1. In the Best-First search with Branch-and-Bound (chapter 6) approach:

- does not limit us to any particular search pattern in the state-space tree
- calculates a "bound" estimate for each node that indicates the "best" possible solution that could be obtained from any node in the subtree rooted at that node, i.e., how "promising" following that node might be
- expands the most promising node first by visiting its children

*Traveling Salesperson Problem* (TSP) -- Find an optimal (i.e., minimum length) tour when at least one tour exists. A tour (or Hamiltonian circuit) is a path from a vertex back to itself that passes through each of the other vertices exactly once. (Since a tour visits every vertex, it does not matter where you start.)

a) For this graph, what are the optimal tour and its length?

To solve a problem using branch-and-bound or backtracking, you need to answer the following questions:

b) What should the state-space tree look like? (i.e., What would the "for each child c" loop iterate over?)

c) What state information is needed at each node?
d) How can we determine a bound to guide our best-first search of the state-space tree?  Hints: We want:
- a lower-bound to allow pruning in a minimization problem like TSP
- consider what your greedy TSP algorithm used for its greedy criteria

e) What information is needed by our bound calculation function?


```c
void bread-first-branch-and-bound( tree T, number & best) {
    priorityQueue Q;
    treeNode u, v;

    initialize(Q)
    v = root of T
    enqueue(Q, v)
    best = value(v)

    while (not empty(Q)) do
        dequeue(Q, v)
        for (each child u of v) do
            if value of (u) is better than best solution
                best = value(u)          # remember as the best solution
            end if
            if bound(u) is better than best then
                enqueue(Q, u)           # u could lead to a better solution
            end if
        end for
    end while
}
```