Q1:

a) s -> h -> k -> c -> a -> b -> d -> m -> e -> n -> g

b)

i) The definition for admissibility is 0 <= h(state) <= h(state)* where h(state)* is the optimal route.

Prove by contradiction: assume h(state) >= h(state)*

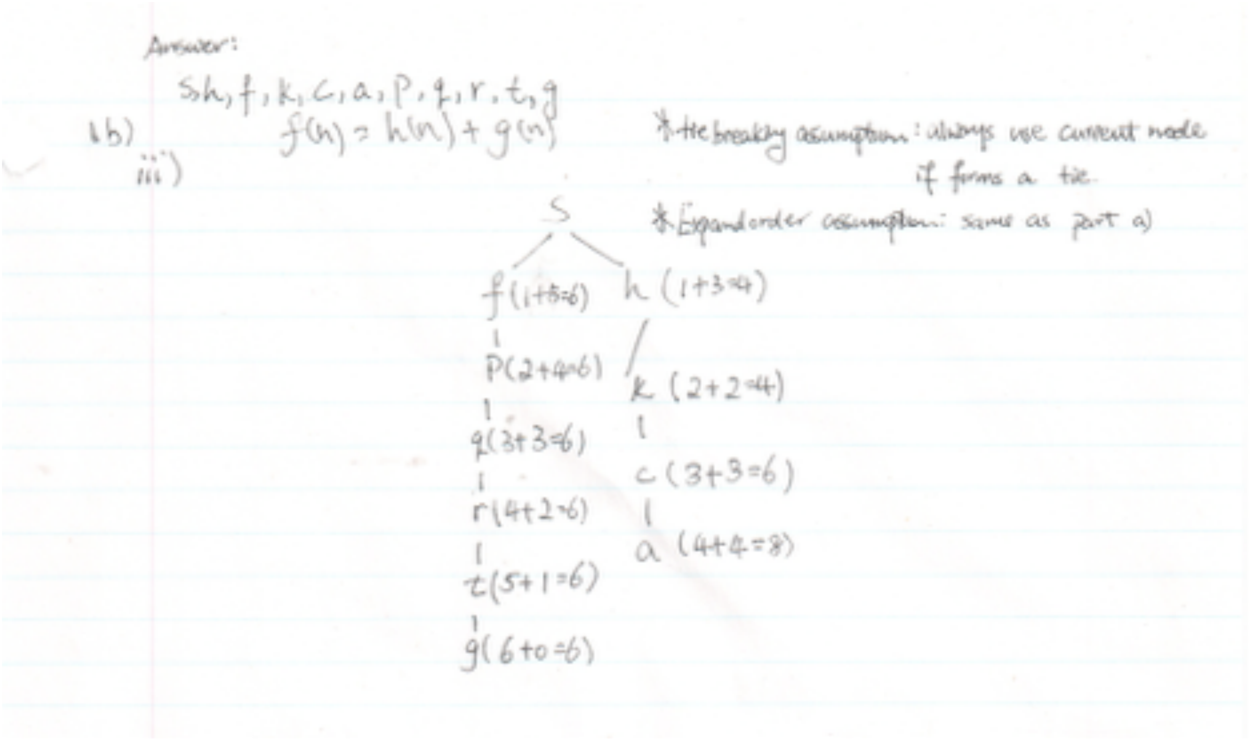
then h(s) >= h(s)*

4 >= 6 (contradiction)

Since h(state) does not take account of the barrier exception, where h(state)* need to go around the barrier in order to reach the goal state, h(state) must be admissible.

ii) s -> h -> k -> c -> a -> b -> d -> m -> g

iii)



Answer:

S,h, f, k, c, a, p, q, r, t, g

1b) $f(n) = h(n) + g(n)$

iii)

* tie breaking assumption: always use current node if forms a tie.

* Expand order assumption: same as part a)

Q2:

a) Breadth-first search is a special case for uniform-cost search.

The difference between BFS and UCS is the order they expand nodes. BFS is depends on depth where UCS expands based on cheapest cost. Therefore, if all step costs are equal, g(n) would be a multiple of depth n.

g(n) = C * (depth of n) where C is a constant among all nodes

Thus, UCS reproduces BFS.

b) DFS is a special case for best-first search.

