Oxen pairing Consider the following problem: We have \( n \) oxen, \( O_{x_1}, \ldots, O_{x_n} \), each with a strength rating \( S_i \). We need to pair the oxen up into teams to pull a plow; if \( O_{x_i} \) and \( O_{x_j} \) are in a team, we must have \( S_i + S_j \geq P \), where \( P \) is the weight of a plow. Each ox can only be in at most one team. Each team has exactly two oxen. We want to maximize the number of teams.

Candidate Greedy Strategy I: Take the strongest and weakest oxen. If together they meet the strength requirement, make them a team. Recursively find the most teams among the remaining oxen.

Otherwise, delete the weakest ox. Recursively find the most teams among the remaining oxen.

Candidate Greedy Strategy II: Take the weakest two oxen. If together they meet the strength requirement, make them a team. Recursively find the most teams among the remaining oxen.

Otherwise, delete the weakest ox. Recursively find the most teams among the remaining oxen.

Candidate strategy 2 is not always optimal. Say \( P = 10 \), and there are 4 oxen, with strengths 1, 2, 8, 9 and \( P = 10 \). If we try to pair the first two, they cannot pull the plow, so we would delete the 1. But then of the three remaining, we’ll get at most 1 pair. In contrast, we can pair the \((1, 0)\), and \((2, 8)\) to get two viable pairs, as strategy one would do on this example.

Candidate Greedy Strategy: Take the strongest and weakest oxen. If together they meet the strength requirement, make them a team. Recursively find the most teams among the remaining oxen.

Otherwise, delete the weakest ox. Recursively find the most teams among the remaining oxen.

This strategy is optimal. We’ll prove it using the following two lemmas:
Lemma 1: Let $s$ be the strongest ox, and $w$ the weakest. If $s + w < P$, then there is no set of teams that assigns $w$ to any team.

Proof: If $Teams$ is a set of teams in which $w$ is assigned to a team with some ox $s'$, since $s' \leq s$, $s' + w \leq s + w < P$, the team could not actually pull the plow. This contradiction proves the lemma.

Lemma 2: Let $s$ be the strongest ox, and $w$ the weakest. Assume $s + w \geq P$. Let $Teams_2$ be a set of disjoint teams that can all pull the plow does not pair $s$ with $w$. Then there is a set of disjoint teams $Teams_1$ that can all pull the plow, which assigns $w$ to a team with $s$ and is so that $|teams_1| \geq |Teams_2|$.

Proof: If $Teams_2$ does not assign both $s$ and $w$ to teams, let $Teams_1$ be $Teams_2$ less any team that includes $s$ or $w$, together with the team $(s, w)$. Since $s + w \geq P$, and $(w, s)$ is the only team we added, this is a set of disjoint teams that can all pull the plow. Since at most one of $s, w$ were in a team we deleted at most one team, and added one team, so $|Teams_1| \geq |Teams_2|$.

Otherwise, let $Teams_2$ partner $s$ with $x$ and $w$ with $y$. Let $Teams_1 = Teams_2 - \{(s, x), (w, y]\} \cup \{(x, y), (w, s)\}$. $Teams_1$ is a set of disjoint teams, and $(w, s)$ can pull the plow by our assumption. Now, since $w$ and $y$ can pull the plow, and $s$ is at least as strong as the weakest ox $w$, $x$ and $y$ can pull the plow. Thus, all teams that we added can pull the plow. Since we deleted two teams and added two teams, $|teams_1| = |teams_2|$. Thus, we have proved the lemma in both cases.

We can now prove that the greedy strategy is optimal:

Theorem: There is no set of legal teams $Teams_2$ greater than that produced by the greedy strategy.

We prove this by strong induction on $n$, the number of oxen. Assume that the greedy strategy is optimal on all sets of size $< n$. Then on a set $Oxen$ of size $n$, let $w$ and $s$ be the weakest and strongest oxen. Assume $Teams_2$ is larger than the greedy strategy’s solution $GreedyTeams$. If $w + s < T$, then by lemma 1, neither $Teams_2$ nor $GreedyTeams$ contains a team with $w$. Thus, by the induction hypothesis, $GreedyTeams$ is the best solution for $Oxen - \{w\}$, and $Teams_2$ is some solution for $Oxen - \{w\}$, so $|Teams_2| \leq |GreedyTeams|$.

