CS 480

Spring 2004
Dr. Duric

Midterm Review

1. Answer briefly:

   a) What is the major difference between blind and heuristic search methods.

   b) Enumerate and define three major blind search methods.

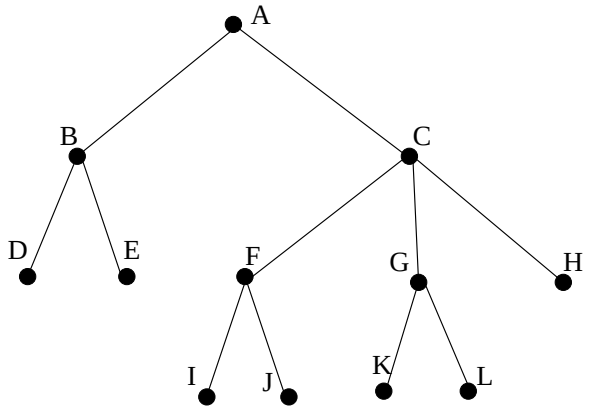
   c) How is the hill-climbing method different from the best-first search method? For what classes of problems are the hill-climbing and the best-first search methods equivalent?

2. List the order in which nodes are visited in the tree below for each of the following search strategies (choosing leftmost branch in all cases):

   a) Depth-first search

   b) Depth-first iterative-deepening search (increasing the depth by 1 each iteration)

   c) Breadth-first search



3. A boy brings a basket of cabbage, a wolf, and a goat to a river. There is a small boat on their river bank that they can use to cross to the other side. However, the boat is so small that it can hold only the boy and the cabbage or the boy and one of the animals. The boy cannot leave the goat alone with either the cabbage or the wolf. How should the boy use the boat to take the cabbage and the animals to the other side of the river in such a way that the number of river crossings is minimal and that the goat is never left alone with either the wolf or the cabbage on any side of the river? Use iterative deepening search to solve this problem.

4. In the water-jug puzzle we are given a 4-liter jug, and a 7-liter jug. Initially, both jugs are empty. Either jug can be filled with water from a tap, and we can discard water from either jug down a drain. Water may be poured from one jug into the other. There is no additional measuring device. We want to find a set of operations that will leave precisely x liters of water in either one of the jugs.

   i. Set up a state-space search formulation of the water jug puzzle:

a) Given the initial iconic state description as a data structure.

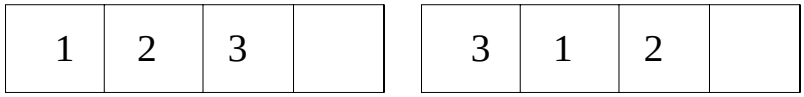
b) Give a goal condition on states as some test on data structures.

c) Name the operators on states and give precise descriptions of what each operator does to a state description.

ii. Find whether the goals $x = \{1, 2, 3, 4, 5, 6, 7\}$ can be accomplished in 8 or fewer steps.

Hint: Use breadth-first search.

5. A simple sliding-tile puzzle consists of 3 numbered tiles and an empty space. The puzzle has two legal moves with associate costs: (i) A tile may move into an adjacent empty location. This has a cost of 1. (ii) A tile can move over one other tile into the empty position. This has a cost of 2. The start and goal positions are given below. Set up a state-space search



start goal

formulation for this puzzle:

a) Specify the form of state descriptions, the starting state, and the goal state for this problem.

b) Name the operators on states and describe what each operator does to a state description. Write a lisp function that takes a state as input and outputs a list of states that can be reached from that state. Include the parent and the cost fields.

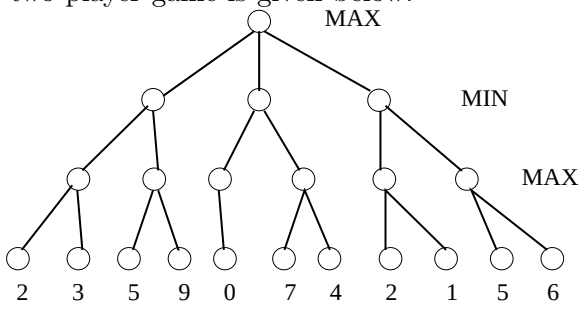
c) Propose a heuristic function $\hat{h}$ for solving this problem.

d) Use the A^* algorithm to find a solution path using your heuristic function.

e) Is your solution path optimal? Give a formal argument.

6. The game *nim* is played as follows: Two players alternate in removing one, two, or three pennies from a stack initially containing five pennies. The player who picks up the last penny loses. Show, by drawing the game tree, that the player who has the second move can always win. Can you think of a simple characterization of the winning strategy? Is it necessary to generate the whole tree to find a winning strategy?

