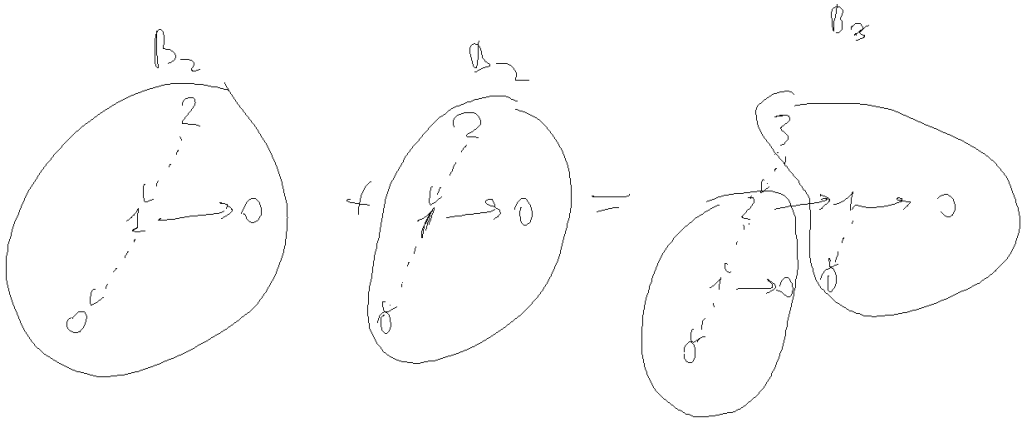
Куча: insert, extract_min, decrease_min, decrease_key, increase_key, merge. Хотим добавить операцию merge - объединить 2 кучи. Считаем, что можем делать операции амортизировано (не разделяем кучи и кучи не персистентные). Меньшую кучу будем добавлять в большую ("переливать"в большую).

Хотелось бы раздать каждой вершине по $\log n$ монет и говорить, что, когда мы "переливаем"кучу, все элементы этой кучи платят по монете. Тогда монет хватит, так как при каждом переливании размер кучи, где находится вершина увеличивается вдвое, значит каждая вершина перельется не более $\log n$ раз. Но все ломается из-за того, что мы можем удалять вершины. Можно ввести потенциалы (подумать), а можно сказать, что у каждого элемента есть ранг (r(i)) и при каждом переливании все ранги меньшей кучи увеличиваются на 1. Тогда амортизировано merge будет работать за $O(\log^2 n)$.

Биномиальная куча:

Вершин в биномиальной кучи 2^n, на k-ом слое C_n^k вершин. Из каждой вершины храним ребро в старшего сына, в предка и в следующего брата. $B_k = \text{merge}(B_{k-1}, B_{k-1})$. Заметим, что merge деревьев одного ранга работает за O(1).



Если у нас есть n чисел, то как мы их поместим в кучу размера 2^k? $n = 2^{k_1} + 2^{k_2} \cdots + 2^{k_m}$. Тогда можем хранить все элементы как кучи рангов $k_1, \ldots k_m$. Тогда при merge нужно объединять два списка биномиальных куч (будем делать это 2 указателями по 2 массивам), будем объединять кучи одинаковых рангов

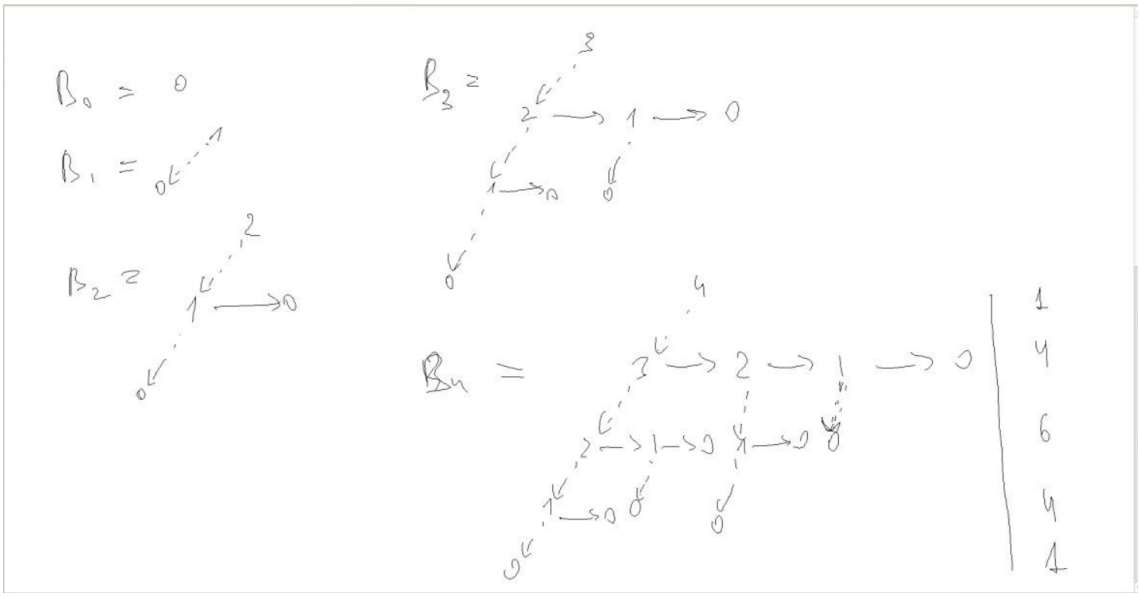
insert - создаем кучу ранга 0 с нашим элементом и делаем merge за $O(\log n)$.

Для поиска минимума будем просто поддерживать глобальный минимум, изменяя его за $O(\log n)$ (пробегаясь по всем кучам) при каждом запросе изменения.

decrease_key

extract_min

increase_key: decrease_key(v, $-\infty$) $\to$ extract_min $\to$ insert(x). Работает за $O(\log n)$.



Фибоначчиева куча: (чет я устал техать, чекайте у Кости)

Операции	binary heap	binomial heap	fibonacci heap
insert	$O(\log n)$	$O(\log n)$	$O(1)$
extract_min	$O(\log n)$	$O(\log n)$	$\tilde{O}(\log n)$
decrease_min	$O(\log n)$	$O(\log n)$	$\tilde{O}(1)$
increase_min	$O(\log n)$	$O(\log n)$	$\tilde{O}(\log n)$
merge	$\tilde{O}(\log^2 n)$	$O(\log n)$	$O(1)$
get_min	$O(1)$	$O(1)$	$O(1)$