Бабушкин А.

1 Обязательные задачи

1. Возьмём первые два числа, сравним их, получим первых кандидатов для max и min. Затем будем брать по два числа, сравним их между собой, теперь большее из них не может быть min, а меньшее – max, то есть с помощью меньшего обновим min, большего – max. Всего сравнений $1 + 3(n - 1) = 3n - 2$.

2.
   a) За $O(m)$ строим кучу на первых элементах массивов, затем $k-1$ раз вытаскиваем наименьший элемент и двигаем указатель, затем просто берём минимальный из тех, что в куче.
   b) Делаем бинарь по ответу ($\log MAX$), затем m бинарей внутри массивов ($\log n$), проверяем, нашли мы больше k элементов или нет.

3. Переберём элементы из a слева направо, поддерживая указатели l и r такие, что $b[l] \leq a[i] \leq b[r]$, причём l будет максимально, а r – минимально. Тогда для a[i] ответ или b[l], или b[r].

4. **Old version:**
   Код
   New version:
   Код

5. **Old version:**
   Заведём две хеш-таблицы: для X- и Y-координат. В X[x] будут лежать отсортированные по y точки, у который x-координата = x. Тогда ответ на запрос это просто бинпоиск по вектору X[x] и Y[y].
   New version:
   Создадим второй массив из таких же точек. Один посортим по <X, Y>, второй по <Y, X>. Тогда точки с одним X будут лежать в первом рядом, причём по возрастанию Y. Для второго наоборот. Ответ на запрос – бинарь в соотв. массиве за $O(\log n)$, или можно для каждой точки сохранить в хеш-таблице её позицию в обоих массивах, тогда ответ на запрос за $O(1)$. Прекалк всегда за $O(n \log n)$, доппамяти ровно n в первом случае и $O(n)$ во втором.

6. **Old version:**
   Пусть $\Phi = -|b|$. Оценим add:
   a) $\sqrt{|a|} \geq |b| \to t_i = 1, \Delta\Phi = -1 \to a_i = 0$
   b) $\sqrt{|a|} < |b| \to t_i = |b| * \log n + |a|, \Delta\Phi = -|b| \to a_i = O(|a|)$
   Оценим count: $a_i = t_i + 0 \leq \log |a| + \sqrt{|a|} = O(\sqrt{n})$.
   Всё вместе работает за $O(add + n * count) = O(max(a_i) + \frac{|\Phi_0 - \Phi_n|}{n} + n\sqrt{n}) = O(n\sqrt{n})$.
   New version:
   Count очевидно работает за $O(\log n + \sqrt{n}) = O(\log n)$ безо всякого аморт. анализа, он неинтересный.

Теперь add. Пусть у нас есть сортировка, работающая за $Cn \log n$, и n – итоговое количество элементов. Тогда введём монетки: если перекидывания нет, то увеличим число монеток на $\sqrt{n} + C \log n + 1$. Если перекидывание есть, то будем монетки тратить. Пусть $|b| = x$. Sort потратит $Cx \log x$ монеток. Merge потратит $|a| + |b| \leq |b|(\sqrt{n} + 1) = x\sqrt{n} + x$ монеток. До этого мы хотя бы x раз сделали накопление монеток, то есть у нас на все эти операции монеток хватит. Тогда все операции add выполняются суммарно за O(сколько монеток) раз, а монеток $n(\sqrt{n} + \log n + 1) = O(n\sqrt{n})$ штук. Среднее время операции $add = O(\frac{n\sqrt{n}}{n}) = O(\sqrt{n})$.

2 Дополнительные задачи

1. Отсортируем коробки по w_i по убыванию.Возьмём первую, дальше будем рассматривать их по две. Пусть мы рассматриваем сейчас коробки $2i + 1, 2i + 2$. Возьмём ту из них, у которой больше b_i. Докажем, что так мы взяли хотя бы $\frac{B}{2}$ чёрных. В каждой паре мы брали ту коробку, где b_i больше, то есть сумма взятых $\geq$ суммы невзятых, то есть сумма взятых хотя бы половина от всех. Теперь докажем, что взяли хотя бы $\frac{A}{2}$ белых. В первой коробке белых не меньше, чем в невзятой из пары (2, 3). Во взятой из пары (2, 3) белых не меньше, чем в невзятой из пары (4, 5), и так далее. То есть опять сумма взятых не меньше суммы невзятых. Успех.

2. **Old version:**
   Сделаем бинпоиск по количеству "двойных" ящиков и проверим, правда ли можно выбрать K пар чисел так, что сумма в каждой из них $\leq W$. Очевидно, выгоднее всего взять префикс из $2K$ чисел в отсортированном массиве. Затем будем пытаться соединить 1-ый элемент с $2K$-ым, 2-ой с $2K - 1$-ым и т.д. Докажем, что если мы наша проверка провалилась, то разбиения такого префикса действительно нет. Пусть оно есть. Посмотрим, с каким элементом в паре $2K$-ый. Если не с 1-ым, то есть пары (1, i) и (j, 2K). Сделаем из них пары $(1, 2K)$ и (i, j). $a[1] \leq a[j], a[i] \leq a[2K] \to max(a[1] + a[i], a[j] + a[2K]) \geq max(a[1] + a[2K], a[i] + a[j])$, то есть мы можем сделать разбиение не хуже, если 1 и $2K$ будут стоять в паре. Аналогично для 2, 2K - 1 и т.д.Наше разбиение оптимально для этого K, поэтому если мы его смогли сделать, то всё хорошо, сдвигаем левую границу бинпоиска, иначе правую.
   New version:
   Посортируем ящики и будем идти указателем l слева, а r – справа. Если $a[l] + a[r] \leq W$, то $l++, r--$ и мы вместе их пихнули в один ящик. Иначе просто $r--$. Повторяем это, пока $l < r$. Работает по тем же соображениям, что работало решение выше: если $\exists$ оптимальное решение, где максимальный предмет лежит в ящике с кем-то ещё, то можно построить и оптимальное решение, где он лежит вместе с минимальным. Работает за $O(n + sort)$, то есть если нам дали их посорченными, то просто линия. $O(1)$ доппамяти.

3.