Name: Jaime Arana-Rochel

Homework #6

Q1: Exercise 6 Chapter 7

Algorithm

```
// function to create a bipartite graph from a floor plan
create_bipartite:
   // we create the two vertex sets
   // a node x_i corresponds to switch i
   Add n switch nodes to set X

   // a node y_j corresponds to fixture j
   Add n fixture nodes to set Y

   // we add edges to set E
   For every x_i in X:
      For every y_j in Y:
         For every wall in the floor plan:
            test whether (x_i, y_j) intersects wall

         If (x_i, y_j) did no intersect any wall:
            add (x_i, y_j) to edge set E

   return graph B(X,Y,E)
end


Ergonomic_Test:
   // we create our bipartite graph
   B = create_bipartite()

   // create graph B' to pass onto Ford-Fulkerson
   B' =
     Connect node s to every node in X from B
     Connect node t to every node in Y from B
     Add capacity 1 to every edge

   max_flow = Ford-Fulkerson(B')

   If max_flow = n:
      return "Ergonomic"
```

```
   else:
      return "not Ergonomic"
end
```

Correctness

The algorithm has two parts. First it builds a bipartite graph $B = (X, Y, E)$. A node $x_i \in X$ corresponds to a switch i, and a node $y_j \in Y$ corresponds to a fixture j. We only add a pairing (x_i, y_i) to the edge set E only if it doesn't intersect any of the walls of the floor plan.

Next, we augment with graph B to graph B' so that we can pass it on to Ford-Fulkerson. This is done in the same way as in the book: adding nodes s, t and having the capacity for each edge be 1. We get a maximum flow from Ford-Fulkerson and then check if it equals n. If it does, then we know that not only did we find a maximum matching in B, but we also found a perfect matching in B. This perfect matching correctly tells us that our floor plan is ergonomic since a perfect matching in B is a pairing of the n switches and fixtures. Since we only added an edge e to E if the switch-fixture pairing wasn't obstructed, this provides us with an ergonomic answer.

Running Time

There are three for loops in the function which creates our bipartite graph. It creates every possible switch-fixture pairing (from n switches and fixtures) and then tests the line segment of this pairing against m walls. Thus the running time of this function is $O(n^2m)$.
$Ergonomic\text{-}Test$ then runs Ford-Fulkerson to get a maximum flow, and then simply tests if the flow equals n. The graph B could at worst have n^2 edges in E, so Ford-Fulkerson runs in $O(n * n^2) = O(n^3)$.
The total running time is then $O(n^3 + n^2m)$.

Q1: Exercise 7 Chapter 7

Algorithm

```
// function to create a flow network graph
create_network:

   // connect client and base station nodes
   For i = 1 to n:
      node u_i = client i
      Add u_i to node set V
      For j = 1 to k:
         node v_j = base station j
         Add v_j to V
```

```
         If the distance from u_i to v_j <= range parameter r:
            add edge (u_i,v_j) to set E with capacity 1

   Add sink node t to V
   Add source node s to V

   Add an edge from s to every client node with capacity 1
   Add an edge from every base station node to t with capacity L

   return G(V,E)
end

Client-Station-Check:
   Graph G = create_network()
   max_flow = Ford-Fulkerson(G)

   If max_flow = n:
      return "All clients connected"
   Else:
      return "Not all clients connected"
end
```

Correctness

The algorithm has two parts. First we build our flow network graph G. For every client i we create a node u_i, and for every base station j, we create a node v_j. We create an edge (u_i, v_j) with capacity 1 if the client i and the base station j are in range of each other. Lastly we connect our source node s to each of the u_i with capacity 1, and we connect each of the v_j to the sink node t with capacity L.

With this graph, we run Ford-Fulkerson and receive a maximum flow. We then make the following claim: We can connect all of the clients to a base station if our maximum s-t flow in G has value equal to n.

Proof: If we can connect every client to a base station, then Ford-Fulkerson will have pushed flow through every s-t path containing each of the (u_i, v_j) edges. Since each edge has capacity 1, then our flow has value n. We do not violate the capacity constraints on the graph at any point. Since the edges connecting the base stations to the sink t have capacity L, Ford-Fulkerson ensures that we don't send flow greater than L through each of these base stations. In other words, we never try to connect clients to a base station such that we violate the base station connection capacity L. Thus, the algorithm correctly tells whether each client can be connected.

Running Time

Creating the flow network graph runs in time $O(nk)$ time due to the two for loops. Then, our final running time is simply the time to execute Ford-Fulkerson on graph G. With $O(n+k)$ nodes and $O(nk)$ edges, that is a running time of $O((n+k)*nk) = O(n^2k + nk^2)$.