CMPSC 465 – Spring 2021 — Solutions to Homework 10

Josiah Kim, SID 948500821, `juk483`

April 22, 2021

1. Getting started

(a) I did not work in a group.

(b) I did not consult with any of my group members.

(c) I did not consult any non-class materials.

2. Unit-Time Task Scheduling

The greedy algorithm for the maximum independent subset problem shows us that checking if $A \cup \{x\} \in I$ for every $x \in S$ will take $O(n * f(n))$ time. Here, $O(n)$ denotes the loop which checks all $x \in S$. For this step of the algorithm to be done in $O(n)$ time, we need to show that $O(f(n)) = O(n)$.

We know that there are only n number of x. To check if $A \cup \{x\} \in I$ we just need to check if $x \leq t$ (according to the lemma). Therefore, $O(f(n)) = O(n)$.

3. Destroying Missiles

Subproblem:
The maximum number of robots that can be destroyed at an instance in $\{x_1, ..., x_j\}$, $emp(j)$.

Recurrence:
$emp(j) = max_{0 \leq i \leq j}(emp(i) + min\{x_j, f(j-1)\})$

Algorithm:

```
Input(s): Missile arrivals {x_1, ..., x_n} and recharging function, f.
Output: Maximum number of missiles that can be
destroyed by a sequence of EMP activation, N.

function destroy_missiles({x_1, ..., x_n}, f):
        emp(0) := 0
        for j=1 to n do
                emp(j)
        return emp(n)
```

Runtime:
The algorithm runs n-times for each n, $O(n^2)$

4. Card Game

(a) The greedy approach to this problem would be to compare the values of the first and last cards of the available sequence. Then, always pick the card with the largest value. Below is an example of when the greedy approach is not optimal for the first player.

A sequence of cards, $S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$.
Values of cards, $v_{s_1} = 2$, $v_{s_2} = 1$, $v_{s_3} = 4$, $v_{s_4} = 3$, $v_{s_5} = 20$, $v_{s_6} = 5$.
Say both players use the greedy approach. The first player would choose $v_{s_6} = 5$. The second player would then end up choosing the greatest valued card, $v_{s_5} = 20$. In the end, the first player would end up with a total value of 10 while the second player ends up with a total value of 25.

The better approach would be to use a dynamic approach to picking cards. The player would want to first look at the entire sequence of cards rather than make "greedy" choices.

(b) Subproblem: $f(i, j) = maxvalueatoneinstancewithiandjcardsareleft$

Recurrence:

$$f(i,j)_{0 \le i \le j \le n} = max \begin{cases} s_i + min(f(i+2, j), f(i+1, j-1)), \\ s_j + min(f(i+1, j-1), f(i, j-2)) \end{cases} \tag{1}$$

Algorithm:

```
Input(s): Sequence of cards, {s_1, ..., s_n} and card values, v.
Output: Matrix with precomputed values for each turn

function card_game({s_1, ..., s_n}, v):
  E = initialize n*n matrix
    for i = 0 to n do
      for j = 0 to n do
        when only two cards left:
          E(i, j) = max(s_i, s_j)
        when only one card left:
          E(i, j) = s_i
        else:
          E(i,j) = max(s_i + min(E(i+2, j), E(i+1, j-1)),
                       s_j + min(E(i+1, j-1), E(i, j-2)))
  return E
```