H3.PATEL.NISARGS

PROBLEM 1 *Compute the FFT on the values* $(5, 8, 1, 2, 2, 0, 2, 2)$. *Illustrate the steps for the first level of recursion, you can assume the base case occurs at* $n = 4$. *You can leave your answers in terms of* $\omega_1, \omega_3, \omega_5, \omega_7$, *i.e., without multiplying those roots out.*

Ans. The given function is $A(x) = 5 + 8x + 1x^2 + 2x^3 + 2x^4 + 0x^5 + 2x^6 + 2x^7$. For this equation we have:

$$\begin{aligned} A_e(x) &= 5 + 1x + 2x^2 + 2x^3 \\ A_o(x) &= 8 + 2x + 0x^2 + 2x^3 \end{aligned} \tag{1}$$

Calculating the values at 4th-roots of unity:

$$\begin{array}{cccc} A_e(1) = 10 & A_e(i) = 3 - i & A_e(-1) = 4 & A_e(-i) = 3 + i \\ A_o(1) = 12 & A_o(i) = 8 & A_o(-1) = 4 & A_o(-i) = 8 \end{array}$$

Since, $A(x) = A_e(x^2) + xA_o(x^2)$.
Calculating the values for 8th-roots of unity, we have:

$$\begin{aligned} A(\omega_{0,8}) &= A_e(\omega_{0,8}) + \omega_0 * A_o(\omega_{0,8}) = A_e(1) + A_o(1) = 22 \\ A(\omega_{1,8}) &= A_e(\omega_{2,8}) + \omega_1 * A_o(\omega_{2,8}) = A_e(i) + \omega_1 * A_o(i) = 3 - i + 8\omega_1 \\ A(\omega_{2,8}) &= A_e(\omega_{4,8}) + \omega_2 * A_o(\omega_{4,8}) = A_e(-1) + i * A_o(-1) = 4 + 4i \\ A(\omega_{3,8}) &= A_e(\omega_{6,8}) + \omega_3 * A_o(\omega_{6,8}) = A_e(-i) + \omega_3 * A_o(-i) = 3 + i + 8\omega_3 \\ A(\omega_{4,8}) &= A_e(\omega_{0,8}) + \omega_4 * A_o(\omega_{0,8}) = A_e(1) - A_o(1) = -2 \\ A(\omega_{5,8}) &= A_e(\omega_{2,8}) + \omega_5 * A_o(\omega_{2,8}) = A_e(i) + \omega_5 * A_o(i) = 3 - i + 8\omega_5 \\ A(\omega_{6,8}) &= A_e(\omega_{4,8}) + \omega_6 * A_o(\omega_{4,8}) = A_e(-1) - i * A_o(-1) = 4 - 4i \\ A(\omega_{7,8}) &= A_e(\omega_{6,8}) + \omega_7 * A_o(\omega_{6,8}) = A_e(-i) + \omega_7 * A_o(-i) = 3 + i + 8\omega_7 \end{aligned} \tag{2}$$

Thus we get the following 8 points for the the function using FFT:

$$\begin{array}{cc} A(1) = 22 & A(\omega_1) = 3 - i + 8\omega_1 \\ A(i) = 4 + 4i & A(\omega_3) = 3 + i + 8\omega_3 \\ A(-1) = -2 & A(\omega_5) = 3 - i + 8\omega_5 \\ A(-i) = 4 - 4i & A(\omega_7) = 3 + i + 8\omega_7 \end{array}$$

PROBLEM 2 *Lasers!*

The NASA Near Earth Object Program lists potential future Earth impact events that the JPL Sentry System has detected based on currently available observations. Sentry is a highly automated collision monitoring system that continually scans the most current asteroid catalog for possibilities of future impact with Earth over the next 100 years.

This system allows us to predict that i years from now, there will be x_i tons of asteroid material that has near-Earth trajectories. In the mean time, we can build a space laser that can blast asteroids. However, each laser blast will require *exajoules* of energy, and so there will need to be a recharge period on the order of *years* between each use of the laser. The longer the recharge period, the stronger the laser blast; e.g. after j years of charging, the laser will have enough power to obliterate d_j tons of asteroid material. This problem explores the best way to use such a laser.

Example Suppose $(x_1, x_2, x_3, x_4) = (1, 10, 10, 1)$ and $(d_1, d_2, d_3, d_4) = (1, 2, 4, 8)$. The best solution is to fire the laser at times $3, 4$ in order to blast 5 tons of asteroids.

