Midterm Review

Winter 2020

General Information

All information is on CSC384 web page **Test** tab at the top of the page.

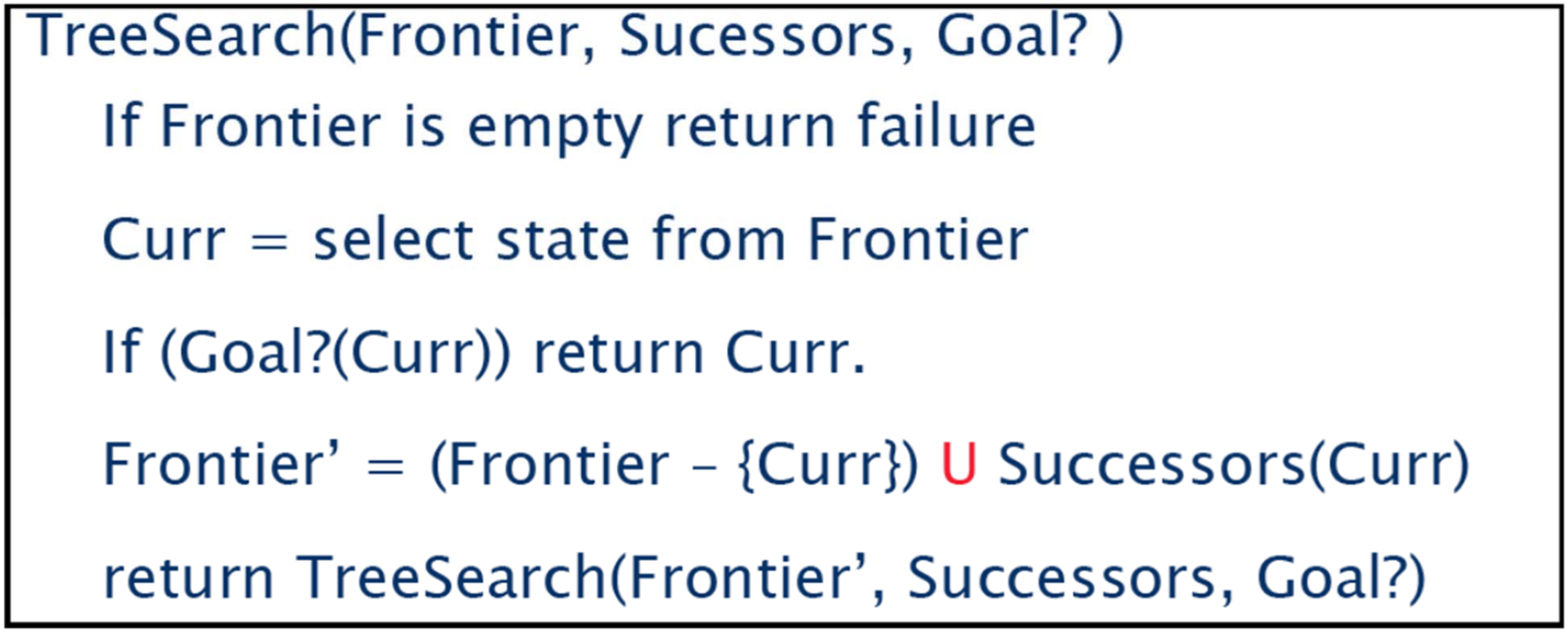
- **50 minutes** in duration
- **No aids** permitted
- Worth **16%** of your course grade
- Times and Location: Please **check the course webpage**.
- Topics:
    - Uninformed and Heuristic Search
    - Constraint Satisfaction Problems

Review

Search

- Understand Search basics
    - Ingredients: State space, initial state, goal condition, successors, cost function, heuristic
    - What problems are easy to formalize, which are not?
    - Search tree (and its key features), search graph, the default search “template”
    - What makes a good search? Optimality, Completeness, Time and Space Complexity

Review - Search Template



```
TreeSearch(Frontier, Sucessors, Goal? )
    If Frontier is empty return failure
    Curr = select state from Frontier
    If (Goal?(Curr)) return Curr.
    Frontier’ = (Frontier – {Curr}) U Successors(Curr)
    return TreeSearch(Frontier’, Successors, Goal?)
```

Review

Search

- Uninformed search
  - BFS, DFS, Depth Limited Search, IDS
  - Remember complexity of each, be able to compare and contrast
  - Uniform Cost Search and its properties. Why/when is it optimal?
  - Know the difference between path checking and cycle checking. When to apply one or the other?