DFS expands current path until leaf node, where depth(n) increases. Best-first search expands the lowest cost according to heuristic, where heuristic is the cost from current node to the goal node. Therefore, DFS is best-first search with f(n) = -depth(n).

c) Uniform-cost search is a special case for A* search.

A* search is based on $f(n) = g(n) + h(n)$. Uniform-cost search is based on $f(n) = g(n)$. Thus, Uniform-cost search is A* search with $h(n) = 0$.

Q3:

a)

States: list of cities that are represented by (x,y) coordinates

State representation: Nodes represent cities and edges represent the cost between cities

Initial state: Salesman has not visited any other city and currently in the start city (A)

Goal state: Salesman has visited all cities and back to the start city (A)

Operators: Update the current route based on all the unvisited cities

Operator cost: Euclidean distance between city nodes

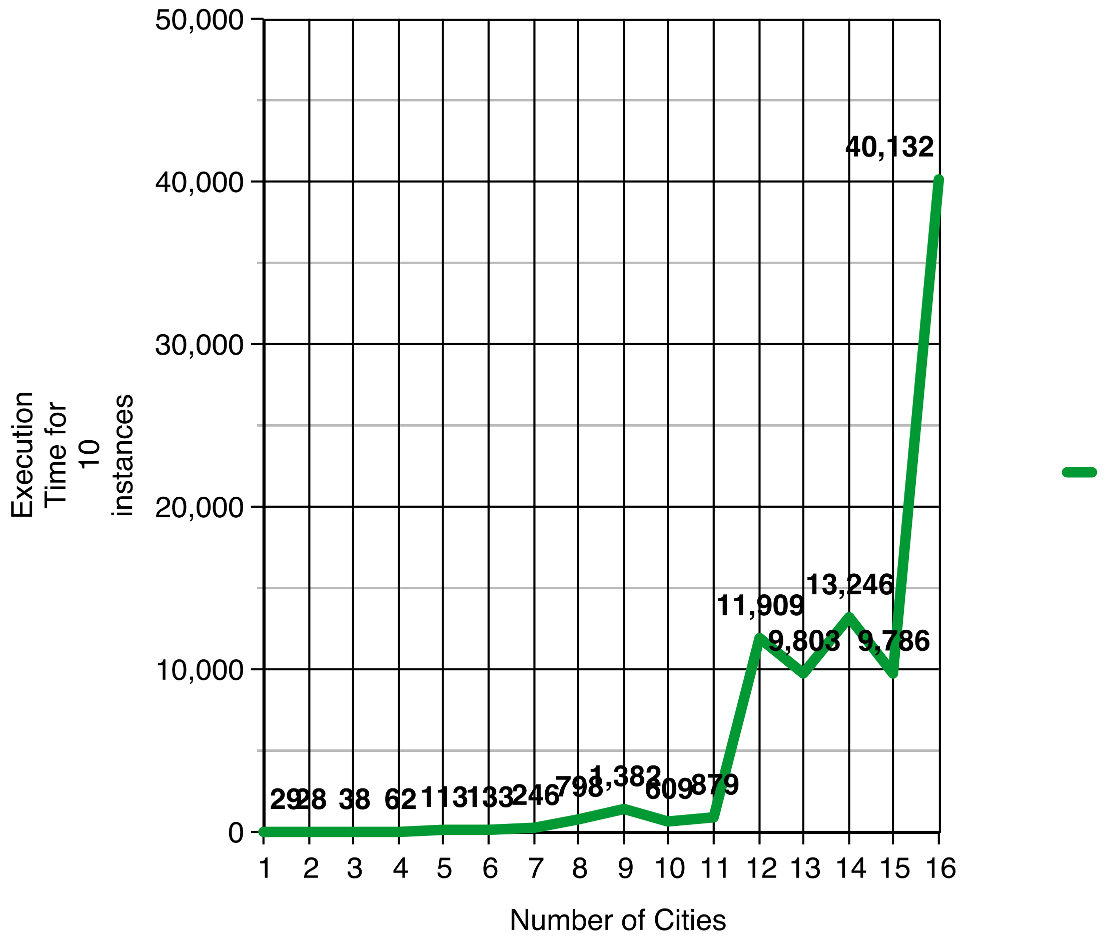
b)

