1. perform $O(\log(nW))$ binary searches on the result value, each check takes $O(n)$ by greedy.
we can also reduce the number of binary searches to $O(\log n)$: (there may be a more generalized reference) there are $O(n^2)$ possible values for the result (sum of interval). if each time we sample a value from the currently remaining possibilities uniformly at random and check that value, the algorithm will terminate in $O(\log n)$ rounds in expectation. we can use $O(n)$ to sample, by determining the current range by two pointers.
now we reduce the total running time for the binary search part to $O(n)$. after determining the current range in $O(n)$, we can sample in $O(\log n)$. use this sample range for the next $\Theta(\log n)$ binary searches, the actual sampling space will shrink, use rejection sampling to repeatedly sample until we get a sample in the actual sample space. if we fail $O(n^\epsilon)$ times ($\epsilon < 1$ is a constant), rebuild the sample space. we need to rebuild $O(1)$ times in expectation.
we can also use the algorithm for selecting the k-th largest element in a sorted matrix in both row and column in deterministic $O(n)$ (but the binary search part in total need $O(n \log n)$). see 378. Kth Smallest Element in a Sorted Matrix.
the running time is $O(n \log n)$ (independent of m).
2. perform each check by exponential search in $O(m \log \frac{n}{m})$. the running time is $O(n + m \log \frac{n}{m} \log n)$.
3. $O(n)$ [1, 2].

References

[1] Greg N Frederickson. Optimal algorithms for tree partitioning. In *SODA*, volume 91, pages 168–177, 1991.

[2] Greg N Frederickson and Samson Zhou. Optimal parametric search for path and tree partitioning. *arXiv preprint arXiv:1711.00599*, 2017.