Problem 1:

a) Algorithm

1) For every day you check the nearest hotels that are on the way to your destination.
2) If a hotels' distance from the current starting point is greater than d, go to the last hotel whose distance was less than distance d.
3) Repeat steps 1 and 2 until you arrive at your destination.

b) Running Time

The theoretical running time of this algorithm is O(n) because the algorithm checks available hotels once.

Problem 2:

With the backwards activity selection algorithm, it is the same as the activity selection algorithm that is running forwards through time. It is still making the same greedy choice at every possible point. Since the only change between the two algorithms is whether they run forwards or backwards through time, the backwards running algorithm should also find the optimal solution because the forward algorithm also found the optimal solution.

Therefore, this backwards algorithm is a greedy algorithm because it makes the greedy choice at every step regardless if it is going backwards or forwards.

Problem 3:

include description of the algorithm, pseudocode, and analysis of running time

An algorithm that can find the optimal solution would be:

```
last_to_start ( s, f ):
        n = length of s
        A = [ a_n ]
        k = n
        while ( m = n - 1 and m > 1 ) :
                if ( f [m] <= s [k] ) :
                        A = A U A_m ( add A_m to activity )
                        k = m
        return A
```

assuming that:

S = [a_1, a_2, … a_n]

Each a_i is equal to [start_i, finish_i)

The algorithm will take an input of A and evaluate its start and finish times. The it looks through each activity in descending order (ie. last to first) and finds an activity to add to A. The only changes

Instead of a while loop, the program reverses the passed in array and loops through it. It then compares the finish time of the current object to the previous start time by checking the grouped element's indexes. If an activity is found it gets appended to the array of selected activities which gets returned.

The run time of the algorithm should be O(n logn). The algorithm should pass through the array once because it checks at every step and makes the greedy choice at every step.