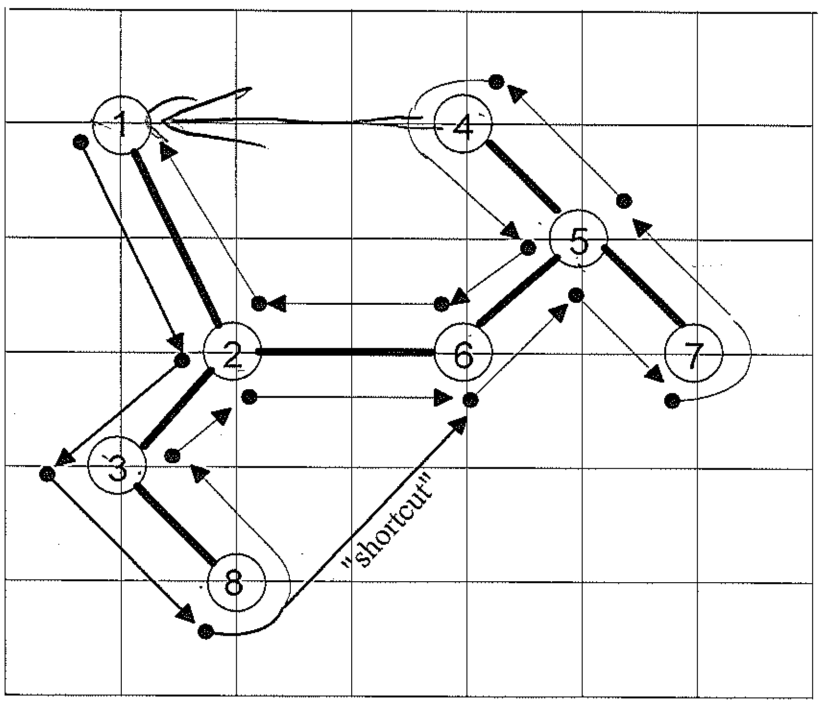
 Name:____________

1. Complete a tour by removing vertices from the path in step 2 by taking shortcuts.

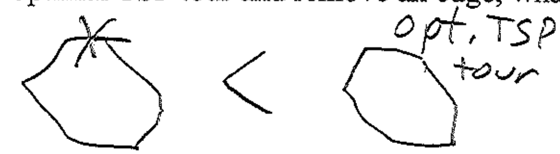


a) Finish removing vertices from the preorder-walk path to create a tour by taking shortcuts:
[1 2 3 8 ~~3~~ ~~2~~ 6 5 7 ~~5~~ 4 ~~5~~ ~~6~~ ~~2~~ 1]

b) When scanning the above path, how did you know which vertices to eliminate to take a shortcut?

seen vertex before

c) If we take the optimal TSP tour and remove an edge, what do we have?



MST ≤ spanning tree < opt. TSP tour

d) What is the relationship between the distance of the MST and the optimal TSP tour?

2 MST < 2 opt TSP tour

e) What is the relationship between the distance of the MST and the distance of the preorder-walk of the MST?

preorder walk distance = 2MST.

f) What is the relationship between the distance of the preorder-walk of the MST and the tour obtained from the preorder-walk of the MST?

tour from preorder walk ≤ preorder walk distance = 2MST < 2 opt TSP tour

g) What is the relationship between the tour obtained from the preorder-walk of the MST and the optimal TSP tour?

