CS 480/580

Introduction to Artificial Intelligence
Final Exam

11/29/2018

Name: John Hessefort UIN: 01046558

Open book and notes. Please write your answers on the test sheet

55 +5= 60/100

Total: 100pts + 5 pts extra credit

1. Multiple Choices (30 pts) 18/30

1) What is the space complexity of depth-limited depth-first search with branch factor b, solution depth d, and maximum search depth m?

A) $O(bm)$
B) $O(bd)$
C) $O(b^m)$
D) $O(b^d)$

3) In A* search, if $f(x)$, $g(x)$, and $h(x)$ are all admissible heuristics, then which of the following are also guaranteed to be admissible heuristics?

~~A)~~ $f(x) + g(x) + h(x)$ Admissible : Never overestimates path cost.
B) $f(x) * g(x) * h(x)$
C) $f(x)/2 + g(x)/3 + h(x)/3$ ← If guaranteed, then D guaranteed too.
D D) $f(x)/2 + g(x)/4 + h(x)/4$ ← Gives lowest value, does not overestimate.

4) In the worst-case, which of the following search algorithms has the lowest space complexity?

A) depth-first ~~(b^d)~~ (bm)
B) breadth-first (b^d)
C) A*
D) bi-directional ($b^{(d/2)}$) : Bi-directional only needs to expand half of the solution path.

5) Completeness of a search algorithm mean that the search

A) finds the solution that has the lowest path cost among all solutions
B) finds all solutions If B, then A. Find all solutions → Found lowest-cost solution.
C) finds a solution using the least amount of memory
D) finds a solution using the least amount of time

6) Which of the following statement about an intelligent agent is true?

A) A model-based reflex agent selects actions on the basis of the current percept, ignoring the rest of percept history.

B) A simple reflex agent keeps track of the percept history.

C) A goal-based agent has a utility function indicating the "degree of happiness" of a state.

D) All of the above statements are not true.

7) The typical implementation of A* Search uses the data structure of

A) A stack (LIFO)

B) A queue (FIFO)

C) A priority queue (FIFO with priority) Need priority queue for heuristic/evaluation function.

D) None of the above

8) The genetic algorithm (crossovers and mutations to advance population by fitness score.)

A) always terminates at the global optimal solution. Not always guaranteed.

B) may terminates at the local minimum. ~~Continues for as long as desired.~~

C) has to maintain a search tree.

D) often yields fast convergence. Ideal populations are found in few generations.

9) Which of the following KB is unsatisfiable?

A) $\{\neg P \vee \neg Q, P \wedge Q\} = \{\neg(P \wedge Q), P \wedge Q\}$ (DeMorgan's Law)

B) $\{P \vee Q, P \wedge Q\}$

C) $\{P \rightarrow \neg Q, \neg P \wedge \neg Q\}$

D) $\{P, P \vee Q\}$

10) Consider the fuzzy logic representation. Assuming P = 0.5, Q = 0.6, what is $\neg P \wedge \neg Q$?

A) 0.6

B) 0.5

C) 0.4

D) 1

$\cancel{\neg P = 0.5 - 1} = -0.5 = -\frac{1}{2}$

$\neg Q = 0.6 - 1 = -0.4 = -\frac{2}{5}$

$\cancel{\neg P \wedge \neg Q = \neg P}$

Correction: $\neg P = 1 - 0.5 = 0.5$

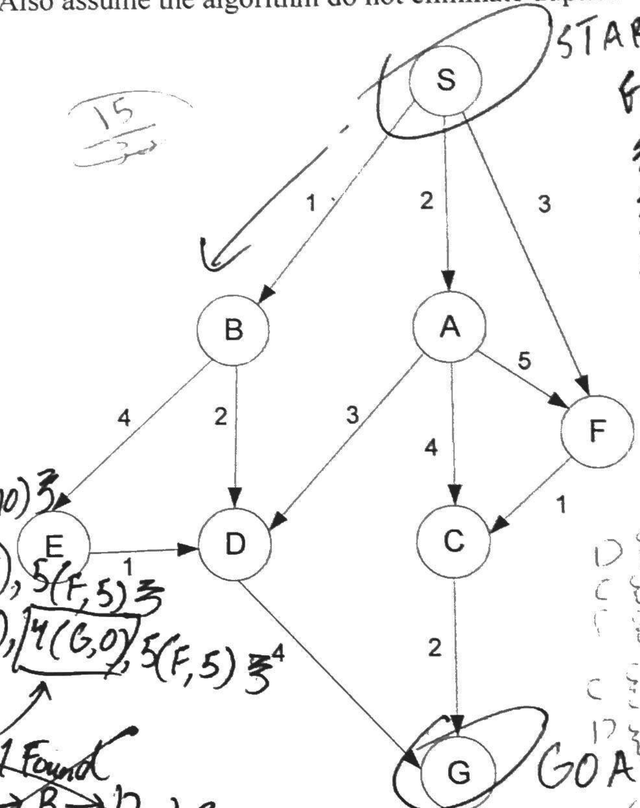
$\neg Q = 1 - 0.6 = 0.4$

$\neg P \wedge \neg Q = \text{Min}(\neg P, \neg Q) = \text{Min}(0.5, 0.4) = \boxed{0.4}$

