Матч тысячелетия

Вначале нужно поделить все значения p_i на НОД. Очевидно, что количеством валунов в i-й куче в итоге станет $x \cdot p_i$, поэтому можно перебрать этот x. Посмотрим, как меняется время подготовки i-й кучи при увеличении x на 1. Есть три варианта:

1. Если $(x+1) \cdot p_i \leqslant a_i$, то время уменьшится на p_i.
2. Если $x \cdot p_i \geqslant a_i$, то время увеличится на p_i.
3. В противном случае время увеличится на $((x+1) \cdot p_i - a_i) - (a_i - x \cdot p_i)$, что можно сократить до $p_i - 2 \cdot (a_i - x \cdot p_i) = p_i - 2 \cdot (a_i \bmod p_i)$

Это означает, что можно обновлять время сразу для всех куч суммарно, кроме переходных моментов в третьем случае. Обозначим эти моменты как x_i, тогда $x_i = a_i \operatorname{div} p_i$, и мы должны остановиться во всех значениях x из x_i и $x_i + 1$.

Отсортируем по значению x_i все кучи и запустим нечто вроде метода сканирующей прямой, чтобы перебрать все значения x по порядку, и для каждого узнать ответ. Будем поддерживать текущее число x, суммарное время T, и коэффициент d. Коэффициент d будет отвечать за изменение T при изменении x.

Начнём с $x = 0$, $T = \Sigma a_i$, и $d = -\Sigma p_i$. Это соответствует тому, что мы можем в каждой кучке собрать $0 \cdot p_i$ камней, на это уйдёт T времени, и при увеличении x наше время будет уменьшаться на Σp_i за каждую единицу сдвига x. Но это время подготовки не должно учитываться в ответе, потому что кучки должны быть непустые.

Затем будет постепенно увеличивать x, рассматривая только интересные нам точки x_i и $x_i + 1$. Найдя следующую точку $x_i > x$, присвоим $T = T + d \cdot (x_i - x)$. Затем нужно будет перейти в $x_i + 1$ и пересчитать T и d, так как у нас теперь камней в i-й куче станет больше текущего количества, а значит потраченное на эту кучу время будет расти. Значение d увеличится на $2p_i$, а T можно посчитать вычтя время для x_i и прибавив для $x_i + 1$, или по вышеуказанной формуле.

Докажем, что для ответа нам достаточно посмотреть T во всех x_i и $x_i + 1$. Пусть это не так, и оптимальный ответ T_{ans} был в $x_l + 1 < x_{ans} < x_r$. Но между $x_l + 1$ и x_r значение d не менялось — на протяжении всего интервала размер каждой кучки либо монотонно приближался к a_i, либо удалялся. Значит, величина T на этом интервале изменялась линейно, а линейная функция на отрезке всегда достигает максимума в одном из концов. А именно, при положительном d время $T_l < T_{ans}$, при отрицательном d — время $T_r < T_{ans}$, а при $d = 0$ — время $T_l = T_{ans} = T_r$.

В итоге, суммарно переход (третий вариант) сделается ровно один раз для каждой кучки, а поэтому сложность алгоритма равна $O(N \cdot \log N)$.

Альтернативное решение:

Аналогично предыдущему решению, поделим все p_i на их НОД, и будем искать лучшее значение x. Пусть $f_i(x) = |a_i - p_i \cdot x|$ — время, нужное чтобы привести i-ю кучу в порядок при заданном коэффициенте x. Заметим, что все функции f_i битонные; то есть, до $x = \frac{a_i}{p_i}$ они строго убывают, затем строго возрастают. Ответ равен сумме $\Sigma f_i(x)$. Можно показать, что эта сумма является выпуклой функцией от x. А значит, можно найти минимум с помощью тернарного поиска.