Due: Tuesday July 3rd at 6:00PM in class, or Thursday July 5th at 6:00PM electronically
Worth: 4%.
For each question you have to clearly write your algorithm in English and prove that it finds the optimal answer. In any of your answers you can use any algorithm we discussed in class without proving it solves the problem we discussed in class optimally. If we discussed the runtime of the algorithm you can also use that without reproving it. The same goes for any Lemma, Theorem or Fact we discuseed in class.
This assignment is about dynamic programming and my solutions are all using that technique, however you are welcome to write any correct solution. If you use dynamic programming it is imperative that you give a very precise, clear definition of the array you are filling. That means something like "$D[k]$ is the minimum number of 1in $\times$ 1in $\times$ 1in bricks you need to build a filled $k \times k \times k$ block made out of bricks." It does *not* mean $D[k] = \min_{1 \le i < k}(D[i] + i^2 k)$, that is a recurrence relation not a definition. You first give the definition of the elements of the array, then the recurrence relation and then your program.

Question #1: PLISP but better (6pt)

PLISP is the future of programming languages. A PLISP program is just a string of open and close parenthesis which makes it perfect for programming on keyboards which only have two keys. A machine "executes" the program by reading it from left to right keeping track of open and closed parenthesis. A valid program will always be a proper parenthetical expression. Furthermore, because the machine has limited memory at no point during the execution there should be more than l parentheses opened which are not closed yet. In other words a string $x[1 \dots n]$ is a valid program if and only if it has all the following properties:

1. It is made of only the two characters "(" and ")",
2. The number of "(" characters in x should be the same as the number of ")" characters,
3. For any $1 \le i \le n$, the number of "(" in $x[1 \dots i]$ has to be no less than the number of ")" in $x[1 \dots i]$,
4. For any $1 \le i \le n$, the number of "(" in $x[1 \dots i]$ has to be no more than the number of ")" in $x[1 \dots i]$ plus l.

We call the number of valid programs (of length n) $P_{l,n}$. Your task is to devise an algorithm that takes n, l, p as input and prints $P_{l,n}$ modulo p. You can assume that $p > 1$ and $l \le n$. Prove that your algorithm is correct and analyze its runtime. To get the full marks your algorithm should run in time $O(n^2)$.

Example For $n = 4$, $l = 2$, $p = 100$ the answer is 2.
For $n = 4$, $l = 1$, $p = 15$ the answer is 1.
For $n = 6$, $l = 3$, $p = 5$ the answer is 0 because the number of possible programs is 5 which is 0 modulo 5.
For $n = 6$, $l = 2$, $p = 5$ the answer is 4.

Hint: Don't try to give a closed form for the answer, use dynamic programming.

Question #2: Catching Apples and oranges (5pt)

It is raining apple and oranges and you need to catch them! To be precise there are n fruits that are falling on an l meter long road and you have a truck which is originally parked at the start of the road that you can use to catch them. You know that the ith fruit will drop at position x_i of the road (i.e. x_i meters from the start of the road) at time t_i. If your truck is at position x_i at time t_i you catch that fruit (for simplicity we think of the truck as a point) otherwise the fruit will drop on the road and will ruin. You can drive the truck on the road but its maximum speed is s meters per second. The t_i's are given in seconds from the beginning (at the beginning the truck is at the start of the road.)
Your task is to devise an algorithm that takes $l, n, s,$ and $x_1, t_1, x_2, t_2, \ldots, x_n, t_n$ as input and outputs the maximum number of fruits that you can catch. Prove that your algorithm is correct and analyze its runtime. To get the full marks your algorithm should run in time $O(n^2)$.