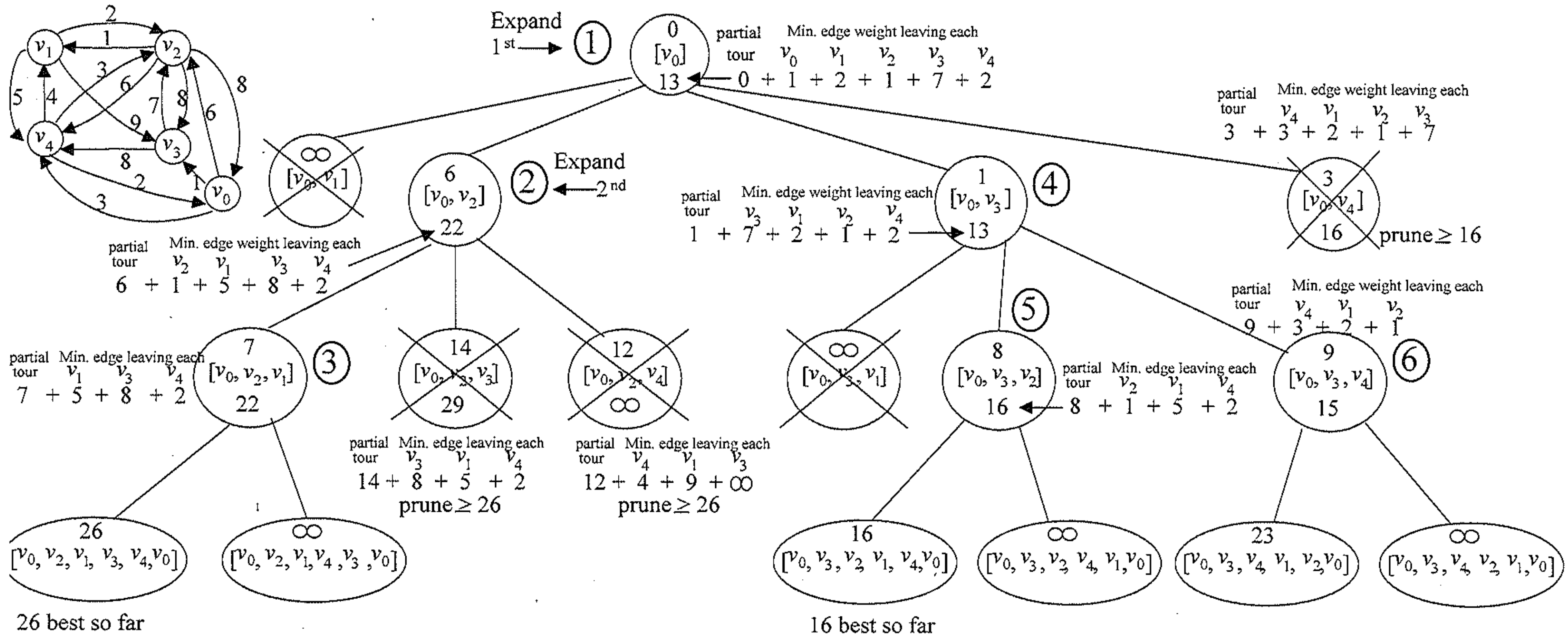
tour from preorder walk < 2 opt. TSP tour

Handling "Hard" Problems: For many optimization problems (e.g., TSP, knapsack, job-scheduling), the best known algorithms have run-time's that grow exponentially ($O(2^n)$ or worse). Thus, you could wait centuries for the solution of all but the smallest problems!

Backtracking general idea: (Recall the coin-change problem from lectures 10 and 13)

- Search the "*state-space tree*" using depth-first search to find a suboptimal solution quickly
- Use the best solution found so far to prune partial solutions that are not "promising,", i.e., cannot lead to a better solution than one already found.

2. To prune a node in the search-tree, we need to be certain that it cannot lead to the best solution. We can calculate a "bound" on the best solution possible from a node (e.g., say node with partial tour: $[v_0, v_4, v_1]$) by summing the partial tour with the minimum edges leaving the remaining nodes. Complete the backtracking state-space tree with pruning.



 Name:________________

3. In the *Best-First search with Branch-and-Bound* 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 ("best") node first by visiting its children

a) What type of data structure would we use to find the most promising node to expand next?

b) Complete the best-first search with branch-and-bound state-space tree with pruning. Indicate the order of nodes expanded.