## Solve

## Set A

```
def find(self, num_vertices, adj_list):
                                               public int[] find(int numVertices, Node[]
    max_outgoing = -1
                                               adjLis) {
    vertex_with_max = -1
                                                 int maxOutgoing = -1;
                                                 int vertexWithMax = -1;
    for i in range(self.num_vertices):
       count = 0
                                                 for (int i = 0; i < numVertices; i++) {
                                                    int count = 0;
       current = self.adj_list[i]
       while current:
                                                    Node current = adjList[i];
                                                    while (current != null) {
         count += 1
         current = current.next
                                                      count++;
                                                      current = current.next;
       if count > max_outgoing):
         max_outgoing = count
         vertex_with_max = i
                                                    if (count > maxOutgoing) {
                                                      maxOutgoing = count;
                                                      vertexWithMax = i;
    if max_outgoing < 2:
       return -1
                                                   }
                                                 }
    outgoing_weights =
np.zeros(max_outgoing, dtype=int)
                                                 if (maxOutgoing < 2) {
    current = self.adj_list[vertex_with_max]
                                                    return new int[]{};
    index = 0
    while current:
                                                 int[] outgoingWeights = new
                                               int[maxOutgoing];
       if current.weight > 5:
         outgoing_weights[index] =
                                                 Node current = adjList[vertexWithMax];
current.weight
                                                 int index = 0;
         index += 1
       current = current.next
                                                 while (current != null) {
                                                    if (current.weight > 5) {
                                                      outgoingWeights[index] =
    return outgoing_weights
                                               current.weight;
                                                   } else {
                                                      outgoingWeights[index] = 0;
                                                    index++;
                                                    current = current.next;
                                                 return outgoingWeights;
```

## Set B

```
def find(self, num_vertices, adj_list):
                                               public int[] find(int numVertices, Node[]
    min_outgoing = float('inf')
                                               adiList) {
    vertex_with_min = -1
                                                  int minOutgoing = Integer.MAX_VALUE;
                                                  int vertexWithMin = -1;
    for i in range(self.num_vertices):
       count = 0
                                                  for (int i = 0; i < numVertices; i++) {
       current = self.adj_list[i]
                                                    int count = 0;
                                                    Node current = adjList[i];
       while current:
         count += 1
                                                    while (current != null) {
         current = current.next
                                                      count++;
                                                      current = current.next;
       if 0 < count < min_outgoing:
                                                    }
         min_outgoing = count
         vertex_with_min = i
                                                    if (count > 0 && count < minOutgoing)
                                               {
    if vertex_with_min == -1:
                                                      minOutgoing = count;
       return -1
                                                      vertexWithMin = i;
                                                    }
    outgoing_weights =
                                                 }
np.zeros(min_outgoing, dtype=int)
    current = self.adj_list[vertex_with_min]
                                                  if (vertexWithMin == -1) {
    index = 0
                                                    return new int[]{};
                                                 }
    while current:
       if current.weight < 5:
                                                  int[] outgoingWeights = new
                                               int[minOutgoing];
         outgoing_weights[index] =
                                                  Node current = adjList[vertexWithMin];
current.weight
         index += 1
                                                  int index = 0;
       current = current.next
                                                  while (current != null) {
                                                    if (current.weight < 5) {
    return outgoing_weights
                                                      outgoingWeights[index] =
                                               current.weight;
                                                      index++;
                                                    current = current.next;
                                                  return outgoingWeights;
```

## Rubric

Giving correct parameters	1
Finding max/min outgoing edged vertex	5
Returning -1 / int [] {}	2
Initializing result array with proper size	2
Selecting the correct vertex for building the resulting array	1
Building Resulting array	3
Returning proper array	1
Total	15