Code Jam 2022 - Round 1B

Analysis: Pancake Deque

Test Set 1

For the first test set, we could try a brute force solution for this problem. Given a remaining deque of pancakes $\mathbf{D}$, we could choose to serve the next customer from either the front or back of the deque. Meanwhile, we could update the maximum deliciousness of the pancakes served every time we decided to serve a pancake out. We could use a recursive brute force method to simulate the process.

Given that every time we have two choices for serving the pancake (either the first or the last one) , the overall time complexity of the approach will be $O(2^{\mathbf{N}})$, which is sufficient to solve the first test set.

Notice that the brute force method can be improved by using memoization to record the current optimal solution for a partial deque. There are $\binom{\mathbf{N}}{2} + \mathbf{N} = \frac{\mathbf{N}\cdot(\mathbf{N}+1)}{2}$ partial deques in total to take into consideration. Therefore, this will reduce the time complexity down to $O(\mathbf{N}^2)$.

Test Set 2

The same approach cannot be applied to the second test set, since it will result in a time limit exceed. Therefore we need a more clever way other than trying to serve the pancakes brute-force. We could quickly make an observation that, if the deliciousness of the pancakes we could serve at some point are $\mathbf{D}_{\text{left}}$ and $\mathbf{D}_{\text{right}}$, it will always be better to serve the one with less deliciousness. To prove this, we could do a quick analysis. For simplicity, let us assume that $\mathbf{D}_{\text{left}} \le \mathbf{D}_{\text{right}}$, and denote $\mathbf{D}_{\text{max}}$ as the greatest deliciousness of the pancakes served out so far. If $\mathbf{D}_{\text{left}} < \mathbf{D}_{\text{max}}$, this means that the pancake will be served out free no matter when we serve it, so we could easily serve it out now and will not affect our final answer. Otherwise, since $\mathbf{D}_{\text{left}} \le \mathbf{D}_{\text{right}}$ it will always be better to serve $\mathbf{D}_{\text{left}}$ first, or else $\mathbf{D}_{\text{max}}$ will be updated to at least $\mathbf{D}_{\text{right}}$. In that case $\mathbf{D}_{\text{left}}$ will be served out free. Therefore we could summarize into a criteria for serving pancakes: serve out $\min(\mathbf{D}_{\text{left}}, \mathbf{D}_{\text{right}})$, and update $\mathbf{D}_{\text{max}}$ if needed. For each customer, this criteria takes $O(1)$ time to perform, therefore the total time complexity of this approach is $O(\mathbf{N})$.