Homework 8, additional problems, solution set

1. $\mathrm{Lgl}(j)$ evaluates to True if $s1s_2\dots s_j$ forms a sequence of known words and to False otherwise. It is given by the following recursive expression.

$$\mathrm{Lgl}(j) = D(s,1,j) \bigvee_{1\le i<j} \big[\mathrm{Lgl}(i) \wedge D(s,i+1,j)\big].$$

This is correct because the recursion considers for each i whether $s_{i+1}\dots s_j$ is a known word and if so recurses on the remaining initial portion of the string, if any.

b. There are n possible recursive calls. In an efficient implementation, $\mathrm{Lgl}(j)$ performs $O(j)$ non-recursive work, yielding a total $O(n^2)$ runtime. We set $i = 0$ to correspond to the choice $D(s,1,j) = \mathrm{True}$.

c. For each recursive call we need to record which choice of the index i, if any, yields an outcome of True.

2.a. Let $\mathrm{Win}(i,k)$ output the maximum value of the first player's winnings when the remaining cards are $C_i, C_{i+1}, \dots, C_k$ and the second player plays optimally; note that this value could be negative. It is helpful to compute $\mathrm{Tot}_{ik} = \sum_{j=i}^{k} v_j$ for $1 \le i \le k \le n$. Note that if we compute $\mathrm{Init}[k] = \sum_{j=1}^{k} v_j$ for $0 \le k \le n$, then we have $\mathrm{Tot}_{ik} = \mathrm{Init}[k] - \mathrm{Init}[i-1]$ in $O(1)$ additional time.

$\mathrm{Win}(i,k)$ is given by the following recurrence.

$$\mathrm{Win}(i,j) = \begin{cases} \max\{\mathrm{Tot}_{ik} - 2\cdot \mathrm{Win}(i+1,k), \mathrm{Tot}_{ik} - 2\cdot \mathrm{Win}(i,k-1)\} & i > k \\ v_i & i = k \end{cases}$$

The reason this is correct is that if the first player takes C_i, the second player can obtain $\mathrm{Win}(i+1,k)$ but no more from the remaining cards, and if the first player takes C_k the second player can obtain $\mathrm{Win}(i,k-1)$. The first player obtains the remainder of Tot_{ik}, i.e the better of $\mathrm{Tot}_{ik} - \mathrm{Win}(i+1,k)$ and $\mathrm{Tot}_{ik} - \mathrm{Win}(i,k-1)$. So the difference in the winnings is as given in the recurrence.

b. As $1 \le i \le k \le n$, there are $\frac{1}{2}n(n-1) = O(n^2)$ possible recursive calls. In an efficient implementation, each recursive call performs $O(1)$ non-recursive work, yielding an overall $O(n^2)$ runtime.

c. For each recursive call we need to record which choice is the better play, breaking ties arbitrarily.