Review

Search

- Heuristic Search Basics
    - What is a heuristic? How can it help us search? How can we come up with a good one?
    - Understand admissibility, consistency (or monotonicity)
- Heuristic Search Strategies
    - Greedy Search
    - A* Search
    - Weighted A* Search
    - IDA* Search
    - Be able to compare and contrast, and to consider various properties of each search, e.g. optimality, completeness, space and time complexity.

Review

CSPs and backtracking search algorithms for solving them

- Understand the CSP representation
  - Variables, constraints, constraint tables and graphs.
  - How a problems can be **represented** in the CSP formalism.
- Backtracking search
  - How it works.
  - How constraint propagation works in **Forward Checking** and know how to enforce it on a set of constraints.
  - How constraint propagation works in **GAC** and know how to enforce GAC on a set of constraints.
  - How **Degree Heuristic** and **MRV** can be used to improve the efficiency of search.

Review

CSP Representation

- A (finite) **domain** must be defined for each variable.
- **Constraints** are defined over variables.
- A constraint is an **expression** over the variables in its scope (could be a mathematical expression, a logical expression, etc.)

Review

CSP Search Tree

- **Root:** Empty Assignment.
- **Children of a node:** All possible **value assignments** for a particular unassigned variable.
- The tree **stops** descending if an assignment **violates a constraint**.
- Goal Node: The assignment is complete and No constraint is violated.

Backtracking Search

- If **all variables** are assigned to a value, we have a **solution**.
- Otherwise:
    - Pick an **unassigned** variable V and assign it a value.
    - Check **all constraints** that **include V** in their scope and all of their variables are assigned.
        - If a constraint is unsatisfied, backtrack and try other values for V.
        - Otherwise, go **one level deeper** into the search tree (by **assigning** values to another **uninstantiated variable**)

Review - Forward Checking

- If **all variables** are assigned to a value, we have a **solution**.
- Otherwise:
    - Pick an **unassigned** variable *V* and assign it a value *d*.
    - Do the followings for each **constraint** *C* over *V* such that *C* has only **one unassigned variable** *X* in its scope (e.g., *C(V, X, Y)* and both *V* and *Y* are instantiated):
        - for each member t of CurDom(X):
            - If *X*=*t* (together with previous assignments to variables in the scope of *C*) **falsifies** *C*, **remove** *t* from CurDom(X).
            - If CurDom(X) becomes **empty**:
                - **Stop** checking the constraints.
                - **Undo** all the **pruning** caused by assigning *d* to *V,* try **another value** for *V*.
    - If all constraints are ok, go **one level deeper** into the search tree (by **assigning** values to another **uninstantiated variable**).

Review - GAC

- If **all variables** are assigned to a value, we have a **solution.**
- Otherwise:
    - Pick an **unassigned** variable *V* and assign it a value *d*.
    - Put all the **constraints** whose scope **contain *V*** on the **GAC queue**.
    - Repeat the followings for **all constraints** *C* on the **queue**:
        - Check **all the values** in the domain of **all variable** of *C*:
            - If a value has **no support prune it**.
            - If domain of a variable is **empty**, **stop** checking the constraints, **undo** all the **pruning** caused by assigning *d* to *V* and try **another value** for *V*.
            - Otherwise, put all **constraints** that are **affected by pruning** on the **GAC queue**.
    - If all constraints are ok, go **one level deeper** into the search tree (by **assigning** values to another **uninstantiated variable**).