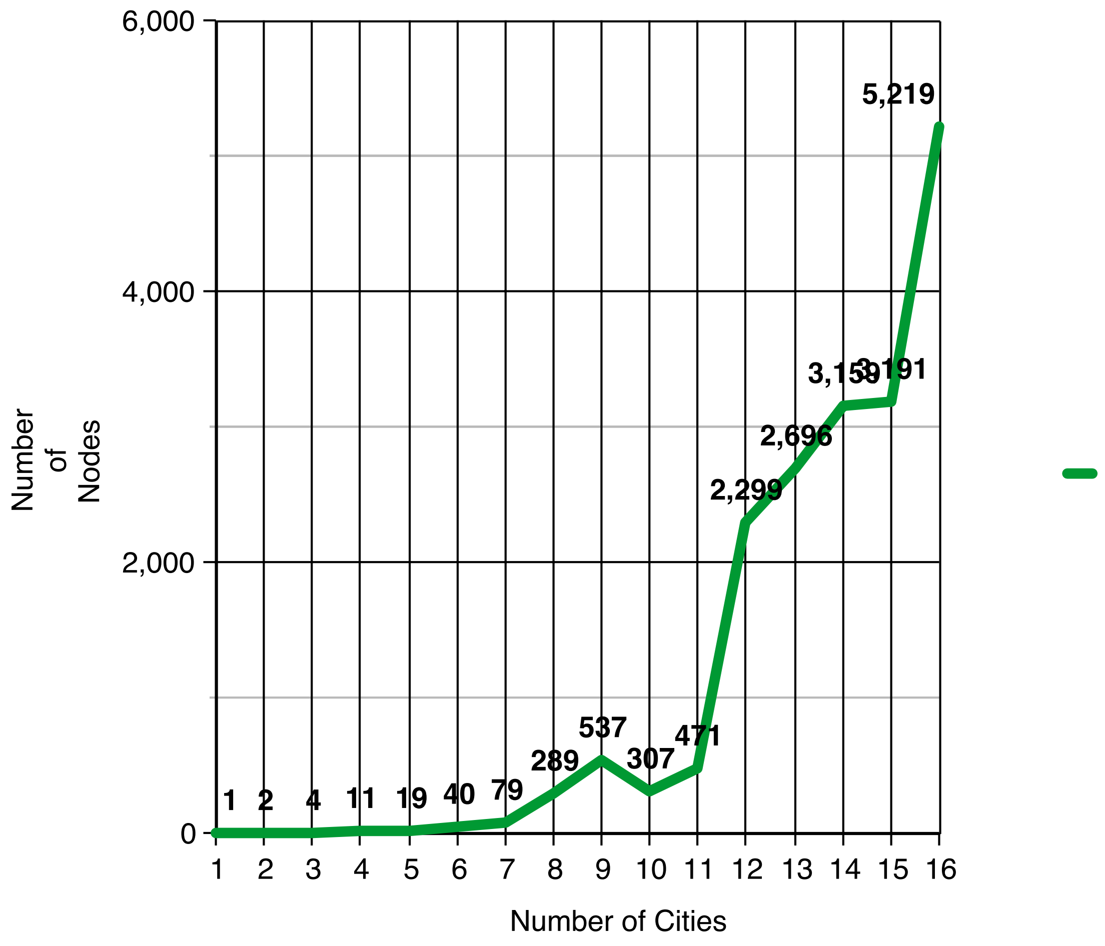
h(n) = total cost of minimum spanning tree of unvisited cities plus the starting node

h(a(1) -> a(k)) = MST cost of a(2) -> a(k-1)

This is admissible because MST is the minimum cost for traveling all nodes without a cycle. Since traveling sales man need to form a cycle in order to travel all nodes and goes back to the start node, MST must be admissible (less cost than TSP completion).

c)