(a) Construct an instance of the problem on which the following "greedy" algorithm returns the wrong answer:

```
BadLaser($(x_1, \ldots, x_n), (d_1, \ldots, d_n)$)
1  Compute the smallest $j$ such that $d_j \geq x_n$. Set $j = n$ if no such $j$ exists.
2  Shoot the laser at time $n$.
3  if $n > j$ then BadLaser($(x_1, \ldots, x_{n-j}), (d_1, \ldots, d_{n-j})$).
```

Intuitively, the algorithm figures out how many years (j) are needed to blast all the material in the last time slot. It shoots during that last time slot, and then accounts for the j years required to recharge for that last slot, and recursively considers the best solution for the smaller problem of size $n - j$.

(b) Given an array holding x_i and d_i, describe an algorithm computes the most amount of asteroid material that can be blasted. Before presenting the pseudo-code for the algorithm, present a DP equation that characterizes the optimum choice as we have done in class. Analyze the running time of your solution.

Ans.

(a) The "greedy" algorithm fails for $(x_1, x_2, x_3, x_4) = (1, 7, 7, 9)$ and $(d_1, d_2, d_3, d_4) = (1, 6, 8, 10)$. BadLaser shoots only once at $n = 4$ blasting total of 9 tons as $d_4 = 10$ is the smallest index of d such that $d_4 > n_4$.
A better solution can be obtained by shooting at $n = 2$ and 4 for total of 12 tons.

(b) The problem at any particular time i can be thought to be the maximum of shooting at time i with an arbitrary j years of recharge + the maximum tons shot till time $i - j$.

Thus, based on inputs x_i and d_i, we can write a DP equation as:

$$Best_i = \begin{cases} 0, \text{ if } i = 0 \\ \max_{j=1}^{i}\{Best_{i-j} + min(x_i, d_j)\} \end{cases} \tag{3}$$

The answer is $Best_n$. Momoization can be used by finding $Best_0$, then $Best_1$, then $Best_2$, and so on till $Best_n$.

Pseudocode for BestLaser:

```
BestLaser$((x_1, \ldots, x_n), (d_1, \ldots, d_n))$
1  $Best_0 \leftarrow 0$
2  for $i \leftarrow \{1, ..., n\}$
3      for $j \leftarrow \{1, ..., i\}$
4          $Best_i \leftarrow \max(Best_{i-j} + \min(x_i, d_j))$
5  return $Best_n$
```

Time Complexity Analysis:
Line 2 is called n times. Line 3 for each iteration is called for i times.
Thus this will be called for $1 + 2 + 3 + .. + n = (n)(n+1)/2 = \Theta(n^2)$.
So running time of BestLaser is $\Theta(n^2)$.

PROBLEM 3 *Price Run*

Given a list of closing stock ticker prices $p_1, p_2, \ldots, p_n$, devise an $O(n^2)$ algorithm that finds the length longest (not necessarily consecutive) streak of prices that increase or stays the same. For example, given the prices $2, 5, 2, 6, 3, 3, 6, 7, 4, 5$, there is the streak $2, 5, 6, 6, 7$ of prices that increase or stay the same, but an even longer streak is $2, 2, 3, 3, 4, 5$. Thus, the algorithm should return 6.

(Challenge: by using both dynamic programming and binary search, you can solve this problem in $O(n \log n)$ time.)

Note, you do not need to output the actual run of prices, only the length of the longest run.

Ans. For the last stock price to be included in the solution p_i, the length of longest streak of price will be 1+ the the maximum of all the streaks that are before i and whose last value is $< p_i$.

Thus, given the inputs $p_1, p_2, ..., p_n$, we can write a DP equation as:

$$Best_i = 1 + \max_{j=1}^{n}\{0, Best_j \text{ if } p_j \le p_i\} \tag{4}$$

After calculating, the answer is $\max_{j=1}^{n}\{Best_j\}$. Momoization can be used by finding $Best_1$, then $Best_2$, and so on till $Best_n$.

Pseudocode for PriceRun:

```
PriceRun($p_1, p_2, \ldots, p_n$)
1  for $i \leftarrow \{1, ..., n\}$
2      $Best_i \leftarrow 0$
3      for $j \leftarrow \{1, ..., i-1\}$
4          if $p_j \le p_i$, $Best_i \leftarrow \max(Best_j)$
5      $Best_i \leftarrow Best_i + 1$
6  return $\max(Best_i$, for $i \leftarrow \{1, \ldots, n\})$
```

Time Complexity Analysis:
Line 1 is called n times. Line 3 for each iteration is called for $i-1$ times.
Thus Line 4 will be called for $\Theta(n^2)$. Line 7 is called for $\Theta(n)$.
So, overall running time of PriceRun is $\Theta(n^2)$.