Problem Solving (First Partial Test)

November 18, 2022, Classroom Q1/1011, 17:00–19:00

No notes, no communication.
The **number between brackets** [ ] indicate the points of the exercise.
Write the answer to the questions and each problem in **a different piece of paper**.
Always **reason your answer**; otherwise, it will not be considered.
Do not forget **to write you name** and coordinates (row and column) in every page.

1. [**0.7**] Identify which of the following algorithms are complete:

   - DFS, BFS, ID
   - A*, IDA*, SMA*, RBFS, these four with an inadmissible heuristic

   in two cases: when the search space is finite or infinite (always with an finite branching factor b).

2. [**1**] Given an optimization problem P, there is an admissible heuristic for it but the computer where you have to process it has low memory. It is known that local search (hill climbing) works fine with P, if the objective function in the initial state is close to the optimum value. You have a few minutes to find a good-quality solution (no necessarily optimal). Reason what algorithm or sequence of algorithms you will use to solve P.

3. [**0.7**] Consider the IDA* algorithm. There is an the extreme case when, at each iteration, IDA* visits one more node than in the previous iteration. In this case: (i) regarding the algorithm behavior, what is recommended? (ii) compute the temporal complexity of IDA* for a binary tree.

4. [**0.8**] You want to solve an unknown problem P by A*, for which you manage to compute a consistent heuristic. If the state space of P is a tree, what is the best possible case and the worst possible case for A* execution? And when the search space of P is a graph?

5. [**0.8**] Consider the tic-tac-toe (tres en raya) game: you are the player X. Provide a heuristic (a procedure to compute a number for each free position of the board) that help you to play.

6. [**1**] In an execution of RTA* with an admissible heuristic, consider a node x that has been visited once by the algorithm. What can you deduce when, in this execution RTA*

   (a) does not visit again x;

   (b) visits x for second time, coming back from the path RTA* used to leave that node in the previous visit;

   (c) visits x for second time, coming back from a path different from the one RTA* used to leave that node in the previous visit.

Two hints:
The sum of all nodes of a complete binary tree of depth d is $2^{d+1} - 1$
The sum of the n first elements of an arithmetic progression $a_1, a_2, \ldots, a_n$ is $\frac{(a_1+a_n)n}{2}$

7. **[3]** The high school graduation party is coming. All students have to come in pairs. In order to simplify the always conflicting partner choice, we decided to automatize the process. Every student will write a secret list of acceptable partners into a program, that will assign a partner from the list to each student, if it is possible. We assume that there is a even number of students and we do not make any distinction among sexual orientation.

   Example:

Alex	**B**ernie, **C**asey
Bernie	**A**lex, **C**asey
Casey	**A**lex, **B**ernie, **D**enny
Denny	**A**lex, **B**ernie, **C**asey

   (Unique) solution:

   Alex - **B**ernie
   Casey - **D**enny

   Although this problem is a variation of the *perfect matching* problem, that can be solved in polynomial time, we have decided to use a SAT solver.

   (a) Let n be the even number of students, and $L_i \subset \{1, \dots, n\}$ be the set of possible partners selected by i. Describe what clauses would you generate to encode the problem. Describe clearly the set of variables and their intended "meaning".

   (b) Describe how many clauses your encoding would generate (as a function of n and the sizes $|L_i|$).

   (c) For the previous Example, write explicitly the set of clauses that your encoding would generate. Display also the (directed) graph representing the problem.

   (d) Notice that, if for some x and y, we have $y \in L_x$ but $x \notin L_y$ we can remove y from L_x, and the solution is unaffected.

   Apply this simplification to the lists of the Example and display the (undirected) graph representing the problem.

   Does this suggest you how to improve your encoding using less variables and clauses? If so, describe this new encoding.

   (e) In some problems, like the k-coloring, we can omit the typical AMO clauses that ensure that no two colors are assigned to a node. Can we do something similar in this problem? If yes, reason why. If not, could you find an example that would become solvable when it has no solution?

8. **[2]** For the following set of clauses and assuming that we assign preferably the variable with the smallest index to true:

```
p cnf 8 12
1 -2 4 0
1 2 -4 0
-1 2 0
-1 5 -6 0
2 4 0
-2 3 0
-2 -4 0
-2 5 6 0
-2 -5 -6 0
-2 7 8 0
-3 6 -7 0
6 -8 0
```

   (a) Display the search tree followed by the DPLL algorithm, writing the reason for each unary propagation

   (There are 3 decisions, hence 4 conflicts)

   (b) Represent the implication graph for the first conflict and compute the first and second UIP.