Big Persimmon

To begin with, let's notice two useful things:

- Maximizing the sum of taken elements is the same as maximizing the difference between what you take and what your opponent takes.
- As long as there is at least one piece left on the table, both people have eaten the same amount, so the difference between them is zero.

To start solving the problem in $O(n^2 \cdot \max_i(w_i))$, we will use dynamic programming:

Let's denote dp[l][r][dif] as the difference between the amount the first person takes and the amount the second person takes, if they were to play on the segment of elements [l, r], with the first person starting to eat after dif seconds from the second person (where dif can be negative).

The base case for the dynamic programming is: dp[i][i-1][dif] = 0, and the answer lies in dp[0][n-1][0].

The sign of dif determines who will make the decision about which piece to take first. Thus, the transitions are as follows:

For dif ≤ 0: dp[l][r][dif] = max of

- $dp[l+1][r][dif + w[l]] + w[l]$
- $dp[l][r-1][dif + w[r]] + w[r]$

For dif > 0: dp[l][r][dif] = min of

- $dp[l+1][r][dif - w[l]] - w[l]$
- $dp[l][r-1][dif - w[r]] - w[r]$

In this dynamic programming, $|\text{dif}| \leq \max_i(w_i)$, so there are a total of $O(n^2 \cdot \max_i(w_i))$ states and two transitions from each.

To solve subgroups with the restriction $w_{i+1} \leq 2 \cdot w_i$, we will prove the following fact: in such constraints, for any achievable state (l, r, dif), it holds that $|\text{dif}| \leq w_{r+1}$ (exception: if $r = n-1$, then $|\text{dif}| \leq w_{n-2}$).

To prove this, note that in any transition, dif changes towards zero, so its absolute value either decreases or becomes no more than w_i (where w_i is the last taken piece). Thus, in transitions [l, r] $\rightarrow$ [l+1, r], the relationship is preserved in any case, and in transitions [l, r] $\rightarrow$ [l, r-1], it is preserved because $w_{r+1} \leq 2 \cdot w_r$.

Therefore, for each l, there are only $O(\sum_{i=0}^{n-1} w_i) = O(W)$ achievable pairs r, dif, so there are a total of $O(n \cdot W)$ states and two transitions from each.

Now, to solve the full problem, we will change the transitions in the dynamic programming to preserve the relationship $|\text{dif}| \leq w_{r+1}$.

To do this, note that if one person took several pieces before passing the turn to the opponent, they could always do so by first taking the smallest pieces and then the largest ones. Therefore, we will keep the transitions [l, r] $\rightarrow$ [l+1, r] as they preserve the invariant, and instead of transitions [l, r] $\rightarrow$ [l, r-1], we will add transitions [l, r, dif] $\rightarrow$ [l, r', dif'], meaning that the person takes the largest elements so that the turn passes to the opponent, or the game ends.

These transitions preserve the invariant $|\text{dif}| \leq w_{r+1}$, so in this dynamic programming, there are $O(n \cdot W)$ states and still two transitions from each.

Also note that the values of r' depend only on r and dif, but not on l, so they can be computed in $O(W \cdot \log(n))$.

The resulting solution works in $O(n \cdot W)$, which is sufficient to pass all tests.