Name: HAVI BOHRA
Roll No.: 210429

Solution (1):
Problem is a variation of bipartite matching problem, with added constraints on distance and station capacity.
Idea: Model problem as a graph with nodes as buses and stations, with edges based on distances between them.
Algorithm:

- Construction of Graph:
  - Construct a bipartite graph with buses on one side and stations on the other.
  - Add two additional vertices: a source and a sink
  - Connect the source to each bus with an edge of capacity 1 (since each bus can only be connected to one station).
  - Connect each station to the sink, setting the capacity of these edges to the capacity-parameter **L** (assuming L<=no. of buses, otherwise take L=no. of buses), (the maximum number of buses a station can handle).
  - Connect each bus to each station for which the distance between them is less than or equal to range-paramete **r** with edges of capacity 1 (connecting a bus).
- Compute the maximum flow in this network from the source to the sink.
- If the maximum flow equals the total number of buses, it means every bus can be connected to a station within the given constraints, and the answer is YES. Otherwise, the answer is NO.

Pseudo-Code:

```
ConnectBus( buses, stations, r, L):
    • Initialize graph G with source node S and sink node T.
    • For each bus i in buses:
        For each station j in stations:
            If distance(i, j) ≤ r:
                Add edge from bus[i] to station[j] in G with capacity 1
    • For each bus i in buses:
        Add edge from source S to bus[i] in G with capacity 1
    • For each station j in stations:
        Add edge from station[j] to sink T in G with capacity L.
    • Apply Ford-Fulkerson algorithm to calculate the max flow
        maxFlow = FordFulkerson(G, S, T)
    • If maxFlow == number of buses, return "YES".
    • Else, return "NO"
```

Time-Complexity:

- *O(n+k+2),* to construct vertices for n buses, k stations, source and sink
- *O(n*k +n+k)* to connect edges between n buses and k stations (max n*k edges), n source-bus edges, k station-sink edges
- To find max flow, in graph with max. E= n*k+n+k edges and V= n+k+2, time complexity is *O (E*n)* , since Ford Fulkerson algorithm has time complexity O(E*maxCapacity), and L<=n. (L is max capacity which is atmost n) . So, total time complexity for max-flow is *O((n*k+n+k)*n)*
- So overall **O((n*k+n+k)*n)**, **polynomial time algorithm**.

Name: HAVI BOHRA
Roll No.: 210429

Solution (2):
Idea: Using hint, the idea is to transform the graph such that vertex-disjoint paths can be treated as edge-disjoint paths.
Algorithm:

- Construction of modified graph G'=(V',E') from G=(V,E):
  - For each vertex v ∈ V , (except s and t),
    - Add two vertices v_{in} and v_{out} in G'
    - Add directed edge (v_{in}, v_{out}) with capacity 1 in G', so that at most 1 path can pass through this vertex.
  - Add vertex s and t in G'
  - For every directed edge (u, v) ∈ E, add directed edge (u_{out} , v_{in}) with capacity 1 in G'
  - (Note: each v_{in} has only incoming edges except outgoing to v_{out} , similarly each v_{out} has only outgoing edges except incoming from v_{in})
- Find the maximum number of edge-disjoint paths from s to t in G'
  - This maximum number of edge-disjoint paths in G' will be the maximum number of vertex-disjoint paths from s to t in G

Pseudo-Code:

```
maxVertexDisjointPaths(G, s, t):
    • Initialize G' = empty graph
    • For each vertex v ∈ G (except s and t):
        Add two vertices v_in and v_out to G'
        Add a directed edge (vin, vout) with capacity 1 to G'
    • Add vertex s and t to G'
    • For each edge (u, v) ∈ G:
        Add a directed edge (u_out, v_in) with capacity 1 to G'
    • Apply Ford-Fulkerson algorithm to calculate the max flow
        maxFlow = FordFulkerson(G', s, t)
    • Return maxFlow
```

Proof of Correctness:

- Suppose two edge-disjoint paths(in G') path1 and path2 has common vertex v
- If v is in-vertex corresponding to a vertex in G:
  - both path has to go from v_in->v_out edge as it is only outgoing edge of v_in

  If v is out-vertex corresponding to a vertex in G:
  - both path has to go from v_in->v_out edge as it is only incoming edge of v_out
- But this contradicts the assumption that these paths are edge-disjoint
- Hence contradiction, v comes in only one of path
- Hence corresponding path in G is vertex-disjoint
- Hence max. number of edge-disjoint paths in G' will be the max. number of vertex-disjoint paths in G

Time-Complexity:

- *O(n+m),* to construct graph G', as n-2 vertices of G(every except s and t) split into two vertices in G', so total 2*(n-2) +2 vertices and #edges increases by n-2, so m+n-2 edges, so O(n+m) time complexity
- *O(n+m)*, for calculating maxFlow in G' using Ford-Fulkerson, since Ford-Fulkerson algorithm has time complexity O((#edges)*maxCapacity), since capacity for every edge is 1, and #edges= m+n-2
- So overall **O(n+m)**, **polynomial time algorithm**.