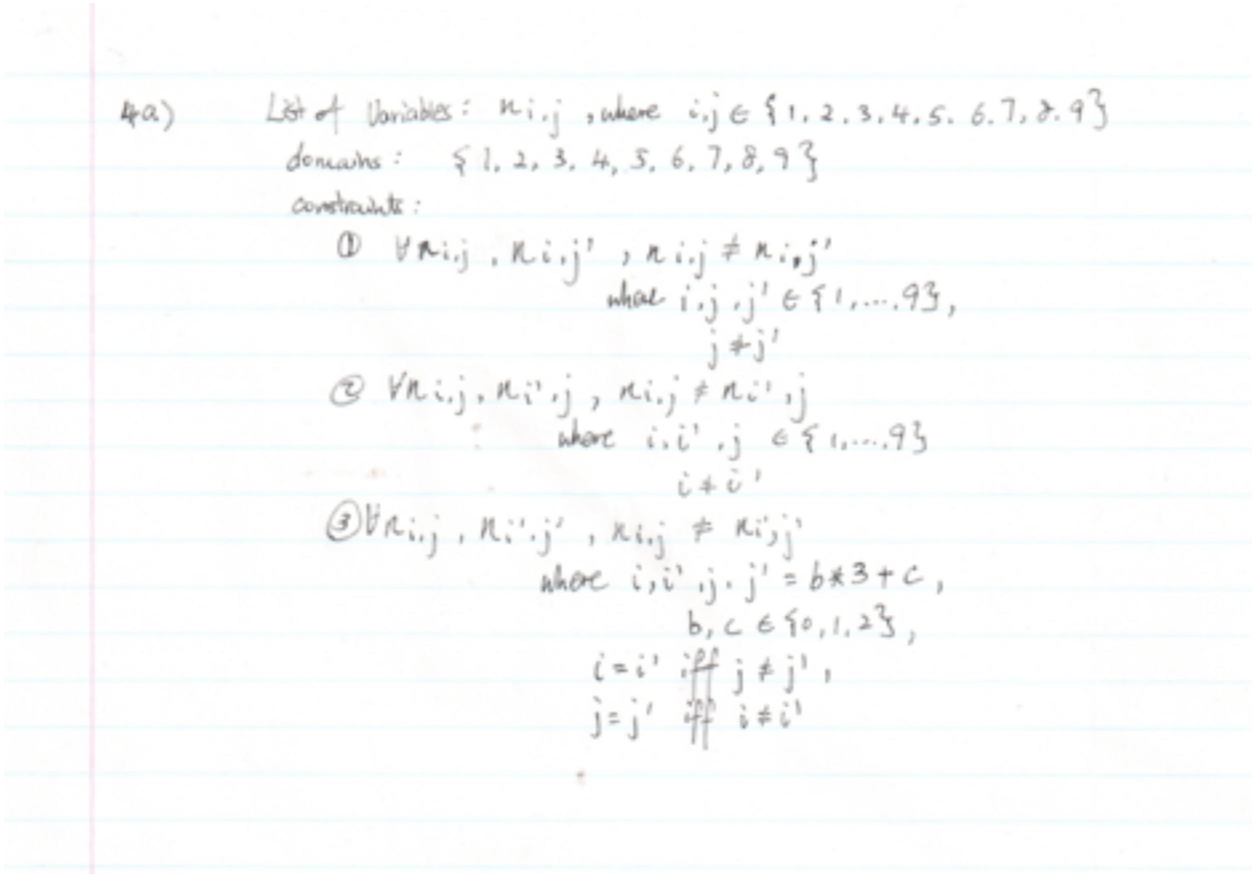




According to the graph, the growth of number of nodes seems to be exponential as number of cities increases. On a 36-city problem instance, the number of nodes should be around 15000 and the execution time would be around 3~4 minutes per instance. When I run my program against one instance of 36-city problem set, the number of nodes are way less than expected, which generates around 5000 nodes. The execution time is quite accurate which was timed to be about 4 minutes.

