1. Your reading class has given you a list of books to read. Each book can be considered as an ordered pair (t, d) where t is the time it will take for you to read the book and d is the date the book is due (maybe there is an exam on the book that day and you will only succeed if you read the whole book.)

   You realize that this class is challenging and you may not be able to read all the books by their due dates. Your goal is to read the maximum number of books before they are due. You can only read one book at a time.

   For Example: for the following input,

   $$[(5, 8), (3, 6), (2, 5), (4, 7)]$$

   this would mean that book 1 takes 5 days to read and it is due on the eighth day, book 2 takes 3 days to read and it is due on the sixth day, book 3 takes 2 days to read and it is due on the fifth day, and book 4 takes 4 days to read and it is due on the seventh day.

   One solution would be to read the books 3 then 2 and your output would look like: $[(2, 5), (3, 6)]$. This would correspond to reading book 3 for days 1 and 2, completing it on time. Then reading book 2 for days 3,4 and 5, completing it on time. (Note: this may or may not be the optimal solution)

   Candidate Greedy Strategy 1: Sort the books by t. Read the book with the shortest time first (that can be completed by the due date.) Then continue with the remaining books that can be completed on time.

   Candidate Greedy Strategy 2: Sort the books by d. Read the book with the earliest achievable due date first. Then continue with the remaining books that can be completed on time.

   Consider the two Greedy strategies above. **Neither strategy will always produce an optimal solution.**

   - give an example input to where Candidate Greedy Strategy 1 does not output an optimal solution (3 points)

     Solution: Consider the input:

     $$[(2, 13), (3, 16), (9, 10)]$$

     Then if you pick the shortest time book first, you would pick $(2, 13)$ then $(3, 16)$ and you wouldn't have time to finish $(9, 10)$. Instead, you could read the $(9, 10)$ book first, then read $(2, 13)$ on time and $(3, 16)$ on time.

   - give an example input to where Candidate Greedy Strategy 2 does not output an optimal solution (3 points)

     Solution: Consider the input:

     $$[(1, 6), (1, 6), (1, 6), (1, 6), (1, 6), (4, 5)]$$

     Then if you pick the earliest due date book first, you would pick $(4, 5)$ then one of the $(1, 6)$ books and you wouldn't have time to finish the other books. Instead, you could read all five $(1, 6)$ books.

2. Suppose you are running an airline that only runs old-timey planes. A typical plane in your fleet has n rows such that each row has 2 seats, one on each side of the plane. The plane will not properly function unless each row is *reasonably* balanced, meaning that the difference in the weights of the passengers in each row is less than or equal to W. There are $2n$ people who are scheduled for your next flight and you are given a list of their weights.

Your goal is to arrange as many people on the plane so that in each occupied row, the difference of the weights of the two passengers is less than or equal to W. (It may be the case that not everyone will fit on the plane.)

Candidate Greedy Strategy 1: Sort the passengers by weight: $w_1 \leq w_2 \leq \ldots \leq w_n$. If $w_2 - w_1 \leq W$ then put w_2 and w_1 in the first row and continue on the remaining passengers starting with w_3 and w_4. Otherwise, consider w_2 and w_3.

Candidate Greedy Strategy 2: Sort the passengers by weight: $w_1 \leq w_2 \leq \ldots \leq w_n$. Start with w_1: If there are no passengers to pair with w_1, i.e., $w_j - w_1 > W$ for all $j = 2, \ldots, n$ then start with w_2. Otherwise, find the heaviest passenger w_i that you can pair with w_1, i.e. that $w_i - w_1 \leq W$. Put w_1 and w_i in the first row and continue with the remaining passengers.

Consider the two Greedy strategies above. One produces an optimal solution the other does not.

- Identify the one that produces optimal solutions (2 points)

  Solution: Candidate Greedy Strategy 1

- give a counter-example to where the other strategy does not output an optimal solution (3 points)

  Solution: Consider the input:

  $$W = 3, [3, 4, 6, 8]$$

  Then Candidate Greedy Strategy 2 pairs $(3, 6)$ but then we can't pair $(4, 8)$. Instead, pair $(3, 4)$ and $(6, 8)$.

- prove that the correct strategy is optimal (10 points)

  Proof:(exchange argument)
  ExArg Claim: *For some input $I = (w_1, \ldots, w_n)$, assuming that $w_2 - w_1 \leq W$, let OS be some solution that does not pair (w_1, w_2). Then there is a solution OS' that does pair (w_1, w_2) and has at least as many pairs as OS.*
  Proof of ExArg: Suppose that OS is a solution of I that does not pair (w_1, w_2). Consider the following 3 cases:

  Case 1: w_1 and w_2 are both unpaired in OS. Then create OS' by adding the pair (w_1, w_2) to OS.