If $w + s \geq T$, then by lemma 2, there is a solution $Teams_1$ which like the greedy solution, pairs $(w, s)$ and $|Teams_1| \geq |Teams_2|$. Then, since $GreedyTeams - \{(w, s)\}$ is an optimal solution for $Oxen - \{s, w\}$ by the induction hypothesis, and $Teams_1 - \{(w, s)\}$ is a legal solution for this problem, $|Teams_1 - \{(w, s)\}| \leq |GreedyTeams - \{(w, s)\}|$ so $|Teams_1| - 1 \leq |GreedyTeams| - 1$, so $|Teams_2| \leq |Teams_1| \leq |GreedyTeams|$. So the greedy solution also produces optimal teams on a set of size $n$. 

2
By induction, the greedy solution is optimal for all $N$.

To get an efficient version of the algorithm, first sort the oxen by strength. We either delete the weakest or both the weakest and strongest, so the set that is left is of the form $Oxen[i..j]$. We just need to keep track of $i$ and $j$. The following algorithm, after the input is sorted, does so:

1. $Teams \leftarrow \emptyset$
2. $I \leftarrow 1, J \leftarrow n.$
3. While $I < J$ do:
   4. IF $Oxen[I] + Oxen[J] \geq T$ THEN $Teams \leftarrow Teams \cup \{(I, J)\}$, $I + +, J - -$.
   5. ELSE $I + +$.
4. Return $Teams$.

Since $J - I$ always decreases by at least one, the above loop executes at most $n - 1$ times, so the above loop takes $O(n)$ time. However, we need to spend $O(n \log n)$ time to sort the inputs, which gives $O(n \log n)$ total time.

**Spectrum** You want to create a scientific laboratory capable of monitoring any frequency in the electromagnetic spectrum between $L$ and $H$. You have a list of possible monitoring technologies, $T_i, i = 1,...,n$, each with an interval $[l_i,h_i]$ of frequencies that it can be used to monitor. You want to pick as few as possible technologies that together cover the interval $[L,H]$.

**Candidate Greedy Strategy I:** First, buy the technology that covers the longest sub-interval within $(L,H)$. (i.e., The longest interval $(l,h)$, but not including the sub-intervals $(l,L)$ and $(H,h)$ outside the interval we need covering.) At each subsequent step, buy the technology that covers the largest total length that is still uncovered.

**Candidate Greedy Strategy II:** Look at all the technologies with $l \leq L$. Of these, buy the one $T_i = (l_i,h_i)$ with the largest value of $h_i$. Repeat the process to cover the remaining interval, $(h_i,H)$.

Say that the three technologies cover $[1,5],[5.9]$ and $[2.9]$ respectively, and we wish to cover $[1,0]$. If we pick the longest, which is the third, we need to also pick both of the others to completely cover the interval. But only picking the first 2 completely covers the interval. So Candidate Startegy I is not always optimal.

Candidate strategy 2 would only consider the first interval, since it is the only one containing 1. Then it would set $L$ to 5, and consider the second and third. It would buy the second and terminate, since the high end is greater for the second. Thus, in this example, candidate strategy 2 would be optimal.
Our overall strategy is: RecSpectrum (Techs, L, H):

1. If $L > H$ return $\emptyset$.
2. Let $Span$ be the set of technologies $T_i = (l_i, h_i)$ so that $l_i \leq L < h_i$.
3. If $Span$ is empty, return “Impossible”.
4. Otherwise, let $T_g = (l_g, h_g)$ be the technology in $Span$ with largest value of $h$.
5. $B_R \leftarrow$ RecSpectrum($Techs - Span, h_g, H$).
6. IF $B_R \neq “Impossible”$ return $B_R \cup \{T_g\}$, else return “Impossible”.

We want to show this algorithm is correct. First, we use an exchange argument.