2. Consider the search algorithm for the following graph with node S as the start node and node G as the goal node. The path cost value is showed next to the edge. The heuristic value is shown in the heuristic table. During search, if there are multiple nodes directly accessible from one node, the search always visits the one on the left first. If a new added node has the same cost as an existing one in the fringe, it is always put after the existing node. Also assume the algorithm do not eliminate duplicated nodes in the fringe (30 pts)

15

BFS : queue (priority)
Answer
~~Fringe: {0(S,5)}~~
~~{1(B,6), 2(A,1), 3(F,5)}~~
~~{2(A,1), 2(D,1), 3(F,5), 4(E,10)}~~
~~{2(D,1), 3(F,5), 3(D,1), 4(C,4), 5(F,5)}~~
~~{3(F,5), 3(D,1), 4(C,4), $\boxed{4(G,0)}$, 5(F,5)}~~
~~Goal Found~~
~~Path: S→B→D→G~~



Final Answer (BFS)
{S}
{B, A, F}
{A, F, E, D}
{F, E, D, D, C, F}
{E, D, D, C, F, C}
{D, D, C, F, C, D}
{~~D~~ D, C, F, C, D, $\boxed{G}$}
D{C,F,C,D,G,G}
Goal found
Path: S→B→D→G
G{G,G,G,G,G}

Heuristic

S	A	B	C	D	E	F	G
5	1	6	4	1	10	5	0

1) Simulate the behavior of Bread-First Search. Show the node visited and the nodes in the fringe at each step. Print out the final solution.

Breadth-First Search : Expand every neighbor node first. Begin with S.

Fringe: ~~{0(S)}~~ {0(S,5)}
~~{1(B), 2(A), 3(F)}~~ ~~{1(B,6), 2(A,1), 3(F,5)}~~
~~{4(E,10), 2(A,1),~~ 3(F,5), 4(E,10), 2(D,1)}
~~{3(F,5),~~ 4(E,10), 2(D,1), 3(D,1),

4) Simulate the behavior of Greedy Best-First Search. Show the node visited and the nodes in the fringe at each step. priority-queue

Expand closest neighbors first. Only considers current path cost, not total.

-3

Fringe: {S}, {1(B), 2(A), 3(F)}, {2(A), 2(D), 3(F), 4(E)}

{2(D), 3(F), 3(D), 4(E), 4(C), 5(F)}

{3(F), 3(D), 4(E), 4(C), $\boxed{4(G)}$, 5(F)} Path: S → A → D → G

{1(C), 3(D), 4(E), 4(C), 5(F)} Goal found.

Path: S → B → D → G (found)

{$\boxed{2(G)}$, 3(D), 4(E), 4(C), 5(F)} Path: S → ~~B~~ F → C → G (cost = 7)

5) Simulate the behavior of A* Search. Show the node visited and the nodes in the fringe at each step.

A* has f(n), g(n), h(n) where f(n) estimates cost to cheapest solution, g(n) is path history (cost from root ~~till~~ until node n), and h(n) is estimate of cost of cheapest path n → goal (heuristic).

Priority-Queue Format: (currentCost(PathName, PathCostTotal, Heuristic))

Fringe: ~~{0(S, 5)}~~ ~~{0({~~ {0(S, 0, 5)}

-3

Path: S → A → D → G

6) Justify your answer on question 5). Does A* search provide the optimal solution? Why or why not?

No

A* Search, when implemented with estimable and valid heuristic will always find the ideal solution.

-3

3. (15 pts) Consider using first-order logic to describe the rules of British crown succession. Assuming the following relations and functions are given.
Monarch(x): True if x is the Monarch.
Child(x, y): True if y is x's child.
EldestMaleChild(x, y): True if y is x's eldest male child.
EldestFemaleChild(x, y): True if y is x's eldest female child.
Male(x): True if x is male.
Female(x): True if x is female.
Catholic(x): True if x is a Roman Catholic.
Heir(x, y): True if y is the heir of x.
Age(x): Function returns the age of x.

$\frac{13}{15}$

1). There is one and only one monarch.

-1

$\exists_m \text{Monarch}(m) \rightarrow \forall_p \neg\text{Monarch}(p) \wedge p != m$

There is some monarch m s.t. for all people p, p is not the monarch and p is not m.

