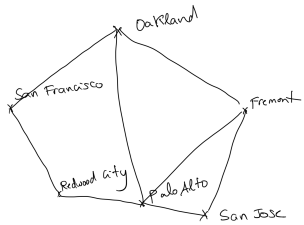
CS 5100 01/09

* Search problems

ex: * Route finding problems
* games (chess)

Defining a problem (formulating a problem)

x Napa Valley



Start: San Jose

Destination: San Francisco

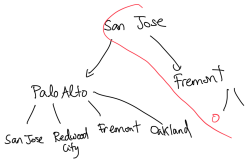
Actions: From San Jose, I can go west or north

Transition Model: Depending on the state I am in and the action taken, I will end up in a new state

Cost Function: quantify how much it will cost me to transition to a new state

* Initial state
* Actions
* Transition model
* Cost function
* Goal test

Let's search for a solution




```
def tree_search (start_node):
    Frontier = { start_node }
    loop if len(frontier) == 0:
             | return "failure"
         choose a node from the frontier
          (and remove it from frontier
         if the chosen node is the goal node:
             | return "YAYY"
         else:
             | expand further
```

* Assume we have a function which returns true if the node is the goal node and False otherwise

* Frontier will contain all the expanded nodes

Main issue with the tree search algorithm are loops which can cause us not to find the goal node

⇒ Solution: keep tracks of the nodes we have already explored

```
def graph_search (start_node):
    frontier = [start_node]
    explored = { }
    loop:
        if len(frontier) == 0:
            return "Failure"
        choose a node from the frontier and remove it from frontier
        as long as we haven't explored it yet
        if the chosen node is the goal node:
            return "YAY!"
        else:
            expand frontier and only add the nodes that were not previously explored nor were in frontier
```

Criteria to assess a search algorithm

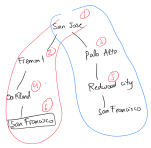
- Am I guaranteed to find a solution (assuming it exists)? (Completeness)
- Is the solution found the optimal one? (Optimality)
- How long will it take the algorithm to find the solution? (Time complexity)
- How much memory will we need to find the solution? (Space complexity)

Looking at the algorithm we have written, we have a choice in terms of which node to select from the frontier.

* Breadth First Search

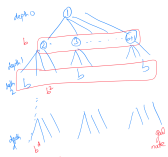
Expanding the nodes going layer by layer

* Completeness? Yes, as long as there is a solution exists, breadth first will keep on expanding up until it finds the goal node.
* Optimality? Does not have to be optimal depending on the order of how the nodes were expanded



* Exercise: Compute time and space complexity of BFS

Let's assume that every node expands to b nodes and assume the goal node is at depth d



Time Complexity

$1 + b + b^2 + \cdots + b^d \in O(b^d)$

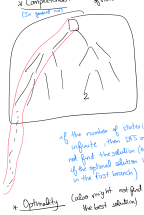
Space complexity $O(b^d)$

explored set will be storing $O(b^d)$ nodes

* Depth First Search

explore a branch till the end

* Completeness: If there is finite number of states and $b \neq \infty$



If the number of states is infinite then DFS might not find the solution (especially if the optimal solution is not in the first branch)

* Optimality (also might not find the best solution)

* Time Complexity

Assume that the number of states is finite and lets the max length is m)

worst case scenario: goal node is in the last branch explored

$O(b^m)$

* Space Complexity $O(m)$