7. A partial search tree for a two player game is given below.



a) Find the best move for the MAX player using the minimax procedure.

b) Using alpha-beta pruning show which parts of the tree do not need to be searched. Indicate where the cutoffs occur.

8. A simple version of the *nim* game is played as follows: Two players alternate in removing stones from three piles initially containing one, one, and five stones, respectively. The player who picks up the last stone wins. Each player can pick one or more stones from a single pile; at least one stone has to be picked every time. At every turn the players can pick from different piles.

   a) Show, by drawing a game tree, which player can always win. You may use either DFS or BFS method to generate nodes.

   b) Is it necessary to generate the whole tree to find a winning strategy? Explain why or why not. Is it possible to use the alpha-beta pruning on this game tree?

9. Write a recursive lisp function that takes a list as an argument and returns the number of atoms on any level of the list. For instance, list $(A \ \ B \ \ (C \ \ D \ \ E) \ \ ())$ contains six atoms $(A, B, C, D, E,$ and $NIL)$.

10. Write a lisp function $shuffle$ that takes a list of 52 different symbols and returns a list in which the original 52 elements are mixed in random order. Hint: Use functions *random* and *remove*.

11. Write a lisp function $last5$ that takes a list A as its argument and returns a list B consisting of the last five elements of A. You are not allowed to use the built-in function *last*.
    (last5 '(A B C)) should return *(A B C)*
    (last5 '(A B C D E F G H)) should return *(D E F G H)*

12. A simple version of a "Grundy's game" is played as follows: Two players have in front of them a single pile of objects, say a stack of 7 pennies. The first player divides the original stack into two stacks that *must* be unequal. Each player alternatively thereafter does the same to *some* single stack when it is his turn to play. The game proceeds until each stack has either just one penny or two—at which point continuation becomes impossible. The player who first cannot play is the loser. Show, by drawing a game tree, whether any of the players can always win.

13. The 8-puzzle consists of eight numbered, movable tiles set in a 3×3 frame. One cell of the frame is always empty thus making it possible to move and adjacent numbered tile into the empty cell—or, we could say, to move the empty cell. Two configurations of tiles are given. Consider the problem of changing the initial configuration into the goal configuration.

2	4	3
1	6	8
5		7

start

1	2	3
4		8
5	6	7

goal

Set up a state-space search formulation for this puzzle:

   a) Specify the form of state descriptions, the starting state, and the goal state for this problem.

b) Name the operators on states and describe what each operator does to a state description.

c) Propose a heuristic function $\hat{h}$ for solving this problem.

d) Use the A^* algorithm to find a solution path.

e) Is your solution path optimal? Give a formal argument.

14. Write a recursive function flip that takes a binary tree as input and returns a binary tree that it is its mirror image. You can represent binary trees as nested structures:
Nested (recursive) representation: **(<root> (<left subtree>) (<right subtree>))**
Examples:

(flip '(1 2 3))	should return **(1 3 2)**
(flip '(1 (2 3 4) ()))	should return **(1 () (2 4 3))**
(flip '(1 (2 (3 4 5) (10 11 12)) (6 () (7 () 8))))	should return **(1 (6 (7 8 ()) ()) (2 (10 12 11) (3 5 4)))**

15. a) Write a lisp function *funny_first* that takes a list of flat lists and returns a new list composed of the first elements of the original flat lists.

b) Write a lisp function *funny_last* that takes a list of flat lists as its argument and returns a new list composed of the last elements of the original flat lists.

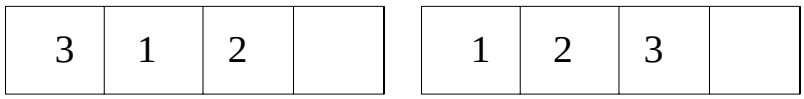
c) Write a lisp function *funny_len* that takes a list of flat lists as its argument and returns the sum of the lengths of the nested lists.

d) Write a lisp function *funny_sum* that takes a list of flat lists of numbers and returns the sum of the elements of the nested lists.

