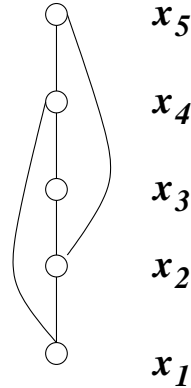


ICS 275, Assignment 5

1. (question 1, chapter 6) Let G be a constraint graph of a problem having 5 variables x_1, x_2, x_3, x_4, x_5 with arcs $(x_1, x_2), (x_2, x_3), (x_3, x_4), (x_4, x_1), (x_4, x_5), (x_5, x_2)$.



For a problem having the constraint graph G and using ordering $d_1 = (x_1, x_2, x_3, x_4, x_5)$, confirm, reject, and justify your answer either with an argument or a counter example:

- (a) Graph-based backjumping will always behave exactly as backtracking.
 - (b) Gaschnig's backjumping will always behave like backtracking.
 - (c) Conflict-directed backjumping and Gaschnig's backjumping are identical on G .
 - (d) Graph-based learning over G and ordering d_1 will never record constraints of size greater than two.
 - (e) If a leaf dead-end occurs at x_5 , what is the induced-ancestor set $I_5(\{x_5\})$? what is the induced ancestors $I_3(x_3, x_4)$? of $I_3(x_3, x_4, x_5)$.
 - (f) Propose a better ordering for graph-based jumping. Justify your answer.
 - (g) If a leaf dead-end occurs at x_5 and another internal dead-end at x_4 , what is the conflict-set recorded by graph-based learning?
 - (h) How would your answer on all previous questions be different if constraint (x_2, x_3) is omitted.
2. (question 5, chapter 6) Let m_d be the depth of a DFS tree of an induced graph along some ordering d_1 . Let d be its DFS ordering. a) prove that graph-based backjumping always jumps to an ancestor along the path from the root to the node (its parent or to its earlier variables on the path) in the DFS tree. (b) (**extra credit**) Prove that backjumping along d is bounded exponentially by m_d .
3. (question 7, chapter 6) Consider the crossword puzzle in Figure 2 as described in Chapter 2 formulated using the dual-graph representation. Using the ordering $(x_1, x_2, x_5, x_6, x_4, x_3)$.

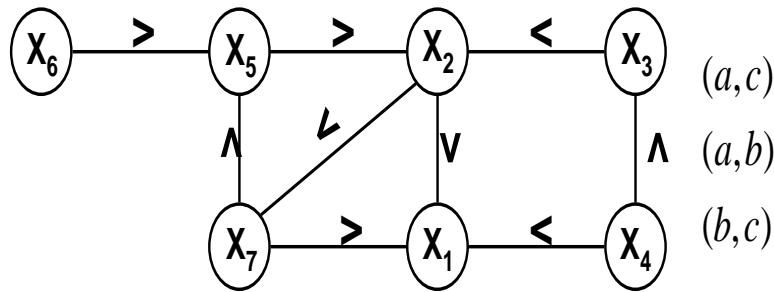


Figure 1: A constraint graph. The domains are a,b,c

1	2	3	4	5
		6		7
	8	9	10	11
		12	13	

Figure 2: A crossword puzzle

- (a) (optional) show a trace, whose length is limited to a 1-page description for a) Graph-based backjumping. b) Graph-based backjumping-learning. c) Jumpback-learning.
 - (b) Bound the complexity of graph-based backjumping on the crossword puzzle along the ordering above.
 - (c) Bound the complexity of graph-based backjumping-learning using that ordering.
4. (question 8, chapter 6) Consider the graph in Figure 1. a) Provide a DFS ordering of the graph and bound the complexity of solving any problem having that graph. b) analyze the complexity of solving this problem with backjumping and with learning.
 5. (question 13, chapter 6) Prove that when using the same variable ordering, Gaschnig's backjumping always explores every node explored by forward-checking.
 6. (question 12b,chapter 6) Analyze the overhead of each node generation in SAT-CBJ-LEARN.
 7. (extra credit) Do question 15 chapter 6 regarding the combinatorial auction only. (Can develop into a project)
 8. (Experimenting with SAT solvers) Download and compile the following sat solvers on your platform (windows or linux, if a solver does not compile on windows VC++ compiler, please download cygwin and use its g++ (or gcc) compiler). The link to these SAT solvers can be found on the class web page.

- (a) Minisat 2.0
- (b) WALKSAT
- (c) RSAT

We will use the *Langford problem SAT encoder* that you developed in the previous homework (Homework 4). If you remember, you encoded the Langford problem as a CNF instance. Compare the performance of the three SAT solvers on the SAT encodings of $L(2,n)$ where n is increased from 5 to 29 in increments of 2. Use a time-bound of 10 minutes i.e. kill the SAT solver if 10 minutes have elapsed (assuming that it did not terminate by itself). Report the results. Which solver was better and why?

(Hint: You can write a python (or shell or perl) script to run a SAT solver with a time-bound of 10 minutes. If you are using Linux, then you may find the *ulimit* command helpful. Do not turn in your script. We are interested in the results.)