2). Roman Catholic may not be the monarch.

~~Roman Catholic does not imply monarch / impt~~ RC implies not monarch.

$\forall_r \text{Catholic}(r) \rightarrow \neg\text{Monarch}(r)$ ✓

3). The eldest child has the oldest age among one's male children.

-1

$\cancel{\forall_{ec, c, p} \text{EldestMaleChild}(p, ec) \rightarrow}$

$\forall_{ec, p} \text{EldestMaleChild}(p, ec) \rightarrow \forall_c \text{child}(p, c) \{age(c) < age(ec)\} \wedge c \,!{=}\, ec$

The eldest male child of p? Then all other children have smaller age.

4). If the monarch has male children, the eldest male child will be the heir.

$\cancel{\forall_{m,c} \exists}$ $\forall_{m,c} \text{Monarch}(m) \wedge \text{Child}(m, c) \wedge \text{Male}(c) \rightarrow \exists_{ec} \text{EldestMaleChild}(m, ec) \wedge \text{Heir}(m, ec)$

$\forall x \text{ Monarch}(x) \exists y (\text{EldestMaleChild}(x, y) \rightarrow \text{Heir}$

$\forall x \text{ Monarch}(x) \exists y \text{ EldestMaleChild}(x, y) \rightarrow \text{Heir}(x, y)$

5). If there are no male children, the eldest female child will be the heir.

$\forall_{m,c} \text{Monarch}(m) \wedge \text{Child}(m, c) \wedge \neg\text{Male}(c) \rightarrow \text{Female}(c) \wedge \exists_{ec} \text{EldestFemaleChild}(m, ec) \wedge \text{Heir}(m, ec)$

$\forall x \text{ Monarch}(x) \wedge \exists_y \text{Male}(y) \wedge \text{Child}(x, y) \wedge \exists z \{\text{EldestFemaleChild}(x, y) \rightarrow \text{Heir}(x, z)$

For all Monarchs that have no children (male), there is an eldest female child s.t. (such that) she is the heir to the Kingdom.

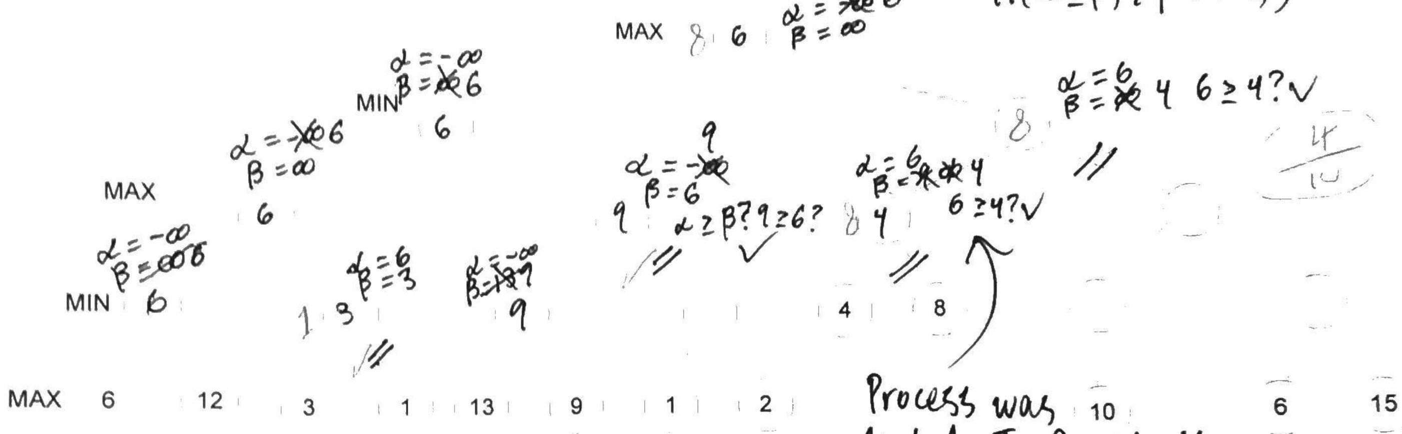
9/20

4. For the tree below representing a minimax game use the minimax algorithm with alpha-beta pruning to determine the optimal moves to get the highest possible score if MAX and MIN make perfect moves.

1) Show your work by labeling the nodes with the values of minimax algorithm. (5 pts)

2) Highlight the optimal move on the game tree. (5 pts)

if($\alpha \geq \beta$) { prune(); }



4/10

Process was correct, but I forgot this level was for player MAX, so I changed β on accident.

Optimal move is missing.

3) Cross out all the pruned branches in the above game tree using the alpha-beta pruning algorithm. Show the alpha and beta values at every step. (10 pts)

5/10

Optimal move

Other branches are pruned.