Examples:

(funny_first '((A B) (C) (D E) (F G H)))	should return **(A C D F)**
(funny_last '((A B) (C) (D E) (F G H)))	should return **(B C E H)**
(funny_len '((A B) (C) (D E) (F G H)))	should return **8**
(funny_sum '((1 2) (3) (4 5) (10 20 30)))	should return **75**

16. A simple sliding-tile puzzle consists of 4 numbered tiles and an empty space. The puzzle has two legal moves with associate costs: (i) A tile may move into an adjacent empty location. This has a cost of 1. (ii) A tile can move over one other tile into the empty position. This has a cost of 2. The start and goal positions are given below. Set up a state-space search



start goal

formulation for this puzzle:

a) Specify the form of state descriptions, the starting state, and the goal state for this problem.

b) Name the operators on states and describe what each operator does to a state description.

c) Propose a heuristic function $\hat{h}$ for solving this problem.

d) Use the A^* algorithm to find a solution path.

e) Is your solution path optimal? Give a formal argument.

17. The sliding-tile puzzle consists of three black tiles, three white tiles and an empty space in the configuration shown below

B	B	B		W	W	W

The puzzle has two legal moves with associate costs:

> A tile may move into an adjacent empty location. This has a cost of 1. A tile can move over one or two other tiles into the empty position. This has a cost equal to the number of tiles jumped over.

The goal is to have all the white tiles to the left of all the black tiles with the blank in the middle.

a) Choose a representation for this problem so that A^* algorithm can be applied to find a solution.

b) Propose two heuristics for solving this problem.

c) Show all states that can be reached from the start and compute their g and h values for one of your heuristics (state clearly which one you are using). Mark the state that would be expanded next by A^* algorithm — you do not need to expand that state.

18. The 8-puzzle consists of eight numbered, movable tiles set in a 3×3 frame. One cell of the frame is always empty thus making it possible to move an adjacent numbered tile into the empty cell—or, we could say, to move the empty cell. Two configurations of tiles are given. Consider the problem of changing the start configuration into the goal configuration.

1	2	3
4		8
5	6	7

start

2	4	3
1	6	8
5		7

goal

Set up a state-space search formulation for this puzzle:

a) Specify the form of state descriptions, the starting state, and the goal state for this problem.

b) Name the operators on states and describe what each operator does to a state description.

c) Propose a heuristic function $\hat{h}$ for solving this problem.

d) Use the A^* algorithm to find a solution path.

e) Is your solution path optimal? Give a formal argument.

19. A simple version of a "Kayles game" is played as follows: Two players have in front of them a single contiguous sequence of objects, say 5 pennies, which are placed next to each other so that the first penny touches the second, the second touches the third, the third touches the fourth, and the fourth touches the fifth penny. The first player removes 1 or 2 pennies whose sides that are touching (for example, the first player can remove any single penny, or pennies 1 and 2, or pennies 2 and 3, or pennies 3 and 4, or pennies 4 and 5). Each player alternatively thereafter removes 1 penny or 2 pennies whose sides are touching. (Note: If the first player removes penny 2, the pennies 1 and 3 will have a gap between them and cannot be removed at the same time.) The last player to pick a penny (or two) loses. Show, by drawing a game tree, whether any of the players can always win. You can ignore symmetries and obvious losing plays if there is a winning play available, such as picking both pennies when there are 2 pennies left.

20. a) Write a non-recursive function that takes a list as its argument and reverses it. Hint: use the *dolist* statement.

    b) Write a recursive function that takes a list as its argument and reverses it.

    Examples:

(reverse '(A B C (D E) F))	should return **(F (D E) C B A)**
(reverse '(1 2 3 (4 5 6)))	should return **((4 5 6) 3 2 1)**

21. What do these functions do?

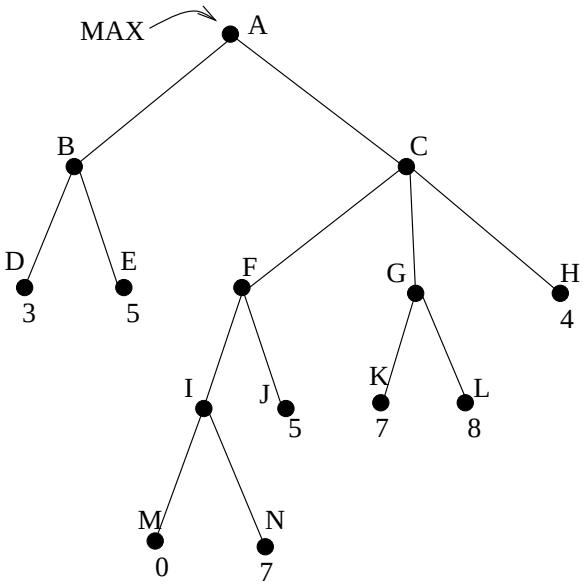
```
(defun xx (lst)
   (if (atom lst) 0
      (if (null lst) 0
         (if (member (first lst) (rest lst)) (+ 1 (xx (rest lst)))
            (xx (rest lst))))))

(defun yy (lst)
   (if (atom lst) (yy (list lst))
      (mapcar #'(lambda (x) (if (atom x) (list t x) nil)) lst)))

(defun dd (lst)
   (cond ((atom lst) lst)
         ((atom (first lst)) (cons (first lst) (dd (rest lst))))
         (t (append (dd (first lst)) (dd (rest lst))))))
(defun zz (lst)
   (cond ((null lst) nil)
         ((numberp (first lst)) (zz (rest lst)))
         (t (append (list (first lst))
                    (zz (rest lst)) (list (first lst))))))
```

