Problem Set 7

Name: Gabriel Chiong

Collaborators: None

Problem 7-1.

Subproblems:

- Let $x(i)$ denote the maximum delegates that Torr can win in state i for all $i \in \{1, \ldots, n+1\}$.

Relate:

- Torr can either choose to campaign or not on any day.
- If she chooses to campaign on day i, then $x_c(i) = d_i + z_{i+1} + z_{i+2} + x(i+3)$.
- If she chooses to do nothing on day i, then $x_n(i) = z_i + x(i+1)$.
- Therefore the overall relation can be written as $x(i) = \max\{x_c(i), x_n(i)\}$.

Topological order:

- $x(i)$ only depends on subproblems with strictly larger i, therefore the relation graph is a DAG.

Base:

- For $i > n$, we have $x(i) = 0$.
- For $i = n$, we have $x(i) = d_n$.
- For $i = n - 1$, we have $x(i) = \max\{d_i + z_n, z_i + d_n\}$.

Original:

- Calculate $x(1)$ and check if $x(1) > \lfloor D/2 \rfloor + 1$.

Time:

- There are $n + 1$ subproblems including the base case, and constant $O(1)$ work done per sub-problem. Therefore, the overall running time is $O(n)$.

Problem 7-2. We first sort the tigers by age using an $O(n \log n)$ algorithm like merge sort. Next, sort the cages by distance in increasing order in $O(n^2 \log n)$ time, again using merge sort.

Subproblems:

- Define $x(i, j)$ be the minimum total discomfort for tigers `T[i:]` in cages `C[j:]` for all $i \in \{0, \ldots, n\}$ and $j \in \{0, \ldots, n^2\}$.

Relate:

- Let the discomfort $d(i, j)$ of tiger i in cage j be $s_i - c_j$ if $s_i > c_j$ and 0 otherwise.
- We can choose to match every i, j or not, therefore we have $x(i, j) = \min\{d(i, j) + x(i + 1, j + 1), x(i, j + 1)\}$.

Topological order:

- $x(i, j)$ at any stage relies on strictly larger subproblem js, therefore the relation graph is a DAG.

Base:

- $x(n, j) = 0$, since there are no more tigers are left to match, there can be no more additional discomfort.
- $x(i, n^2) = \infty$, to penalize the scenario of not matching all tigers to cages.

Original:

- $x(0, 0)$ to calculate the discomfort considering all tigers and all cages.
- We can store parent pointers to reconstruct the optimal assignment.

Time:

- There are $(n + 1)(n^2 + 1) = O(n^3)$ subproblems.
- Each subproblem takes a constant $O(1)$ amount of work.
- Therefore the overall algorithm runs in $O(n^3)$ time as required.

Problem 7-3.

Subproblems:

- We define $i \in \{0, 1\}$ as an indicator of the parity of the number of even or odd paths.
- Let $x(v, i)$ be the number of even paths from (s, v) if $i = 0$ and the number of odd paths is $i = 1$ for all $v \in V$.
- Therefore, there are effectively two vertices for every vertex in the original graph G.

Relate:

- Let $\chi(k)$ be a function with returns 1 if k is even, and 0 otherwise.
- Then we can define $x(v, i) = \sum\{x(u, \chi(w(u, v) + i)) : u \in Adj^{-}(v)\}$.
- u is the set of all incoming vertices to v.

Topological order:

- $x(v, i)$ depends only on $x(u, j)$ where u appears before v int he topological order of G.

Base:

- $x(s, 0) = 1$, since 0 is an even number.
- $x(s, 1) = 0$, for the case where there are no odd length paths from (s, s).
- $x(v, 0) = x(v, 1) = 0$ for all $v : Adj^{-}(v) = \emptyset$, where there are no incoming vertices to v.

Original:

- $x(t, 1)$ calculates the odd paths in reverse topological order.

Time:

- There are $2|V|$ subproblems (a vertex for each odd/even pair).
- Each subproblem requires iterating over all incoming vertices, which is $O(\deg^{-}(v))$.
- Therefore, the overall runtime is $O(2\sum_{v \in V} \deg^{-}(v)) = O(|V| + |E|)$, which is linear in the input graph size.

