Homework-8:

(1) Considering k base stations having some load. Let 1st base station be able to bear l_1 load, 2nd → l_2 load, and so on...

> arranging the base stations in non-decreasing order of their loads.

> Suppose l_i is the min. load, then check l_i nearest clients from base station in the range of b_i and attach them to b_i.

> If there a client in range of more than one base stations, then check the loads of those base stations and number of clients attached to them. Attach the client to any base station of ~~its~~ their range whose load isn't saturated.

> It takes $O(k)$ time to sort base stations as per their load; ~~O(l)~~ $O(l)$ time to find clients in the range of a base station; $O(n_l)$ times to check for each client if it exists in the range of more than one base station.

Time complexity: $O(k) + O(k \cdot l) + O(n_l)$

≈ polynomial time.

(2) Let there be n-injured people, k hospitals, capacity of each hospital = n/k, range for injured people to reach hospital = half hour driving time.

> for a hospital k_i, check all the patients at half hour driving time.

> Pick n/k people of them such that

if n/k ≥ total people in range

then pick all people

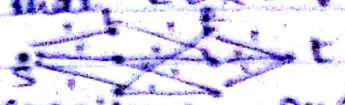
else

remove the people who are in the range of other hospital as well, until n/k is reach

> Repeat this for all hospitals.

Time complexity : $O(n) + O(nk) + O(nk)$
polynomial time.

(3) Let there be two sets L, R such that $L \cup R = V$ and $L \cap R = \phi$
L be on left, R - right :



Let there be a source, sink & capacity = 1, and a cut AB
There can be any number of capacity for the edge between left and right (L & R)
The edges that cross the cut can either incident on source or sink.
Now let P be the union of L - S and R ∩ S
Hence its size is the capacity of the cut AB.
This works true for the converse as well.
G' has a vertex cover of size at most k.

(4) Let there be a source, a sink, vertices u, v.
Let there be a min cut dividing source and sink with capacity set to one on each edge.
For an integer k, there can be a min cut such that there are two or more paths having no edge in common.
So yes, there could exist ~~an~~ edge disjoint paths.
To compute a set of k edge disjoint paths we can make k cuts ~~betw~~ separating the source and sink.
Now these cuts can separate the paths, each path being different from the other. This way it is possible to compute edge disjoint paths.

(5) For an undirected graph with n vertices and m edges. arrange the vertices of the graph in non-decreasing order of the number of edges connected to it.

> The vertex with minimum edges connected gives the final solution.

> The number of edges connected to the vertex that has minimum edges, gives edge connectivity.

> ~~Also on the~~ converse, the So here, for n vertices to sort as per m edges in non-decreasing order it takes $O(m^2n)$ time.

for vertices 1 to n

count edges connected to each vertex

Sort in non-decreasing order of no. of edges

end for.