Beetle

Spoiler

We will assume that $x_1 \leq x_2 \leq \ldots \leq x_s \leq \ldots \leq x_n$, this can be done in $O(n \log n)$. Here $x_s = 0$ is an extra added "start" drop we will find useful, and n is the total number of drops with this new drop included.

Suppose that the beetle drinks k drops at time moments $t_1, t_2, \ldots, t_k$ respectively. The total amount of water drunk will be $(m - t_1) + (m - t_2) + \ldots + (m - t_k) = km - (t_1 + t_2 + \ldots + t_k)$. Since km is fixed for fixed k, we are to minimize the sum $t_1 + t_2 + \ldots + t_k$. Actually, this formula only holds if $t_1, t_2, \ldots, t_k \leq m$, because the beetle does not really get any "penalty points" for passing a drop that is no more. However, letting these penalty points will not change the maximal possible amount, as there is always an optimal route in which the beetle stops before $t > m$ (and it is clever to do so!).

Also notice that the next drop the beetle will drink is always either the closest from left or from right, because there's no point in not drinking a drop if one is passing it anyway.

From here on we use dynamic programming. Let $L(i, j, k)$ be the least possible "penalty sum" beetle would get if at time moment $t = 0$ it started at x_i and would drink exactly k drops, but there were no drops $i, i + 1, \ldots, j$. Similarly, let $R(i, j, k)$ be the least possible "penalty sum" for drinking k drops starting at x_j if there were no drops $i, i + 1, \ldots, j$.

If the beetle willing to drink k drops starts at x_i and there are no drops $i, i+1, \ldots, j$, then it can either go for drop at x_{i-1} or x_{j+1}, which reduces the problem to either $L(i-1, j, k-1)$ or $R(i, j+1, k-1)$. The penalty (time spent) for current drop will be either $x_i - x_{i-1}$ or $x_{j+1} - x_i$. In any case, this penalty will count in for all other $k - 1$ drops the beetle is going to drink. Therefore the following equations hold:

$$L(i, j, k) = \min\{L(i - 1, j, k - 1) + k(x_i - x_{i-1}), R(i, j + 1, k - 1) + k(x_{j+1} - x_i)\}$$
$$R(i, j, k) = \min\{L(i - 1, j, k - 1) + k(x_j - x_{i-1}), R(i, j + 1, k - 1) + k(x_{j+1} - x_j)\}$$

The boundary conditions are:

$$L(i, j, 0) = R(i, j, 0) = 0, \qquad 1 \leq i \leq j \leq n$$

Now we can check what is the maximal amount of water the beetle can drink if starts at time moment $t = 0$ at x_s and there is no drop s:

$$\max\{km - L(s, s, k) : 0 \leq k < n\}.$$

We find it by consequently calculating L and R values in $O(n^3)$ time and $O(n^2)$ memory.

Test data overview

The following sample solutions were tested:

A Correct solution.

B Memory $O(n^3)$, otherwise correct.

C Wrong solution by dynamic programming. Implicitly assumes that most is always drunk in least time.

D Looking for the best time instead of the best sum of times.

E Greedy solution.

F Recursion.

G Less efficient recursion.

H Does not sort the drops, otherwise correct.

I Assuming that all drops are to be drunk, otherwise correct.

Test #	**Group #**	**Points**	n **up to**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
1	1	3	3	+	+	+	+	+	+	+	+	+
2,3,4	2	3	10	+	+	+	+	+	+	+	+	–
5,6,7	3	5	20	+	+	+	+	+	+	+	–	+
8,9	4	5	25	+	+	+	+	–	+	+	–	+
10,11,12,13	5	5	25	+	+	+	–	–	+	+	–	–
14,15,16,17,18	6	10	25	+	+	–	–	–	+	+	–	–
19,20	7	7	60	+	+	–	–	–	+	–	–	–
21,22	8	5	100	+	+	+	+	–	+	–	–	–
23,24,25	9	5	100	+	+	+	–	–	–	–	–	+
26	10	5	100	+	+	–	+	+	–	–	+	+
27,28,29,30	11	15	100	+	+	–	–	–	–	–	–	–
31,32,33,34	12	10	200	+	–	–	–	–	–	–	–	–
35,36,37	13	10	250	+	–	–	–	–	–	–	–	–
38,39,40	14	10	300	+	–	–	–	–	–	–	–	–
41,42,43,44,45	15	2	100	+	+	+	+	+	+	–	–	+
Total				**100**	**70**	**38**	**28**	**18**	**45**	**31**	**11**	**25**

Biggest tests were generated randomly. The last group contains test cases where the answer is zero, and in one of them $n = 0$.