Midterm Answers
New, Improved, and
Including the Postmortem "FFQ" Feature [look for square brackets]

CSC 242

18 March 2002

*Write your **NAME** legibly on the bluebook. Work all problems. Best strategy is not to spend more than the indicated time on any question (minutes = points). Open book, open notes.*

1. Search: 15 Min. Consider (the bad idea of) using AI techniques to sort a sequence of integers. The primitive operator is to swap the positions of two integers in the sequence. Assume (contrary to fact) that this swapping takes much longer than computing intermediate things like the number of inversions (out-of-order integers), or "local inversions" (number of out-of-order neighboring integers), etc.
(a)(1 min.) For a sequence of N integers, how big is the state space?
(b)(5 min.) For heuristic search (A*): what are a g function (informally, how far you've come [Actually, this is ambiguous. I meant "how much work you've done", not "how much progress you've made. Some people got misled.]) and an admissible h function (informally, how far you have to go)? If h is not admissible, what are the consequences?
(c)(5 min.) Next, set up this problem as a constraint satisfaction problem and discuss the issues.
(d)(4 min.) Next, similarly for a hill-climbing or simulated annealing approach.

You may want to use concepts like inversions, NlogN, or whatever you know from data structures about comparison sorting – If any of the resulting sorts remind you of sorts you know about, draw those parallels (up to 5 points extra).

Answer:

State space has $N!$ states.

[Don't confuse this with the size of the *search* space, which could be infinite if you don't take care, or could be small if you give up!] State space size doesn't have anything to do with search trees.

[I didn't say integers from 0 to N-1, for instance. It's important that we don't know "the final position" of these numbers in advance (so it isn't like the 14-15 puzzle...). On the other hand I should have said "distinct" since the size of the state space does depend on that and I left it unspecified.]

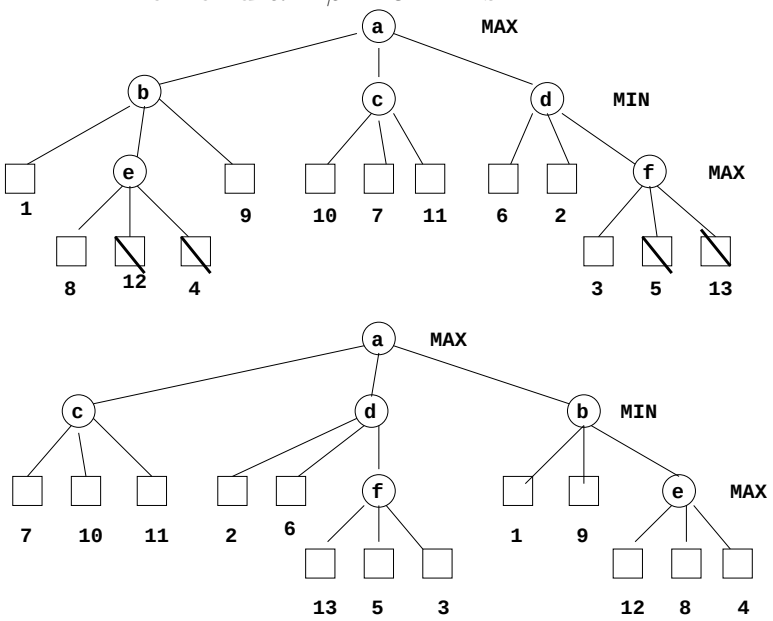
For A*, g could be how many swaps made so far, f could be number of inversions remaining. That underestimates the work and so is admissible. The idea of "local inversions" is interesting. Is

it admissible? Or you could be optimistic and make $f = (NlogN) - g$? Probably not admissible but not crazy...if it's not you won't find the optimal sequence of swaps.

Constraint satisfaction: constraint is that numbers are in order, (there are no inversions) and if you count the number of inversions removed by an operator and pick the one that removes the most you'll have a good sort...the minconflicts approach. Sounds a little like shakersort, but actually should lead to optimal performance, seems to me...

Hill Climbing. Sounds kind of like bubble sort. Think locally and swap with your neighbor if you two are out of order. This approach seems to have the problem that you really *want* to swap with some distant guy and kill off lots of inversions currently between you. Simulated annealing seems to say every now and then swap with some random other element, and presumably move your focus of attention over to his neck of the woods. I don't see how this helps much in this problem.