  OS' is valid: Since OS was valid and did not pair w_1, w_2 and $w_2 - w_1 \leq W$, then (w_1, w_2) is a valid pair so OS' is valid.
  OS' is just as good as OS: there is one more pair in OS' than OS so $|OS'| > |OS|$.

  Case 2: w_1 is paired with w_j and w_2 is unpaired. (or similarly, w_2 is paired with w_j and w_1 is unpaired.) Then create OS' from OS by removing the pair (w_1, w_j) and replacing it with (w_1, w_2)

  OS' is valid: Since OS was valid and $w_2 - w_1 \leq H$, then (w_1, w_2) is a valid pair so OS' is valid.
  OS' is just as good as OS: there the same number of pairs in OS' than OS so $|OS'| = |OS|$.

  Case 3: w_1 is paired with w_j and w_2 is paired with w_i. Then create OS' from OS by removing the pairs $(w_1, w_j), (w_2, w_i)$ and replacing them with $(w_1, w_2), (w_i, w_j)$

  OS' is valid: By assumption $w_2 - w_1 \leq H$ so (w_1, w_2) is a valid pair.

  Subcase 1: $w_i \leq w_j$: Then $w_j - w_i \leq w_j - w_1 \leq W$.
  Subcase 2: $w_i > w_j$: Then $w_i - w_j \leq w_i - w_2 \leq W$.

  So by these two cases, (w_i, w_j) make a valid pair, and with the validity of OS, OS' is valid.
  OS' is just as good as OS: there the same number of pairs in OS' than OS so $|OS'| = |OS|$.

Induction Part:
Claim: *For any input of any size $n \geq 2$, the greedy solution is optimal.*
Base Case: For $n = 2$, then if the two passengers can be paired, the greedy strategy pairs them.
Induction Hypothesis: Suppose that for some $n > 2$, the greedy strategy is optimal for all inputs of size k such that $2 \leq k \leq n-1$.
Induction Step: Suppose $I = (w_1, \ldots, w_{2n})$ is an input of size $2n$ (ordered by height.) Then let OS be any solution.

Case 1: suppose that $w_2 - w_1 > W$, then if w_2 can't sit in the same row with w_1, then nobody can sit in the same row with w_1 because everyone is heavier than w_2, therefore we can consider the subproblem without w_1 with $2n-1$ passengers $(w_2, \ldots, w_{2n})$ and by the inductive hypothesis, since there are fewer than $2n$ passengers, the greedy strategy is optimal.
Case 2: suppose that $w_2 - w_1 \leq W$, then let OS be any solution. Then by the Exchange Argument, there exists a solution OS' that includes (w_1, w_2) and has at least as many pairs as OS. Therefore, we have that:

$$|OS(I)| \leq |OS'(I)| = |\{(w_1, w_2)\} \cup S(I')| \leq |\{(w_1, w_2)\} \cup GS(I')| = |GS(I)|$$

where $I' = (w_3, \ldots, w_n)$

- give an efficient implementation and time analysis for the correct strategy (5 points).

Implementation:

- First sort the passengers by weight in increasing order $w_1 \leq w_2 \leq \ldots \leq w_n$.
- Then proceed with the following recursive algorithm:

```
Passengers(w_1, ..., w_n; W)
    if n < 2:
        return ∅
    if w_2 − w_1 > W:
        return Passengers(w_2, ..., w_n, W)
    else:
        return (w_1, w_2) ∪ Passengers(w_3, ..., w_n; W)
```

Runtime: Sorting takes $O(n \log(n))$. Then if $T(n)$ is the time taken for the algorithm **Passengers**, there is either a recursive call of size $n-1$ or of size $n-2$, in either case, $T(n) \leq T(n-1) + c$ for some constant time c for the non-recursive part. Therefore the recursive algorithm runs in $O(n)$ time.