Q4:

a)



4a) List of Variables: $x_{i,j}$, where $i,j \in \{1,2,3,4,5,6,7,8,9\}$

domains: $\{1,2,3,4,5,6,7,8,9\}$

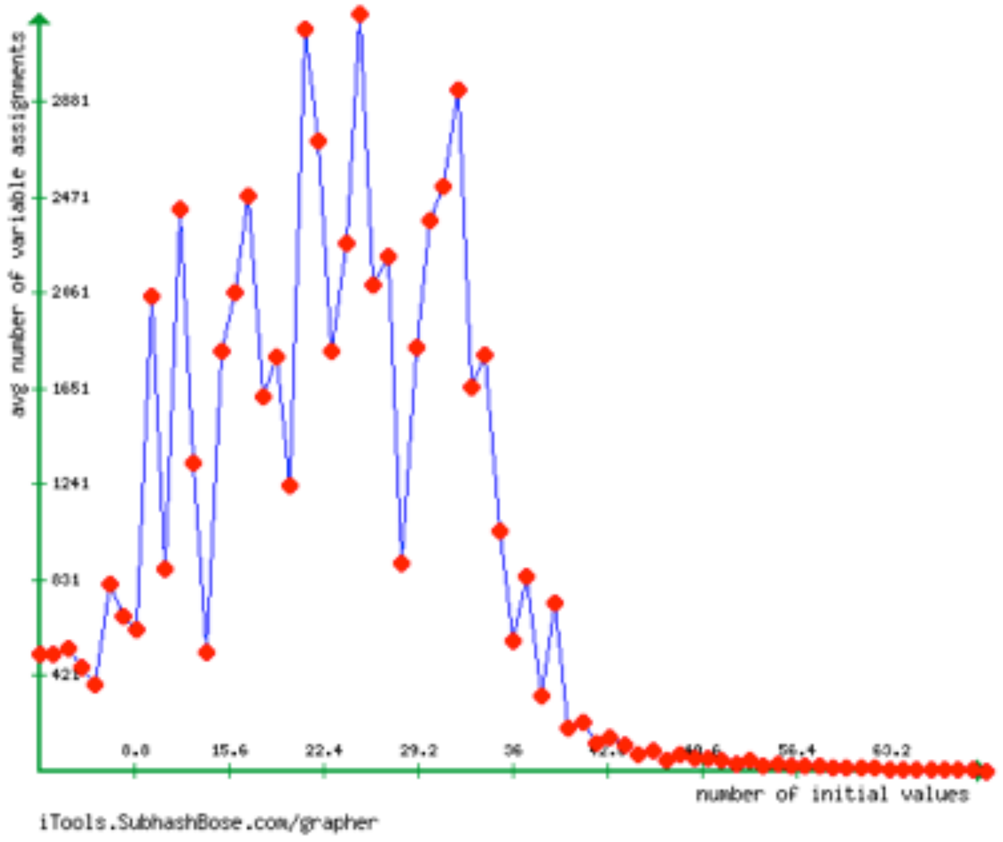
constraints:

① $\forall x_{i,j}, x_{i,j'}$, $x_{i,j} \neq x_{i,j'}$
where $i, j, j' \in \{1, \dots, 9\}$,
$j \neq j'$

② $\forall x_{i,j}, x_{i',j}$, $x_{i,j} \neq x_{i',j}$
where $i, i', j \in \{1, \dots, 9\}$
$i \neq i'$

