WA6 - Assignment Group 16

Sravani Kirla - skirla
Matthew Heil - heilx069
Terrence Hanrahan - hanra028

1.

```
def sortPoints(points):
    points.sort()
    smallestSet = []
    leftPoint = points[0]
    rightPoint = leftPoint + 1
    newPoint = [leftPoint,rightPoint]
    smallestSet.append(newPoint)

    for i in range(1,len(points)):
        if (leftPoint<=points[i]<=rightPoint):
            None
        else:
            leftPoint = points[i]
            rightPoint = leftPoint + 1

            if (leftPoint<=points[i]<=rightPoint):
                newPoint = [leftPoint,rightPoint]
                smallestSet.append(newPoint)
```

Proof:

Let's take the optimal choice A = { [1.5, 2.5], [7.0, 8.0], [8.5, 9.5] }.

Greedy Choice: By applying greedy choice property, as per above pseudocode, we pick first interval starting at 1.8 which leads to unit interval of [1.8, 2.8].

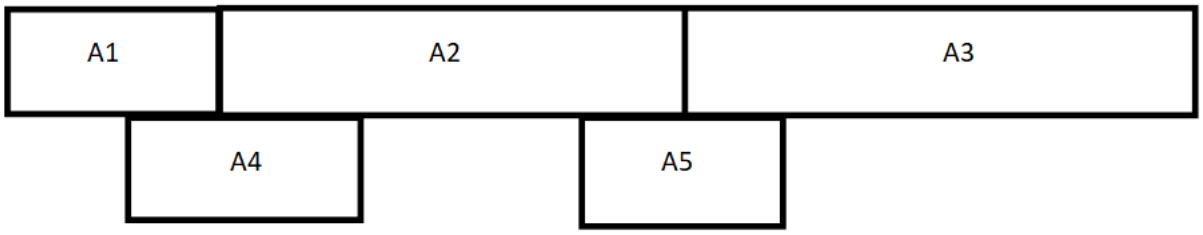
Case 1: A contains greedy choice.

- If A starts with interval [1.9, 2.9] it still satisfies minimum-size set condition.

Case 2: A does not contain greedy choice.

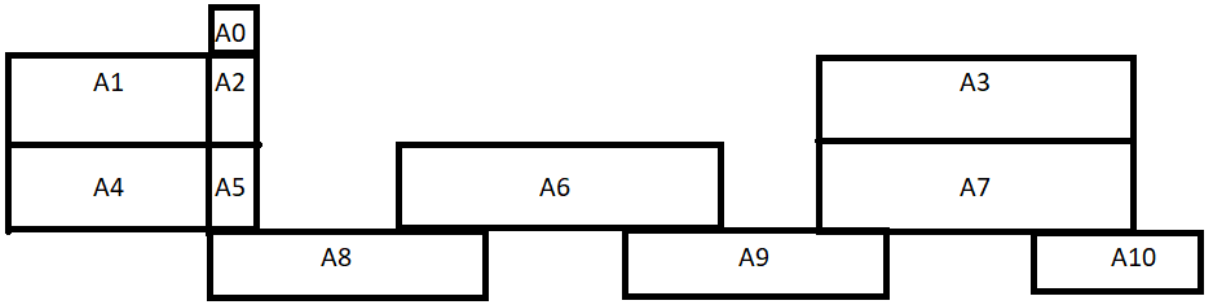
- Let's replace first interval in set A with greedy choice to get A'
- A' = { [1.9, 2.9], [7.0, 8.0], [8.5, 9.5] }
- By substituting greedy choice we can prove that all the points are covered and size of A' is equal to the size of minimum-size set A.
- Assuming an interval in A is [x,y] and A' is [x', y'].
    - Won't miss any points with greedy choice on the left side as we always start interval with a point. (with greedy choice x` is always one of the points in a given set)
    - Can't miss any points in the intervals of set A on the right side of interval because y` > y
    - Can't miss any other points by applying same greedy choices to all other intervals by definition of above two points.

2.a.



Using the above example schedule and the choice of choosing the next compatible activity with the least duration, it is seen that if A1 are chosen as the first activity, A5 will be chosen yielding a subset size of two {A1, A5}. However, this is not the optimal subset since activities A1, A2, and A3 are all compatible with each other and yield a subset size of three {A1, A2, A3}. This is despite of the fact A2 and A3 are much longer length than A5.

b.



Total Overlaps Per Activity:
A1, A4: 1
A6, A10: 2
A0, A2, A3, A5, A7, A9: 3
A8: 4
Using the example above, selecting the compatible activities with the fewest overlaps results in a set like {A1, A6, A10} with set size 3. (A1 could be interchanged with A4 as is both overlap 1 activity). A0, A2, and A5 overlap 3 activities, A8 overlaps 4 activities, whereas A6 only overlaps 2 activities, making it the chosen activity. All the previous activities were compatible with A1 or A4. After A6 is chosen, A9 is incompatible, so the remaining options are A3, A7, and A10. A3 and A7 overlap 3 activities while A10 overlaps 2, making A10 the chosen set. This is not optimal, since a set of {A1, A0, A9, A10} has the optimal amount of 4 activities. (this is not the most optimal set since the ties make A0 interchangeable with A2, A5, and A8, all making set sizes of 4). Since A0, A2, A5, and A8 all are part of a subset with larger size than the one including A6, and A0, A2, A5, and A8 all overlap more activities than A6, this proves that the approach of selecting the next compatible activity with fewest overlapping activities does not lead to an optimal solution. Note, there are other examples of sets with sizes 4 not including A6. Even though there are solutions with set sizes of 4 with A6 included, they do not follow the approach of selecting the fewest possible overlaps (A0, A2, or A5 has to be chosen, all with

overlaps of 3 whereas A6 is compatible with overlap size of 2, making A6 chosen instead of A0, A2, A5).

3.
Adjacency Matrix

	1	**2**	**3**	**4**	**5**	**6**
1	0	0	0	0	1	0
2	1	0	1	0	0	0
3	0	1	0	1	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	1
6	1	0	0	1	0	1