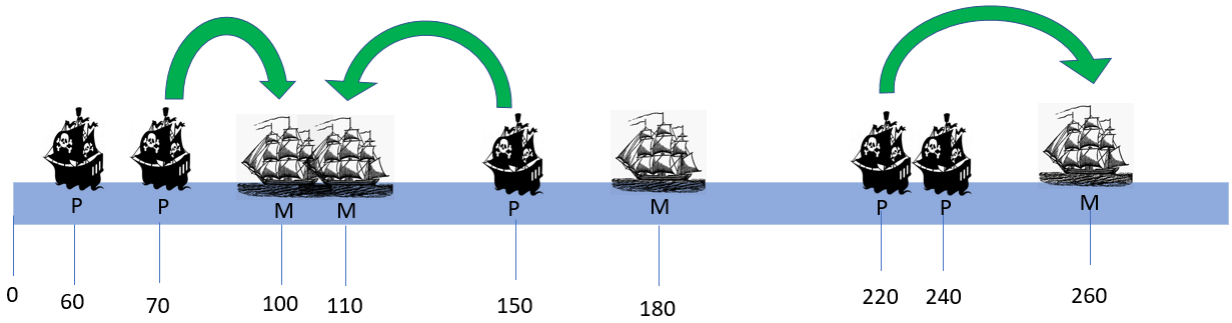
3. Suppose there is a sequence of ships along a straight river. Each ship in the line is either a merchant ship or a pirate ship. The pirate ships are at positions $P = [p_1, p_2, \ldots, p_\ell]$ and the merchant ships are at positions $M = [m_1, m_2, \ldots, m_k]$ measured in meters from the mouth of the river. The pirate ships wish to sink the merchant ships. The pirate ships are armed with cannons that can shoot up to (and including) S meters. Each pirate ship has one cannonball.

   Your goal is to give the pirates orders on how to shoot their cannonballs in order to sink the maximum number of merchant ships. (assume that all ship positions are fixed and assume that all ships are in different positions.)

   For example: in the picture below, the input would be:

$$P = [60, 70, 150, 220, 240], \quad M = [100, 110, 180, 260], \quad S = 50.$$

   The solution corresponding to the green arrows is $[(70, 100), (150, 110), (220, 260)]$ (Note that this may not be the optimal solution.)



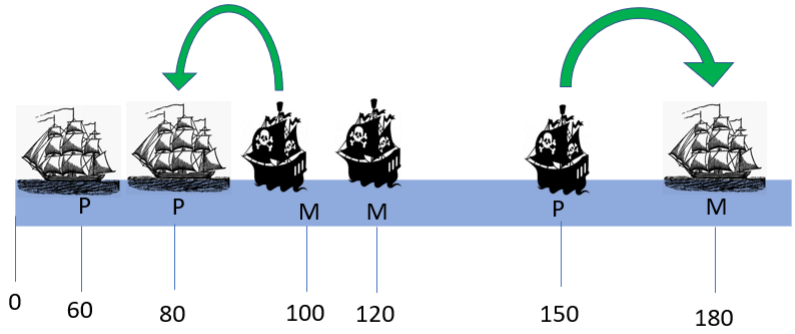
Candidate Greedy Strategy 1: Start from the first pirate ship in the list (with the lowest index.) Have him sink the nearest possible merchant. Then continue on the remaining ships.

Candidate Greedy Strategy 2: Start from the first pirate ship in the list (with the lowest index.) Have him sink the farthest possible merchant. Then continue on the remaining ships.

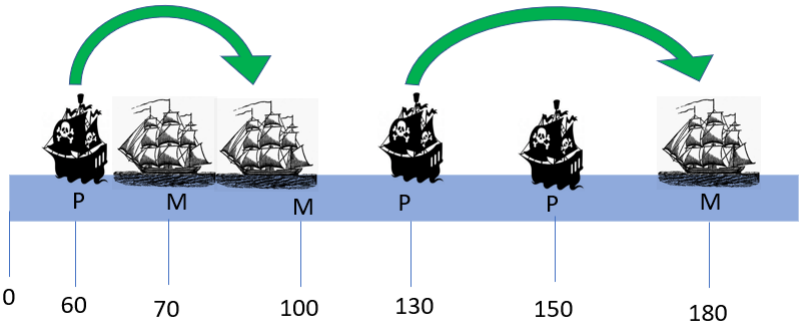
Consider the two Greedy strategies above. Neither of them are optimal.

- Give counter-examples to where the two strategies do not output an optimal solution (3 points each)

  Solution:
  The first greedy strategy on the following example with $S = 50$ will return: $[(100, 80), (150, 180)]$ but there is a solution that sinks 3 merchant ships: $[(100, 60), (120, 80), (150, 180)]$.



  The second greedy strategy on the following example with $S = 50$ will return: $[(60, 100), (130, 180)]$ but there is a solution that sinks 3 merchant ships: $[(60, 70), (130, 100), (150, 180)]$.