Example If $n = 6, l = 10, s = 2$, $x_1 = 6, t_1 = 3$, $x_2 = 2, t_2 = 2$, $x_3 = 1, t_3 = 2$, $x_4 = 1, t_4 = 1$, $x_5 = 10, t_4 = 6$, $x_6 = 0, t_4 = 11$ then the answer is 4. Here is how you drive your truck to catch that many fruits:

- You start your truck at time 0 and drive towards position 1 you will get there at time 0.5,
- You wait until time 1 and collect the fruit number 4,
- You drive towards position 2 and get there at time 1.5,
- You wait until time 2 and collect the fruit number 2,
- You drive towards position 10 and get there at time 6 and collect the fruit number 5,
- You drive towards position 0 and get there at time 11 and collect the fruit number 6.

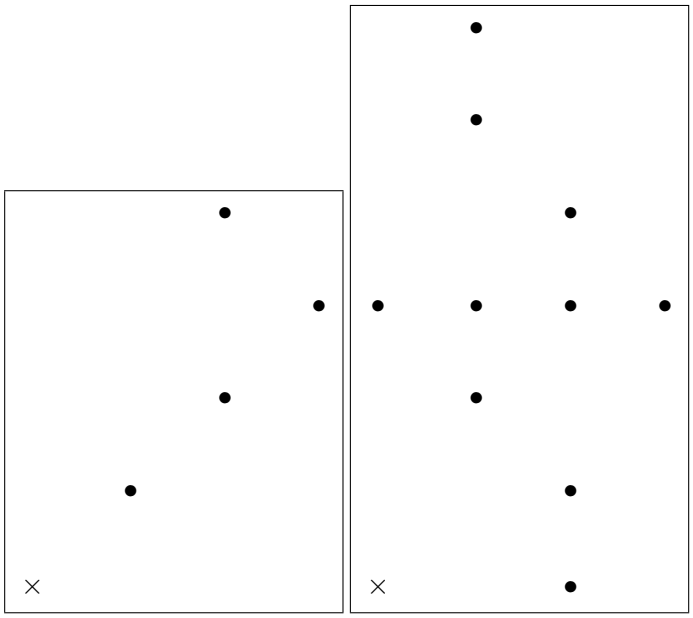
The reason you can not collect more than 4 fruits is that from fruits 1, 2, and 3 you can collect at most 1; you can only collect one of the fruits 2 and 3 because they drop at the same time at different places and if you collect either you can't get in time to position 6 to collect fruit 1.

Question #3: Collection coins: The multiplayer edition (5pt)

You and your friend are playing "Collecting coins: going up and right" game discussed in class but now in multiplayer! To be precise they are n coins on the plain. The ith coin is at point (x_i, y_i) with $x_i, y_i \geq 0$. You and your friend both start at point $(0, 0)$ and you should each take a path starting from there going only up and right, that is you can only move in the direction of increasing x or y. A coin is "collected" if it is on your or your friend's path; but if it is on both of your paths it will only be collected one. Devise an algorithm that takes n and $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ and outputs the maximum number of coins that you and your friend can collect together. Prove that your algorithm is correct and analyze its runtime. To get the full marks your algorithm should run in time $O(n^3)$.

Example: Figure 1a shows an example with $n = 4$ and coins at $(1, 1), (2, 2), (3, 3), (2, 4)$; the answer is 4 You will take the path $(0, 0) \to (1, 1) \to (2, 2) \to (3, 3)$ and your friend will take the path $(0, 0) \to (2, 5)$. Notice how the answer is not 6 because although your friend could take the path $(0, 0) \to (1, 1) \to (2, 2) \to (2, 5)$ the first two coins are already collected by you and won't count twice.

Figure 1b shows an example with $n = 10$ and coins at $(2, 0), (2, 1), (1, 2), (0, 3), (1, 3), (2, 3), (3, 3), (2, 4), (1, 5), (1, 6)$; the answer is 8. You will take the path $(0, 0) \to (2, 0) \to (2, 1) \to (2, 3) \to (2, 4)$ and your friend will take the path $(0, 0) \to (1, 2) \to (1, 3) \to (1, 5) \to (1, 6)$.



(a) Example 1 (b) Example 2

Figure 1: Examples for Question 3. Coins are shown with • and the starting point with ×.