Problem 7-4. As the game progresses, the slices eaten by round are cyclically consecutive (since the slices are eaten in α_i to $\alpha_i+\pi$). Therefore our subproblems can be consecutive cyclic subarrays. Since each sibling wishes to maximize tastiness, our subproblems require an indicator for which sibling is currently making the choice.

Let $v(i,j)$ be the tastiness of the j slices counter-clockwise from angle α_i, specifically $v(i,j) = \sum_{k=0}^{j-1} t_{((i+k) \pmod{2n})}$, where $i \in \{0,\ldots,2n-1\}$ and $j \in \{0,\ldots,n\}$. There are $O(n^2)$ ways to construct a $v(i,j)$, and each can be computed in $O(n)$ time for an overall time of $O(n^3)$.

Subproblems:

- Let $x(i,j,p)$ be the maximum tastiness Liza can achieve with the j slices counter-clockwise from α_i remaining, when either Liza is the chooser ($p = 1$), or Lie is the chooser ($p = 2$).
- Also let i, j, p be defined as $i \in \{0,\ldots,2n-1\}$, $j \in \{0,\ldots,n\}$, and $p \in \{1,2\}$.

Relate:

- Chooser can choose any proper angle and then choose a side. Therefore, we consider all possibilities.
- Angle α_{i+k} is proper for any $k \in \{1,\ldots,j-1\}$.
- Chooser eats either k slices between α_i and α_{i+k} or $j-k$ slices between α_{i+k} and α_{i+j}.
- Neither gains or loses tastiness from the choice of the other.
- For Liza, $x(i,j,1) = \max\{\max\{v(i,k) + x(i+k, j-k, 2), v(i+k, j-k) + x(i,k,2)\} : k \in \{1,\ldots,j-1\}\}$.
- For Lie, $x(i,j,2) = \min\{\min\{x(i+k, j-k, 1), x(i,k,1)\} : k \in \{1,\ldots,j-1\}\}$.

Topological order:

- $x(i,j,p)$ only depends on subproblems with strictly smaller j, therefore the relation graph is a DAG.

Base:

- Liza eats the last slice, $x(i,1,1) = t_i$, for all $i \in \{1,\ldots,2n\}$.
- Lie eats the last slice, $x(i,1,2) = 0$, for all $i \in \{1,\ldots,2n\}$.

Original

- Liza starts the game, and the maximum tastiness on a half and letting Lie choose on the other half.
- $\max\{x(i,n,2) + v(((i+n) \pmod{2n}), n) : i \in \{0,\ldots,2n-1\}\}$

Time:

- There are $2(2n)(n+1) = O(n^2)$ subproblems in total, each requiring $O(n)$ work per subproblem. The original also requires $O(n)$ work, so the overall running time is $O(n^3)$.

Problem 7-5.

Subproblems:

- Let $x(j)$ be the longest decreasing subsequence of prices that doesn't skip days in `P[j:` that includes price `P[j]`, for all $j \in \{0, \ldots, nk-1\}$.

Relate:

- Next stock price information in the sequence is at a later time in the same day or the next, so we iterate over all possibilities.
- At price index j, there are $(k-1)-(j \mod k)$ prices left in the same day, then there are k prices the following day (if it exists).
- The last next price index is $f(j) = \min\{j + (k-1) - (j \mod k) + k, nk - 1\}$.
- This results in the relation $x(j) = 1 + \max\{x(d) : d \in \{j+1, \ldots, f(j)\}$ and $P[j] > P[d]\} \cup \{0\}$.

Topological order:

- $x(j)$ only depends on subproblems with strictly larger j, so relation graph is a DAG.

Base:

- When only one item remains, $x(nk-1) = 1$.

Original:

- The shorting value is $\max\{x(j) : j \in \{0, \ldots, nk-1\}\}$.

Time:

- There are nk subproblems, and $O(k)$ work per subproblem.
- The work for original is $O(nk)$, but is dominated by the work for the subproblems.
- Therefore, the overall running time is $O(nk^2)$.

(a)

(b) Submit your implementation to `alg.mit.edu`.