- Give a greedy strategy that will always return an optimal solution.

Solution: Give your first order to p_1, the left most pirate. Find the left-most merchant ship m_i that is in the range of p_1 and order p_1 to sink m_i and continue in this manner with p_2. If there is no merchant ship in the range of p_1 then continue in this manner starting with p_2.

- Prove that your strategy is optimal (10 points)

  Solution:
  Exchange Argument Claim: Suppose our instance is $I = [p_1, \ldots, p_\ell], [m_1, \ldots, m_k]$. Without loss of generality we can assume that at least one merchant is in the range of p_1, because if not, then p_1 is useless and we can instead consider $[p_2, \ldots, p_\ell], [m_1, \ldots, m_k]$. Suppose that m_g is the left-most merchant that is in the range of p_1. Then the first greedy choice is to pair (p_1, m_g). Let OS be a valid set of pairs that does not include (p_1, m_g).
  We claim that there exists a valid set of pairs OS' that does include (p_1, m_g) and $|OS| \leq |OS'|$.

  proof:
  - **Case 1:** In OS, p_1 is not ordered to sink a merchant and m_g is not targeted by any pirate. Create $OS' = OS \cup \{(p_1, m_g)\}$.
    OS' is valid because all of OS is valid and (p_1, m_g) is a valid pair. $|OS'| = |OS| + 1$.
  - **Case 2:** In OS, p_1 is not ordered to sink a merchant and m_g targeted by p'. Create $OS' = OS \cup \{(p_1, m_g)\} - \{(p', m_g)\}$.
    OS' is valid because all of OS is valid and (p_1, m_g) is a valid pair. $|OS'| = |OS|$.
  - **Case 3:** In OS, p_1 is ordered to sink m' and m_g is not targeted by any pirate. Create $OS' = OS \cup \{(p_1, m_g)\} - \{(p_1, m')\}$.
    OS' is valid because all of OS is valid and (p_1, m_g) is a valid pair. $|OS'| = |OS|$.
  - **Case 4:** In OS, p_1 is ordered to sink m' and m_g is targeted by p'. Create $OS' = OS \cup \{(p_1, m_g)\} - \{(p', m')\}$.
    $|OS| = |OS'|$. In order to show that OS' is valid, we need to establish that $|p' - m'| \leq S$. Suppose by contradiction that $|p' - m'| > S$.
    subcase 1: $p' > m'$. Then since m_g is the left-most merchant in the range of p_1, $m_g < m'$. Then $S > p' - m' > p' - m_g$. But (p', m_g) was a valid pair so $|p' - m_g| \leq S$.
    subcase 2: $m' > p'$. Then since $p' > p_1$, $S > m' - p' > m' - p_1$. But (p_1, m') was a valid pair so $|p_1 - m'| \leq S$.

  Induction Part:
  Claim: *For any input of any size $\ell + k \geq 2$, the greedy solution is optimal.*
  Base Case: For $\ell + k = 2$, then if there is a pirate and a merchant that can reach eachother then the greedy strategy will pair them.
  Induction Hypothesis: Suppose that for some $\ell + k > 2$, the greedy strategy is optimal for all inputs of size t such that $2 \leq t < \ell + k$.
  Induction Step: Suppose $I = [p_1, \ldots, p_\ell], [m_1, \ldots, m_k]$ is an input of size $\ell + k$ (each list ordered by position.) Then let OS be any solution of I.

  Case 1: There is no merchant ship in the range of p_1. Then p_1 is useless and the optimal solution to I is also the optimal solution to $I' = [p_2, \ldots, p_\ell], [m_1, \ldots, m_k]$ which is of size $\ell + k + 1$ which, by the inductive hypothesis, the greedy strategy will find the optimal solution.
  Case 2: Otherwise, suppose that the first greedy choice is (p_1, m_g). Then let OS be any solution of I. Then by the Exchange Argument, there exists a solution OS' that includes (p_1, m_g) has at least as many pairs as OS. Therefore, we have that:

  $$|OS(I)| \leq |OS'(I)| = |\{(p_1, m_g)\} \cup S(I')| \leq |\{(p_1, m_g)\} \cup GS(I')| = |GS(I)|$$

  where I' is the list of pirates without p_1 and the list of merchants without m_g.

- Give an efficient implementation and time analysis for the correct strategy (5 points).

