Class Test 3
Solution Sketch for Questions 5 and 6

Q5. (a)

```
Let A and B be two hash tables of size 2n each, initialized to 0
Let h be a random hash function.
for each number x of person 1
    A[h(x)] = A[h(x)] + 1
for each number y of person 2
    B[h(y)] = B[h(y)] + 1
for i = 1 to 2n
    If A[i] ≠ B[i] return NO
return YES
```

[Can also do with one hash table with increment for person 1 and decrement for person 2. Check for all entries = 0 at the end]

(b)

If the two sets of numbers are different, there exists at least one number p chosen by person 1 such that p is chosen a different number of times by person 2 (including 0 times when person 2 has not chosen p). Suppose h(p) = q. Since the answer is YES, A[q] = B[q]. Since p is chosen a different number of times by person 2, there must exist at least one number $r \neq p$ that hashes to the same index q to make A[q] = B[q]. Now, the expected number of numbers hashing to any one entry of the table is n/2n = 1/2. Thus, the probability that a number different than p hashes to the same index as p is less than 1/2. Hence, with probability at least 1/2, no other number hashes to the same index as p. In that case, there is no chance of A[q] = B[q], as the difference in the number of occurrences of p in the two sets cannot be cancelled out by any other number. Hence, the difference will be detected with probability at least 1/2. Hence the probability of a wrong YES answer (the difference is not detected) is at most 1/2.

[Note: Most of you have used a standard bloom filter, or a modified bloom filter which stores integers instead of 1 and hashed bits are incremented instead of just set to 1. A standard bloom filter is not good as it cannot consider counts at all, so you have got a 6 out of 8 in part (a). In the second case, I have given 8 out of 8 in part (a) though you do not really need a bloom filter type of data structure. However, in both cases, most of you have got a 0 in part (b) as you used the standard analysis of bloom filter false positive probability. It is not accepted because the definition of false positive itself is different here, and for the second case, it is also not a standard bloom filter. So you cannot use any result from class directly (won't be directly applicable anyway) and need to prove the error probability yourself.]

Q6. (a)

Solution structure: $<x = x_1, x_2, x_3, \ldots\ldots x_{n-1}, x_n = y>$, where each x_i is a city in the path

(b)

Suppose you are applying the bounding function to a node corresponding to the partial solution $s_i = <x_1, x_2, \ldots., x_i, \ldots.>$

Pruning infeasible solutions:

Prune if x_i is a city already in the path, or if $x_\mathrm{i} = y$ but all other cities are not covered : *(if $x_i = x_j$, $1 \leq j \leq i$-1) OR (if $x_i = y$ and $i \neq n$) then return false*

Pruning feasible but non-optimal solutions:

if $F(s_i) + v_{lower}(i+1) > v_{upper}$ then return false // (as seen in class)

where

$F(s_i) = \max\{\ w(x_j, x_{j+1}) | \ 1 \leq j \leq i\text{-}1\}$ // the maximum weighted edge in the partial path seen so far

Let $W = \min\{w(x_i, x_j) \mid x_j$ not in $\{x_1, x_2, \ldots., x_i\}\}$ // minimum weight edge from x_i to nodes not in the partial path

$v_{lower}(\mathrm{i}+1) = W - F(s_i)$ if $W > F(s_i)$ // this will be minimum increase in max weight edge

= 0 otherwise // current max weight edge stays as max weight

v_{upper} = max weight of any edge in the graph initially. As each full solution S is found, update v_{upper} as the minimum of current value of v_{upper} and the weight of the maximum weight edge in S

Other variations possible but mainly for $v_{lower}(\mathrm{i}+1)$, have been given credit.

(c) Not shown, easy to compute. Anyway varies with your choice of functions. Not given much credit if your bounding function is too bad anyway, then it hardly prunes anything which gives a trivial tree.

Not many people attempted Q6 anyway.