Problem Set 4

Problem 1

$I = [(0, 1), (0, 3), (4, 5), (2, 5)]$

These intervals are already sorted by finish time.

1. Initially, there are no classrooms allocated, so I_0 is placed in C_0.
2. Next, I_1 is incompatible with I_0, so it is placed in a newly allocated C_1.
3. I_2 is compatible with C_0 and C_1, so it is placed in C_0.
4. I_3 isn't compatible with C_0 or C_1 (because of I_2 and I_1 respectively), so it is placed in a newly allocated C_2

So we have...

- $C_0 = { I_0, I_2 }$
- $C_1 = { I_1 }$
- $C_2 = { I_3 }$

As you can see below, the maximum depth is 2, so this algorithm allocated more than max-depth classrooms.

```
I_1  +-----------------+     +-----+  I_2
I_0  +-----+     +-----------------+  I_3

     +-----+-----+-----+-----+-----+
     0     1     2     3     4     5
```

Problem 2

Algorithm

- Let S be the (initially empty) set of pairs
- Sort $A = a_1, a_2, \ldots, a_n$ in increasing order to get $B = b_1, b_2, \ldots, b_n$.
- While there are elements remaining, pick the smallest and largest element (the first and last b_i, b_j), and remove them from B.
- Add ${b_i, b_j}$ to S
- Once B is empty, return S.

```
# O(nlog(n))
def min_max_pairs(A):
  A.sort()

  l, r = 0, len(A) - 1
```

```
  res = -infinity
  while l < r:
    res = max(res, A[l] + A[r])
    l += 1
    r -= 1
  return res
```

Correctness: Greedy Stays Ahead

For notational convenience, let $a_1, a_2, \ldots, a_n$ be sorted. Let G be my algorithm, and X be the optimum, and for the sake of contradiction, suppose X chooses pairs differently from G, e.g. not of the form $(a_i, a_{n - i})$ as G does.

Let $g(a_i)$ be a function that outputs the "choice" of partner for any a_i by G, so $g(a_i) = a_{n - i}$, and let $x(a_i)$ mean the same for X.

Define $P(n)$ as...

Given n sorted numbers (with n being even) $a_1, \ldots a_n$, $G_{max} \le X_{max}$, where G_{max} and X_{max} are defined as follows:

$$ G_{max} = max({ g(a_i) + a_i : 1 \le i \le n }) $$

$$ X_{max} = max({ x(a_i) + a_i : 1 \le i \le n }) $$

Base Case:

$P(2)$: we have a_1, a_2, with $a_1 < a_2$. There is only one possible pair, so $G_{max} \le X_{max}$, and $P(2)$ holds.

IH: Suppose $P(2) \land P(4) \land \ldots \land P(k - 2)$ holds.

IS:

Let $A = a_1, a_2, \ldots, a_{k - 1}, a_k$ be sorted numbers. Remove a_1 and a_k to get $A' = a_2, \ldots, a_{k - 1}$. Since A' is still sorted, and has $|A'| = k - 2$, we know $P(k - 2)$ holds.

Therefore, we have $G'*{max} |le X'*{max}$ for A'.

Consider an arbitrary pair chosen by G on A', (a_i, a_j), with $a_i \le a_j$. By our sort order, we have $a_1 \le a_i \le a_j \le a_k$. Now, consider the ways we could swap elements between these two pairs

Case 1: $a_1 + a_k > a_i + a_j$

- $(a_1, a_j), (a_i, a_k)$
  - $a_i + a_k \ge a_1 + a_k$, since $a_i \ge a_1$
- $(a_1, a_i), (a_k, a_j)$
  - $a_k + a_j \ge a_1 + a_k$, since $a_j \ge a_i$

Case 2: $a_1 + a_k \le a_i + a_j$

- $(a_1, a_j), (a_i, a_k)$

  - $a_i + a_k \ge a_i + a_j$, since $a_k \ge a_j$
- $(a_1, a_i), (a_k, a_j)$
  - $a_k + a_j \ge a_i + a_j$, since $a_k \ge a_i$

In either of the above cases, we are increasing the max of the two sums by swapping elements, and this holds for an arbitrary a_i, a_j chosen by G', which is at least as good as the optimum. Since we know $X'{max} |ge G'{max}$, and also that any pairing other than the one picked by G (a_1, a_k) leads to an increased sum, it must be the case that $G_{max} < X_{max}$.

Problem 3

Let T_1 and T_2 be edge disjoint spanning trees over G. Consider an arbitrary edge $e = (u, v) \in T_1$.

Let $T_1' = T_1 - e$. Since T_1 was a tree, this splits T_1' into two connected components C_1, C_2, both of which are also trees (**justify**). We have $u \in C_1$ and $v \in C_2$.

Now, consider the vertices of C_1 and C_2.

Since T_2 is also a tree, and is therefore connected, there must exist an edge $f = (x, y) \in T_2$, such that $x \in C_1$ and $y \in C_2$, and so that the two components formed by this cut K_1 and K_2 contain u and v respectively. We can choose such an edge f by virtue of T_2 being a tree. For any two vertices in a tree, there is exactly one path between them. We know $(u, v) \notin T_2$, so letting $u, p_1, p_2, \ldots, v$ be the path between u and v in T_2, choose $f = (u, p_1)$.

Cutting T_2 on f to get T_2', we have two connected components K_1, K_2, both of which are trees. Now, we can add e to T_2', and we get a tree, since K_1 and K_2 are both trees, with $u \in K_1$ and $v \in K_2$.