  Solution:
  Design the algorithm recursively in the following way:

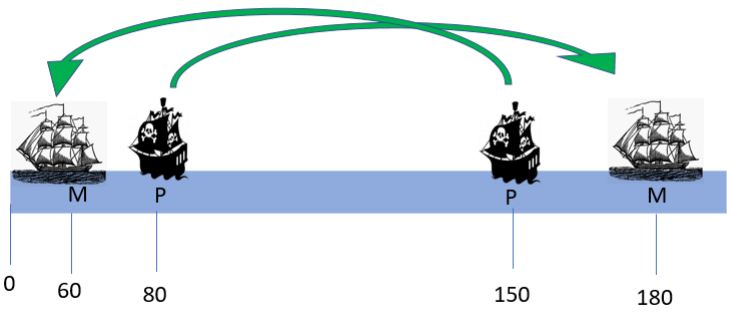
  Pirates$([p_1, \ldots, p_\ell], [m_1, \ldots, m_k], S)$
  if either list is empty:
  return $\emptyset$
  if $p_1 < m_1$:
  if $m_1 - p_1 < S$:
  return Pirates$([p_2, \ldots, p_\ell], [m_1, \ldots, m_k], S)$
  if $m_1 - p_1 \leq S$:
  return Pirates$([p_2, \ldots, p_\ell], [m_2, \ldots, m_k], S) \cup \{(p_1, m_1)\}$
  if $p_1 > m_1$:
  if $p_1 - m_1 < S$:
  return Pirates$([p_1, \ldots, p_\ell], [m_2, \ldots, m_k], S)$
  if $p_1 - m_1 \leq S$:
  return Pirates$([p_2, \ldots, p_\ell], [m_2, \ldots, m_k], S) \cup \{(p_1, m_1)\}$

  Let $n = \ell + k$. This algorithm makes only one recursive call based on p_1, m_1. The worst case would be a recursive call of size $n-1$. So, in the worst case: $T(n) = T(n-1) + O(1)$. So the runtime is $O(n)$.

4. Same setup as problem 3 except for it costs x grams of gunpowder to launch a cannonball x meters. Each pirate still has only one cannonball but now they can launch their cannonball as far as they want. Suppose now that there are n pirates and n merchants and you want to sink all n of the merchant ships using the least amount of total gunpowder.

   - give a counter-example to where **Candidate Greedy Strategy 1** does not output an optimal solution (3 points each)

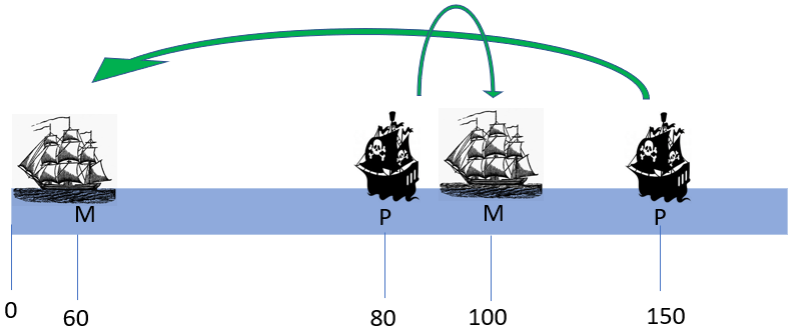
     For Candidate Greedy Strategy 1: Consider the input:

     Then Candidate Greedy Strategy 1 orders the pirate at 80 to shoot the pirate at 180 $(80, 180)$ and the pirate at 150 to shoot the pirate at 60 $(150, 60)$ for a grand total of $180 - 80 + 150 - 60 = 100 + 90$ units of gunpowder. Alternatively, you could have the pirates just shoot the merchant next to it for $20 + 30 = 50$ units of gunpowder.

   - give a counter-example to where **Candidate Greedy Strategy 2** does not output an optimal solution (3 points each)

     For Candidate Greedy Strategy 2: Consider the input:



Then Candidate Greedy Strategy 2 orders the pirate at 80 to shoot the pirate at 100 $(80, 100)$ and the pirate at 150 to shoot the pirate at 60 $(150, 60)$ for a grand total of $100 - 80 + 150 - 60 = 20 + 90 = 110$ units of gunpowder. Alternatively, you could have each pirate shoot the merchant to its left for 80-60=20 and 150-100=50 for a total of 70 units of gunpowder.

- Extra Credit (3 points): design a greedy strategy that will always produce an optimal solution.

  Just pair $(p_1, m_1), (p_2, m_2), \ldots, (p_n, m_n)$.