**Solve**

**Set A**

| def find(self, num\_vertices, adj\_list):  max\_outgoing = -1  vertex\_with\_max = -1  for i in range(self.num\_vertices):  count = 0  current = self.adj\_list[i]  while current:  count += 1  current = current.next    if count > max\_outgoing):  max\_outgoing = count  vertex\_with\_max = i  if max\_outgoing < 2:  return -1  outgoing\_weights = np.zeros(max\_outgoing, dtype=int)  current = self.adj\_list[vertex\_with\_max]  index = 0  while current:  if current.weight > 5:  outgoing\_weights[index] = current.weight  index += 1  current = current.next  return outgoing\_weights | public int[] find(int numVertices, Node[] adjLis) {  int maxOutgoing = -1;  int vertexWithMax = -1;  for (int i = 0; i < numVertices; i++) {  int count = 0;  Node current = adjList[i];  while (current != null) {  count++;  current = current.next;  }  if (count > maxOutgoing) {  maxOutgoing = count;  vertexWithMax = i;  }  }  if (maxOutgoing < 2) {  return new int[]{};  }  int[] outgoingWeights = new int[maxOutgoing];  Node current = adjList[vertexWithMax];  int index = 0;  while (current != null) {  if (current.weight > 5) {  outgoingWeights[index] = current.weight;  } else {  outgoingWeights[index] = 0;  }  index++;  current = current.next;  }  return outgoingWeights;  } |
| --- | --- |

**Set B**

| def find(self, num\_vertices, adj\_list):  min\_outgoing = float('inf')  vertex\_with\_min = -1  for i in range(self.num\_vertices):  count = 0  current = self.adj\_list[i]  while current:  count += 1  current = current.next    if 0 < count < min\_outgoing:  min\_outgoing = count  vertex\_with\_min = i  if vertex\_with\_min == -1:  return -1  outgoing\_weights = np.zeros(min\_outgoing, dtype=int)  current = self.adj\_list[vertex\_with\_min]  index = 0  while current:  if current.weight < 5:  outgoing\_weights[index] = current.weight  index += 1  current = current.next  return outgoing\_weights | public int[] find(int numVertices, Node[] adjList) {  int minOutgoing = Integer.MAX\_VALUE;  int vertexWithMin = -1;  for (int i = 0; i < numVertices; i++) {  int count = 0;  Node current = adjList[i];  while (current != null) {  count++;  current = current.next;  }  if (count > 0 && count < minOutgoing) {  minOutgoing = count;  vertexWithMin = i;  }  }  if (vertexWithMin == -1) {  return new int[]{};  }  int[] outgoingWeights = new int[minOutgoing];  Node current = adjList[vertexWithMin];  int index = 0;  while (current != null) {  if (current.weight < 5) {  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** |