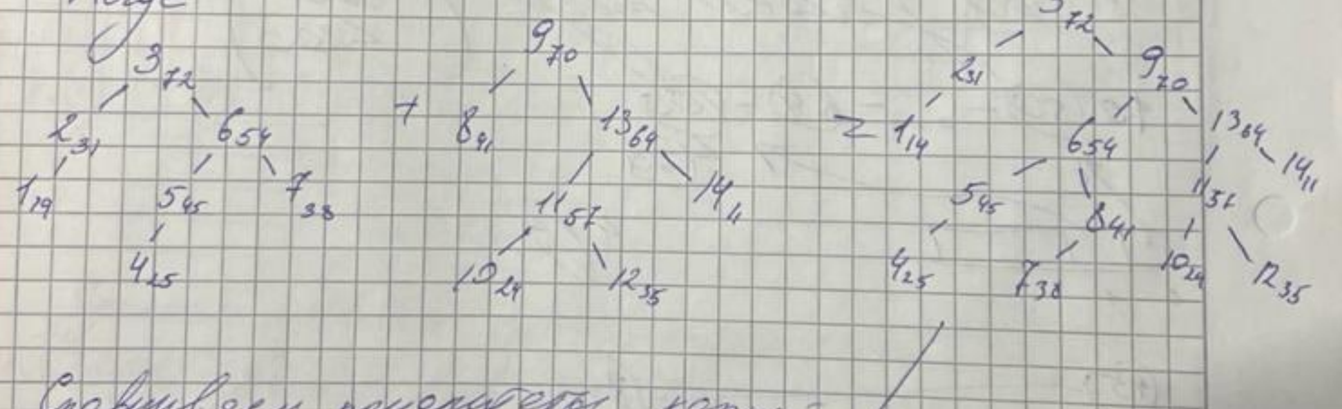


По ключам - BST priority - Heap



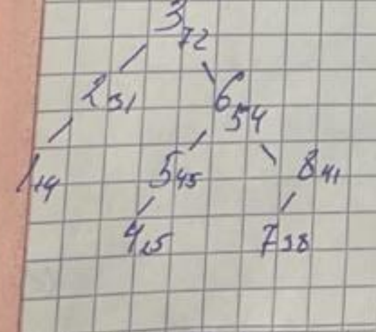
split и merge - только две операции

Merge

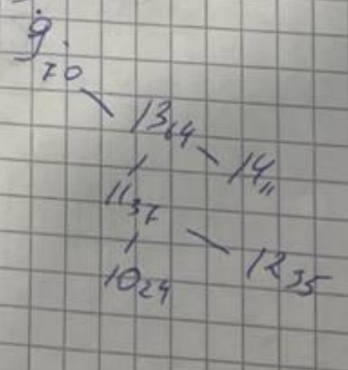


Сравниваем приоритеты корней

Split хотим на 7



7 можно докинуть куда угодно (левое дерево или в правое)



Хэш — отображение чего-то на число

Хэш функции:

Хэш таблицы

индексы в массиве, n=10

0 | [illegible] → ["a", "e"]
1 | [illegible]
2 | ... → ["x"]

"a"

0 % 10 → 5

"e" "e" % 10

"e" % 10 — возникает коллизия (проблема.)

Вероятность $\frac{1}{2}$

Изначально попала a, теперь туда идёт e

Если следующая клетка свободна, то поместить в пустую. Например через 9 проходимся. Идём 9→2, тогда e заменим в 2

в линейном разрешении коллизий

Если не нашли свободное место, то меняем место, делаем новую хэш таблицу

Методы устойчивости: хэш функции, хэш таблицы

Средний случай стремится к 1