Lemma 1: Let $T_g = (l_g, h_g)$ be the greedy technology. Let $B'$ be any set of technologies spanning $[L, H]$, with $T_g \notin B'$. Then there is a set $B$ of technologies spanning $[L, H]$ so that $T_g \in B$ and $|B| \leq |B'|$.

Proof: There must be a technology $T' = (l', h') \in B$ that covers $L + \epsilon$ for an arbitrarily small $\epsilon$. Thus, $l' \leq L < h'$. So $T' \in Span$. Thus, $h' \leq h_g$, since we chose $T_g$ to be the element in $Span$ with the largest value of $h$. Therefore, $B = B' - \{T'\} \cup \{T_g\}$ spans $[L, H]$, since it spans $[h', H]$ via $B' - \{T'\}$ and $T_g$ spans $[L, h']$. Also $|B| = |B'|$.

Next we need to show that our recursion is correct:

Lemma 2: If $B_R$ is a minimal set of technologies from $Tech - Span$ spanning $[h_g, H]$, then $\{T_g\} \cup B_R$ is a minimal set of technologies from $Tech$ including $T_g$ and spanning $[L, H]$.

$B_R$ spans $[h_g, H]$ and $l_g \leq L$ so $B_R \cup T_g$ spans $[L, H]$. Suppose there were a smaller set of technologies, $B_S$ including $T_g$ and spanning $[L, H]$. Then $B_S - \{T_g\}$ spans $h_g, H$. Also, since all elements of $Span$ have $h' \leq h_g$, $B_S - \{T_g\} - Span$ must still span $h_g, H$. $|B_S - \{T_g\} - Span| \leq |B_S - \{T_g\}| = B_S - 1 < |B_R \cup \{T_g\}| - 1 = |B_R|$. This contradicts the minimality of $B_R$. So by contradiction, there is no smaller set of technologies including $T_g$ and spanning $[L, H]$.

It follows that our algorithm is correct:

Theorem: The greedy algorithm outputs an optimal solution, if any solution exists.

Proof: by induction on the number of technologies. The base case, when there are 0 technologies, is possible if and only if we are spanning an empty interval, i.e., if $L > H$. This is what our algorithm checks in this case, since $Span$ will always be empty.

Assume the algorithm is correct when $|Tech| < n$, and let $|Tech| = n$. Then if $L > H$, we return the empty set, which has minimal size and
covers the empty interval. If $Span$ is empty, any technology $T' = (l', h')$ with $l' \leq L$ has $h' \leq L$. (We might have a technology that just detects exactly $L$, but this won’t help.) Then for some small $\epsilon$, no technology can cover $L + \epsilon$ (pick $\epsilon < \min_{T=(l,h), l>L} (l - L)$.) So the algorithm correctly returns “Impossible”.

On the other hand, if $Span$ is non-empty, let $B_R = \text{RecSpectrum}(\text{Tech} - Span, h_g, H)$. By the induction assumption, $B_R$ is a minimal set of technologies from $\text{Tech} - Span$ covering $h_g, H$. Then by Lemma 2, $B_R \cup \{T_g\}$ is a minimal set of technologies from $\text{Tech}$ covering $L, H$ containing $T_g$. From Lemma 1, it is also a minimal set of technologies from $\text{Tech}$ covering $L, H$.

Now, we can implement the algorithm faster without recursion. Namely, if we order the technologies according to $l$, then any technology with $l < L$ is either in $Span$ or has $h \leq L$ and so will never be in $Span$, since $L$ only increases in recursive calls. So if we trace forward in order of $l$, until we reach an $l > L$, keeping track of the largest $h$ encountered, this largest $h$ will either be from $T_g$ or if $h \leq L$ indicates that $Span$ is empty. So if $h \leq L$, we return “Impossible”, otherwise, we delete the earlier part of the list, add the corresponding $T$ to our list of bought technologies, and continue. This will be total time $O(n \log n)$ to sort the list, and then $O(n)$ time because we delete as we scan through the list. So the total time is $O(n \log n)$. 
