Задача 1.

Решение. Создадим по два декартового дерева, для каждого множества, в котором, в каждой ноде будем хранить размер поддерева. В первом будем хранить отсортированные левые границы, во втором правые отсортированные границы отрезка. При втавке в какое-либо множество, будем считать количество отрезков которые точно не пересекаются критерий для этого: $r_1 < l_2$ и $l_1 > r_2$ где l_1, r_1 — отрезок первого множества, l_2, r_2 — отрезок второго множества. Соответственно, чтобы найти это количество будем искать в противоположном первом декартовом дереве правую границу точно непересекающихся в этом случае будет счетчик к которому мы прибавляли размер правого поддерева когда спускались в лево. Аналогично для левой границы только поменяем лево на право, право на лево и неравенство. И возьмем максимум из этих величин. Сответственно прибавим полученный результат к уже посчитанному на ранних дабавлениях и выведем ответ. Ассимптотика времени вставки и поиска меньше или равна: $O(\log S_1 + \log S_2)$ Затраченная память: $O(S_1 + S_2)$

Задача 2.

Решение. Создадим ДДНК отсортированный по значениям (то есть изначально при вставке будем искать индекс после которого нужно поставить элемент, чтобы получить отсорированный массив). Вставка и Удаление в ДДНК работает за $O(\log N)$. Чтобы обеспечить третью операцию за данную ассимптотику будем в каждой ноде хранить пять сумм которые получились путем сложения значений элементов с индексами, которые дают по модулю 5 $1, 2, 3, 4, 0$, закрепим за каждой этот модуль. Соответственно когда добавляем элемент в отсортированный массив или удаляем, будем пересчитывать уже посчитанные в нодах результаты путем того что найдем в каждой ноде по которой нужно пройти чтобы добраться до элемента и прибавим или отнимем от суммы, которая соответствует модулю который дает рассматриваемый индекс по пятерке, чтобы результат для всего массива был актуален пересчитаем и его, путем того, что по мере того как мы спускаемся до элемента для вставки или удаления, если мы должны спустится в лево отнимем от результата результат правого поддерева по модулю которого мы учитывали в результате и прибавим следующий по циклическому сдвигу(то же самое для удаления).

И того научились исполнять все операции за $O(\log N)$. Затраченная память $O(N)$.

Задача 3.

Решение. Создадим усовершенствованный Sparse Table. Соответственно Sparse Table позваляет нам за предпроцессинг $O(N \cdot \log N)$, отвечать на запросы о минимуме на подотрезке за $O(1)$. Чтобы поддерживать эту ассимптотику при вставке в конец массива добавим к каждому слою Sparse Table по элементу который будет являться результатом подсчета минимума от последнего элемента с предыдущего слоя и элемента с предыдущего слоя индекс которого, будет размер прудыдущего слоя - степень двойки, которой соответствует нынешний слой. В ходе такой вставки может получится новый верхний слой поэтому ассимптотика вставки аммотизированный $O(\log N)$ (слоев в Sparse Table $\log N$ округленный вниз, к каждому дабавляем элемент).

Задача 4.

Решение. Докажем что обработка данных операций не возможна за данную ассимптотику. Создадим массив с рандомными элементами длины N. Построим структуру в соответствии с нашим массивом. Возьмем минимум на всем отрезке запишем его в массив. Так как мы можем получить минимум можем утверждать, что мы знаем индекс или сам адресс из какого элемента этот минимум взялся заменим этот минимум

по индексу на $std :: numeric_limits < int64_t >:: max()$, продолжим эти операции пока реальных минимумов не останется. По итогу получили отсортированный масив за $O(N)$,и предподсчетом $o(N \cdot \log N)$ что невозможно по известным теоремам с доказательством противнго. ЧТД.