Pablo Acuna
CSCI 4020

Computer Algorithms Homework 1

3.4) Answer: To model the problem, we will use an undirected weighted graph. Where edge weights of 1 corresponds to same, an edge weight of -1 corresponds to differrent, and no edge corresponds to ambiguous. To edit BFS we will have an array of size n to store each nodes' previous node. And another array of size n that stores whether a node is either spieces 1 or 2. If the interger for the node is odd then it is spieces 1, if it is even then it is spieces We will choose a start node to be spieces 1 and its previous to be null. We will run the same procedure as BFS as in we will edit the speices array if it has not yet been visited. The only difference is we are also checking connected visited nodes to see if it is does not voliate the matching. If it does we return not possible. If the algorithm traverses through the whole graph and does not detect an error in the pairing, we will return valid matching. This is O(m+n) because we did not add anything in the loop that is more than a constant operation. Making the run time the same as BFS O(m+n).

Modified BFS:
```
for all u $\in$ V:
    species(u) = $\infty$
    prev(u) = null
species(s) = 1
Q = [s] (queue containing just s)
while Q is not empty:
    u = eject(Q)
    for all edges (u,v) E:
        if (species(v) $\neq \infty$ & prev(u) $\neq$ v )
            if ( $l_{u,v}$ = -1 & spieces[u] % 2 = species[v] % 2): return not valid
            if ( $l_{u,v}$ = +1 & spieces[u] % 2 $\neq$ species[v] % 2): return not valid
        else if(spieces(v) = $\infty$)
            inject(Q, v)
            prev(v) = u
            if($l_{u,v}$ = -1)
                spieces(v) = spieces(u) + 1
            if($l_{u,v}$ = 1)
                spieces(v) = spieces(u)
return valid
```

4.7) Answer: The Greedy approach we will use for this problem is by choosing the jobs with the largest finish time f_i first. Therefore, we will sort the jobs by f_i in decreasing order. And choosing the largest finish time availible next. This algorithm is O(nlogn). To prove this is correct we will use the excahnge argument. Say we have an OPT solution where we have at least two jobs (j_k, j_l) that are not in order of largest finish time ($j_k < j_l$ & $f_k < f_l$). Therefore the time it takes to do these jobs are $t_{OPT} = p_k + p_l + f_l$. This is because the finish time of job l is bigger. Swapping the two leads to $t = \max (p_l + f_l, p_l + p_k + f_k)$. Both of these sums in the max function are less than the original oprimal. Therefore contradiction and we have proved the optimal scheduling is largest finish time first.