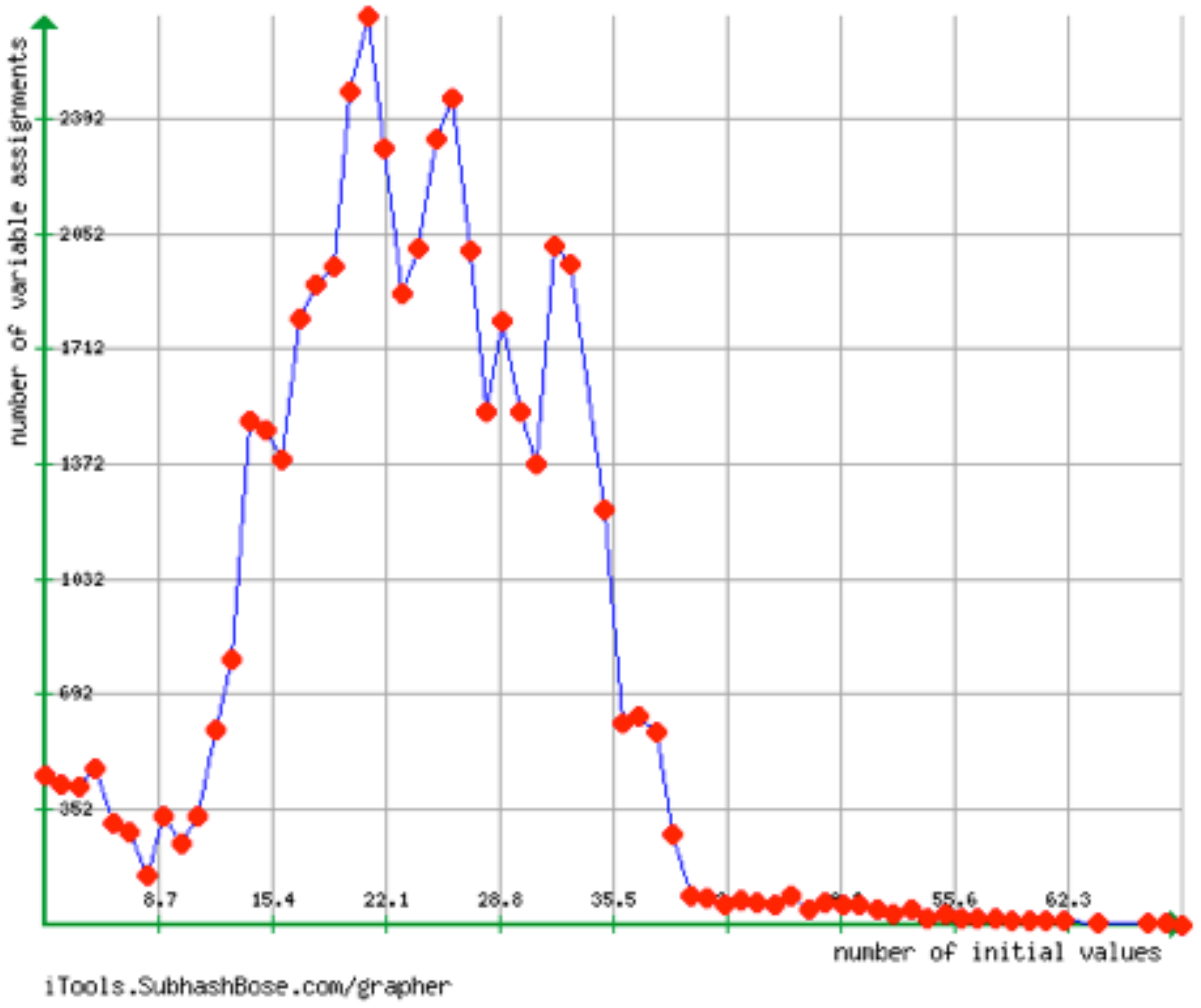
③ $\forall x_{i,j}, x_{i',j'}$, $x_{i,j} \neq x_{i',j'}$
where $i, i', j, j' = b*3+c$,
$b, c \in \{0,1,2\}$,
$i = i'$ iff $j \neq j'$,
$j = j'$ iff $i \neq i'$

c)
Version A plot:



The number of variable assignments are initially low because there aren't many constraints over the value to assign to the grids. Since the initial value in the grids are very little, it can have many possible solutions. Therefore not many backtracking is needed. The number of variable assignments increases as the number of initial value increases until it reaches an equilibrium, which in my case, is around 20~28 number of initial values. Backtracking is done frequently at this level where the constraints and initial values together increases difficulty of this problem significantly. The number of variable assignment drops down as the number of initial values reached past the equilibrium. Since we have so many initial values, not many variable assignment options are left for the program to try. If there are no conflicts of the constraint, there are very likely no backtracking is needed.

Version B plot:



The plot behaves similarly against Version A. However, there are a significant decrease of number of variable assignments for this version with forward checking. This is due to the fact that each node have more knowledge on what are the real available domains, which eliminates many of the backtracking step used for unavailable domains in version A.