1 State Space

1.1 Counting State Space

If you have N nodes in a graph, and you are representing your state based only on where you are, how many different states are there?

Answer: N

What about if you are representing states as only visited or not visited. Now how many states are there?

Answer: 2^N

Lastly, what about if you combine the two and represent state based on where you are and what you have visited?

Answer: $N2^{N-1}$ because including where you currently are in visited is redundant. $N2^N$ would also be acceptable if this optimization is note made. N tracks where you currently are, and $2^{O(n)}$ tracks where you have visited.

How does it change if track the number of visits to each node and the number of visits is bounded to $M-1$?

Answer: NM^{N-1} because including where you currently are in visited is redundant. NM^N would also be acceptable if this optimization is note made as before. Now, each visited entry has M possible values, 0 to $M-1$, rather than just 2.

1.2 Successor Function

Talk about what the abstraction of a successor function is.

If S is your state, S' is the set of possible next states, and F is your successor function, then:

$$S' = F(S)$$

Why is this a useful abstraction for programming? Allows us to abstract the derivation of next states from our current state, and we can reuse serach algorithms by just providing new successor functions. And so on.

2 Search Algorithms

2.1 Terminology

- Fringe - set of nodes generated, but not expanded
- Goal Function - checks if a node is the goal
- Successor Function - defines possible next states from current state (see above section)

Using this, we can define a generic search function as:

Initialize the fringe with the start state. While the fringe isn't empty, pick a state off of the fringe. Use the goal function to check if the state is a goal. If it is then you are done: SUCCESS. If not, then use the successor function to generate all next states S', and add those to the fringe. Repeat until success or the fringe is empty, FAIL.

2.2 BFS

Explain what breadth first search is, and how it uses a first in first out data structure. ****Draw tree search examples****.

Properties:

- Complete: if there is a solution, BFS will find it
- Finds the shallowest solution in the tree (not always the lowest cost with non uniform edge weights)

Runtime

With branching factor b (maximum number of successors), and a max depth d, runtime and space are $O(b^d)$.

2.3 DFS

Explain what DFS is and how it uses a last in first out data structure. ****Draw tree search examples****.

Properties:

- Not complete: could get caught in cycles and never escape
- No guarantees about what depth a solution will be found at

With branching factor b (maximum number of successors), and a depth d, runtime is unbounded due to lack of completeness and space is $O(bd)$. With no cycles or visited state detection to prevent visiting cycles, runtime is $O(b^d)$.