22. Consider the following game tree.

    (a) Find the best move for the MAX player using the minimax procedure.



(b) Perform a left-to-right alpha-beta pruning on the tree. Indicate where the cutoffs occur.

(c) Perform a right-to-left alpha-beta pruning on the tree. Discuss why different pruning occurs.

23. In the water-jug puzzle we are given a 2-liter jug, and a 5-liter jug. Initially, both jugs are empty. Either jug can be filled with water from a tap, and we can discard water from either jug down a drain. Water may be poured from one jug into the other. There is no additional measuring device. We want to find a set of operations that will leave precisely x liters of water in either one of the jugs.

    A. Set up a state-space search formulation of the water jug puzzle:

        Give the initial state description as a data structure. Define the goal predicate as a function of a state. Name the operators on states and give precise descriptions of what each operator does to a state.

    B. Find whether the goals $x \in \{1, 2, 3, 4, 5\}$ can be accomplished in 4 or fewer steps.

24. A simple version of the *nim* game is played as follows: Two players alternate in removing stones from three piles initially containing two stones each. The player who picks up the last stone wins. Each player can pick one or more stones from a single pile; at least one stone has to be picked every time. At any turn the player can pick from any one pile. Show, by drawing the game tree, whether a player can always win, and if so which one. Is it necessary to generate the whole tree to find a winning strategy?

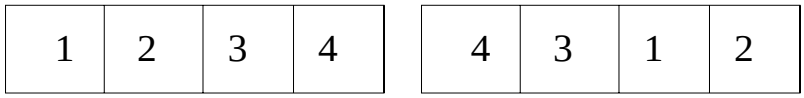
25. Write a lisp function that takes a flat list as an argument and returns a list whose elements are those elements of the original list that are not numbers.

26. Write a lisp function that takes a flat list as an argument and returns a sum of the numbers in the original list. Your function should not add the non-number elements of the original list.

27. Write a lisp function *shuffle* that takes a list of 32 different symbols and returns a list in which the original 32 elements are mixed in random order. Hint: Use functions *random* and *remove*.

28. Write a lisp function *d-shuffle* that takes a list of 32 different symbols and returns a list in which the first 16 original symbols are interleaved with the second 16 original symbols, i.e. list $(s_1\ s_2\ s_3\ s_4 \ldots s_{29}\ s_{30}\ s_{31}\ s_{32})$ becomes $(s_1\ s_{17}\ s_2\ s_{18} \ldots s_{15}\ s_{31}\ s_{16}\ s_{32})$.

29. A simple sliding-tile puzzle consists of 4 numbered tiles. The puzzle has two legal moves. (i) Two adjacent tiles can switch positions. This has a cost of 2. (ii) Two non-adjacent tiles can switch positions. This has a cost of 3. Set up a state-space search formulation for this puzzle:



start goal

a) Specify the form of state descriptions, the starting state, and the goal state for this problem.

b) Name the operators on states and describe what each operator does to a state description.

c) Propose a heuristic function $\hat{h}$ for solving this problem.

d) Use the A^* algorithm to find a solution path.

e) Is your solution path optimal? Give a formal argument.

30. A simple version of the *nim* game is played as follows: Two players alternate in removing stones from three piles initially containing two, two, and three stones, respectively. The player who picks up the last stone wins. At any given turn a player can pick one or more stones from a single pile; at least one stone has to be picked every time.

a) (20p) Show, by drawing a game tree, which player can always win.

b) (5p) Is it necessary to generate the whole tree to find a winning strategy? Explain why or why not.

31. Show how different techniques can be used to solve the 4-queens placement problem (place 4-queens on a 4×4 board so that there is at most one queen in each row, column, or a diagonal of the board. Solve this problem using the following tecniques:

a. Backtracking search.

b. Forward checking.

c. Min-conflicts algorithm. To initialize the problem place all four queens on the main diagonal of the board.