2. Minimax and $\alpha - \beta$: 15 Mins.



Two different programs playing the same game at the same stage produce the game trees shown. You'll notice they differ only in the order in which moves are generated: the move generators are different, in short. Tthe square nodes are moves whose evaluation function is known (sometimes MAX has only one move, sometimes three).

(a)(1 min.) What score is MAX guaranteed? (1 Min).

The upper tree shows the result of $\alpha - \beta$ pruning: crossed nodes are not in fact evaluated.

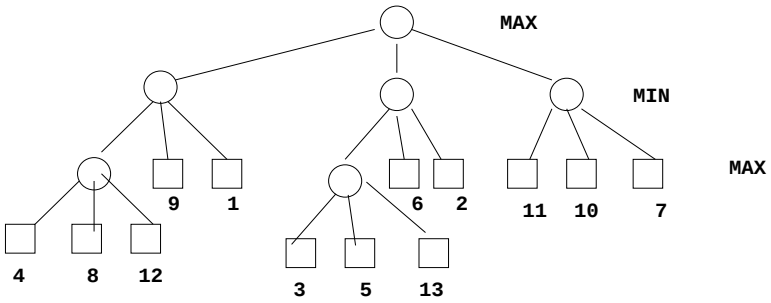
(b)(7 min.) Copy the second tree and similarly mark the unevaluated nodes. If there are more or fewer, say why (7 min).

(c)(7 min.) Rewrite the tree for a "pessimal" move generator that produces nodes in the worst possible order for $\alpha - \beta$ pruning. How many nodes are pruned in this case? (7 min.).

Answer:

MAX gets 7.

Nodes f, 6, 13, 5, 3, e, 9, 12,8, 4 don't get explored. The generator has produced alternatives in the optimal order, in that both MAX and (max simulating) MIN find out the best moves as soon as possible. If you could always do this you wouldn't need to search, of course. Knuth proved that in the infinite limit you can double the depth of search (not clear in this small example) with optimal pruning.



With this move generator, there is no pruning. It does the reverse of the optimal one. [Several people generated trees where there might not be any (or much) pruning but they created different paths, so in a sense it wasn't comparable, since if the paths aren't the same it means the same moves aren't legal at a given point, so it's really a different game.]

3. Clause Form: 5 Min.

Put the following expression into clause form.

$$(\forall x[(\exists y[P(x,y) \vee Q(x,y)]) \Rightarrow (\exists y R(x,y))]).$$

Answer:

[This is a bit embarrassing, but the book doesn't use term "clause form", preferring "normal form". I caught this partway thru exam and put it on board.]

Two clauses result:

$$\neg P(x,y) \vee R(x,F(x)), \neg Q(x,y) \vee R(x,F(x))$$

Important to do things in the right order. In particular, Skolemizing too early is death!

4. Resolution Proof: 10 Min.

Given these premises:

1. Some professors are not entertainers.
2. Only entertainers are highly paid.

Express them in FOPC, put them into clause form, and use resolution to answer the question: Who is not highly paid (*id est*, is anyone not highly paid, and if someone is not, who is it that is not?).

Answer:

[Sigh. As usual, most people weren't comfy with this, but performance this year may be better than past years. It doesn't get much more straightforward than this, so it may repay some study.]

$$P1. \quad \exists(x)(P(x) \wedge \neg E(x))$$

$$P2. \quad \forall(x)(H(x) \Rightarrow E(x))$$

Query (not negated – To Be Proved:)

$$\exists(x)(\neg H(x))$$

negate the conclusion, clause form:

Negated Query:

A: $H(x)$
P1 yields
B: $P(A)$
C: $\neg E(A)$
P2 yields
D: $\neg H(x) \vee E(x)$
Resolutions:
C,D2 yields E: $\neg H(A)$
E,A: yields null
with substitution x/A, so A is the person who is not highly paid. Note that A is also a professor, which we could also prove (but this proof does not).

5. Odds and Bayes' Rule: 15 Mins.

An event X with probability P has (prior) *odds* $O(X) = P(X)/(1 - P(X))$.

