Computer Science Department - Rutgers University **Fall 2017**

CS 520: Midterm Notes

16:198:520
Instructor: Wes Cowan

Each of the following sections represents material that is fair game on the midterm. I have included motivational questions for each. I cannot stress enough the importance of looking at small examples.

1 Uninformed Search

- Give an example of when the 'goal' of a problem may not be easy to identify beforehand, but once you reach it, you know. Will you always be able to tell when you've reached the 'goal'?
- What are the relative performances of DFS vs BFS? Complete or Optimal (with what exceptions)?
- What is the function / purpose of the fringe? What is the function / purpose of the closed list?
- What are the tradeoffs between BFS and DFS, when might you prefer one to the other?
- If you wanted to count the number of reachable goal nodes, what kind of search would you use?
- What are the structural / conceptual / implementational connections between BFS, DFS, and UFCS?
- Given a directed graph and a given vertex in that graph (or for instance an address in a network of one way streets), how could you go about finding the smallest cycle back to that vertex? How could you go about finding the longest cycle back to that vertex?
- Imagine running a search (of your choice) with a fringe *with a fixed, maximum size*. What would the effect of this limitation be on the performance of the search? Are there any situations where this might be a useful or beneficial thing to do?

2 Informed (Heuristic) Search

- In the context of search, what is a heuristic, and what is it used for? Where do they come from and how are they computed?
- Why is A^* important?
- What is an admissible heuristic, and why is the property important? What about consistency? (When I say *why*, looking at the proofs is important too.)
- If I have a collection of heuristics, how do I decide which one is the best one to use? Is there a best one, or can I get useful information potentially out of all of them? If each individual heuristic is admissible or consistent, how can I ensure that the combination of heuristics is admissible or consistent?
- What are the structural / conceptual / implementational connections between A^*, UFCS, and GBFS? What are the tradeoffs and when might you prefer one to the other?
- Give a heuristic for solving the Wolf/Goat/Cabbage problem in as few moves as possible. Argue that this heuristic is admissible.
- Any Rubik's cube configuration can be solved in at most 20 moves. How can this information be used to direct a search?

3 Local Search

- What distinguishes a local search algorithm from a global search algorithm? Why might a local search algorithm be useful at all when global search algorithms exist?
- Pros and Cons of Hill Climbing. How can it, or any local search algorithm, be (naively) improved upon? What is the point and purpose of simulated annealing?
- Many local search algorithms have a lot of tunable aspects or parameters (mutation rates in genetic algorithms, fitness functions, etc) - how should you go about making the relevant choices for your implementation of the algorithm?
- A genetic algorithm attempts to find the maximum of a function $f(x) = x(1 - x^2)$ on the interval $[0, 1]$ by representing solutions as a 5-digit decimal number, and 'breeding' them based on shared digits 0.12345 and 0.32315 might recombine to give 0._23_5, where the blank digits might be selected at random between the two parents. Solutions are then compared for fitness based on the size of f at that point. Do you think this is a reasonable means if implementing 'genetic recombination' of possible solutions? Why or why not? What other algorithms might you use? (Imagine, for instance, that f were a much more complicated function.)
- What's the difference between beam search, and just running several instances of hill-climbing (or hill-climbing in parallel)?

4 Adversarial Search

- What is a two-person zero-sum game with total information? How can these descriptors be varied?
- What is MiniMax search? Why is it useful in terms of game playing? What *kind* of search should you implement minimax search as? Why not the *other* kind of search?
- Generate some small examples of games (two player, with terminal utilities of your choice), and practice running minimax search to figure out how the game will unfold. What if the game is not zero-sum - what issues can you potentially run into?
- Game trees can be incredibly large for even simple games of interest. How do we simplify the problem of (minimax-)searching the game tree?
- What is alpha-beta pruning, how does it work, and why do we do it?
- Relate alpha-beta pruning and utility estimation to A^*.

5 Constraint Satisfaction

- Examples of constraint satisfaction problems other than sudoku. What is the basic formulation of constraint satisfaction problems we are interested in?
- Relate constraint satisfaction problems to search. What kind of search is backtracking search, and why is it good?

- Imagine trying to assign a set of variable to satisfy a large system of equations. How do you decide what variable to try assigning first? What variable to you try assigning second? How do you decide what values to try assigning?
- With a large number of variables, and large number of possible values for each variable, the search tree of possible assignments can easily be vast. What kinds of tricks can we use to avoid / break out of non-viable search tree branches as quickly as possible?
- How could we apply local search algorithms to constraint satisfaction problems? What would the pros and cons be?

6 Logic and Satisfiability

- Logic as Modeling, Equivalence between Logical Statements, Truth Tables, CNF vs DNF, Converting Logical Forms
- Basic familiarity with rules of inference, recombining logical statements to conclude new logical statements.
- Euclid defined five geometric axioms which he then used to prove all his theorems with, logically. Model this process as a search. What are the states, what are the actions that move between states? If mathematics can be thought of as (automatable) logical search from axioms to conclusion, what is the purpose of mathematicians? What are they for?
- Does logical inference generate *new* information?
- Resolution Inference Rule: How, What, and Why (and what is the cost?)
- Proof by Contradiction: How, What, Why (and what should you be careful of?)
- How can we apply the notion of proof by contradiction to constraint satisfaction problems?
- Relate 'satisfiability' to proof by contradiction. How can constraint satisfaction algorithms be applied to logical inference?
- Which is more useful, showing that a knowledge base is satisfiable, or that it is unsatisfiable?
- Davis-Putnam, and rules for simplifying the search for logical satisfiability.

7 Classical Planning

- Think of some simple examples and try to describe them in terms of classical planning formulations.
- What are the state descriptions? What are the possible actions? What are the pre-conditions and effects? What is the goal state?
- Heuristics relevant to your example, and heuristics relevant to planning in general.

8 Probability and Inference

- Marginalization and Conditioning. What are they, what are they good for. Be. Able. To. Use. Them.
- Independence and Conditional Independence. What do they *mean* and how can we use them?
- Be able to break probabilistic queries down into factors and terms that you know / can access in your probabilistic knowledge base.
- Bayes. Know it. Use it. Love it.