Kick Start 2019 - Round B

Analysis: Energy Stones

Test set 1 (Visible)

For this test set, it is guaranteed that $\mathbf{S}_i = \mathbf{S}_j$ for all i, j. For simplicity, we will assume that we never eat a stone with zero energy. Consider two energy stones i and j that will be eaten back-to-back. If $\mathbf{L}_i > \mathbf{L}_j$ then we should eat i before j. This is because stone i loses energy faster than j, so taking it first will result in a smaller overall loss of energy.

Thus, no matter which set of energy stones are eaten, that set should be eaten in non-increasing value of $\mathbf{L}_i$. So we should first sort the stones by $\mathbf{L}_i$ and then the only decision to be made is which stones should be eaten and which should not be eaten. This reduces the problem to a 0/1 Knapsack problem. This can be solved with dynamic programming.

Define `max_energy(time, i)` as the maximum total energy that can be achieved given the current time and considering only the suffix of energy stones sorted in decreasing $\mathbf{L}_i$ from i to $\mathbf{N}$. The recurrence relation for this function considers two cases. Either take the i-th energy stone (with its energy adjusted by the time), or do not take it. So, `max_energy(time, i)` is the maximum of:

- max_energy(time+$\mathbf{S}_i$, i+1) +max(0,$\mathbf{E}_i$-$\mathbf{L}_i$time)
- max_energy(time,i+1)

The maximum possible time is the sum of all $\mathbf{S}_i$ because an optimal strategy might eat all the stones and will not use any time waiting. Call this sum($\mathbf{S}$). The time complexity of this approach can be described as O($\mathbf{N}$ × sum($\mathbf{S}$)). This is fast enough for both test sets. However, sorting energy stones by $\mathbf{L}_i$ is incorrect for Test set 2.

Test set 2 (Hidden)

We will need to find a different way to order the energy stones to solve Test set 2. As before, consider two energy stones i and j assuming that we can take both i and j without either going to zero energy. We know that $\mathbf{S}_i$ might not equal $\mathbf{S}_j$. However, there is an ordering for taking both i and j that is always optimal. Observe that $\mathbf{S}_i\mathbf{L}_j$ is the total loss of energy if i is used first. Likewise, $\mathbf{S}_j\mathbf{L}_i$ is the loss if j is used first. Thus, if $\mathbf{S}_i\mathbf{L}_j < \mathbf{S}_j\mathbf{L}_i$ then taking i first leads to a smaller overall loss of energy. It may not be obvious that we should always take i before j even if it leads to a smaller loss of energy. This is because there may be other stones between i and j in some potential ordering. However, if i and j are adjacent in some ordering, then we will achieve more energy by swapping them if $\mathbf{S}_i\mathbf{L}_j > \mathbf{S}_j\mathbf{L}_i$. Applying this rule iteratively will eventually sort the stones. Therefore, this rule defines an ordering on our energy stones.

Formally, suppose for a contradiction, we have an optimal solution that eats X stones in the order P_1, P_2, ..., P_X, where each stone gives Duda a positive amount of energy, but there exists an i such that $\mathbf{S}_{P_i}\mathbf{L}_{P_{i+1}} > \mathbf{S}_{P_{i+1}}\mathbf{L}_{P_i}$. If we swap the order we eat these two stones, we gain exactly $\mathbf{S}_{P_i}\mathbf{L}_{P_{i+1}}$ more energy and lose at most $\mathbf{S}_{P_{i+1}}\mathbf{L}_{P_i}$ (we may lose less than that, if the stone's energy drops to zero).

Since we assumed that $\mathbf{S}_{P_i}\mathbf{L}_{P_{i+1}} > \mathbf{S}_{P_{i+1}}\mathbf{L}_{P_i}$, this would increase the amount of energy Duda gains, which contradicts the assumption that this is an optimal solution.

Thus, we can use the dynamic programming solution from Test set 1 to solve Test set 2 with the same time complexity.

The reader may have noticed that this sort order is equivalent to comparing fractions; it is the same as sorting by $\mathbf{S}_i/\mathbf{L}_i$. However, one must be careful when $\mathbf{L}_i$=0.