(a)(1 min.) Given the odds of an event, what is its probability?

The *likelihood ratio* λ of evidence E given a hypothesis H is $P(E \mid H)/P(E \mid \neg H)$. Such ratios are provided by experts or from experimentation.

(b)(7 min.) Show that when evidence E comes in, we update the odds on the hypothesis H (that is, we compute $O(H \mid E)$) simply by multiplying $O(H)$ by the likelihood ratio λ. (Hint: you want to show the product is equal to $P(H \mid E)$ / $P(\neg H \mid E)$ — use Bayes' rule.)

(c)(7 min.) The prior probability of a car being a lemon is 0.2. What are the prior odds? The probability of a lemon having an oil leak is .3, and the probability that you have an oil leak on a non-lemon auto is .1. The car you're thinking of buying has an oil leak. What are the odds it is a lemon?

Answer:

[Big hint, which applied to EVERYONE this year: Mathematics is NOT a bunch of equations in a row. You've got to use English (or I can deal with German or French) prose to tie together the equations. This year I was a big woolly bear and gave full credit for some answers (I had lots of time on the beach to grade) but I'm fairly sure some people lost points, and if I get in a hurry next time you'll really lose points for not writing math on these tests.]

$P(x) = O(x)/(O(x) + 1)$

By Bayes' rule we know

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Likewise,

$$P(\neg H \mid E) = \frac{P(E \mid \neg H)P(\neg H)}{P(E)}$$

It's easy to show that the quotient of these two is equal to the product of O(H) and λ, which both equal

$$O(H \mid E) = \frac{P(E \mid H)P(H)}{P(E \mid \neg H)P(\neg H)}$$

Hypothesis is "Lemon". Evidence is "Leak". Prior Odds of H = 1/4.

$P(E \mid H) = .3$ and $P(E \mid \neg H) = .1$. So by our simple multiplicative odds formulation, the odds $O(H \mid E)$ are 3(1/4) or 3/4, or if you like probabilities, $P(lemon \mid oil - leak) = 3/7$.

[Some people worked out the Probability, using Bayes' theorem. I'm quite proud of them since that involved knowing how to compute the prior odds of an oil leak from other things that were known, but it wasn't in the spirit of the question and only got 4/7 points, as I recall. The idea is that odds can make evidence combination easier.]

6. Planning: 15 Mins.

Use planning to develop an algorithm to swap the contents of two registers. Let's say we have three registers, A, B, and C, and the initial conditions are: Contains(A,x), Contains(B,y), Contains (C, z). The goal state is Contains(A,y), Contains(B,x). The operator is Assign(A,B), which replaces the contents of B with the contents of A and leaves the contents of A alone.

(a)(7 min.) 1. How would STRIPS solve this problem? (We know the answer already! The issue is how STRIPS gets it, i.e. we need definitions of the operator and what goes on in the planning process, what the resuling series of steps is and why, etc.)

(b)(8 min.) How would UCPOP solve this problem?

Answer:

[Well, frankly about half the answers were better than these below, but I got lazy. People got full credit for defining the strips operators, talking about the control structure, and describing how a solution to the register-swapping could be found. For UCPOP, I saw some nice graphical realizations of the planning that showed people knew what was going on. Along with some prose about the concepts.

What did NOT work is copying true but totally general descriptions of STRIPS and POPlanners out of the book! The whole idea is to apply the concepts to a particular problem. Remember Feynmann and the Brazilian Physics Students!]

STRIPS: the Assign(A,B) operator has preconditions Contains (A,x), Contains(B,y); Add List: Contains(B,x); Delete List: Contains(B,y).

Concepts: need to hear about linear planning (sequence of steps), search in state space, the Goal Stack, Difference Reduction, note how you get in trouble with subgoal interaction, and if people can actually work thru the whole thing or part of it, great.

UCPOP: Start with the start and goals state encoded as states with preconditions equal to the start and goal conditions. Add steps that achieve goal "preconditions" one at a time, check for potential clobbers with other steps preconditions. If have a problem, promote or demote the added step.

Concepts; search in plan space, threat links, causal links, demotion and promotion, partial order of operations.