CSE 505 HW2 Report
Enbo Yu
113094714
14 Oct

Q1:
p cnf n 1
p1 p2 … pn 0

Q2:
p cnf n n
-p1 0
-p2 0
…
-pn 0

Q3:
p cnf n k
p1 p2 … pk-1 pk 0
p2 p3 … pk pk+1 0
…
pn-k pn-k+1 … pn-1 pn 0

Q4:
see Q4.c
Test:
3 6 8
1 2
2 3
3 4
4 5
5 6
6 1
2 5
3 6
Output: SATISFIABLE

Q5:

1.
For all steps s ∈ [0, K], integer num (s, i, j) ∈ [0, N × N − 1] denoting which number or label the tile contains at step s.
2.
For all space(i, j) : num (0, i, j) is the number of the (i, j) in the initial configuration. The empty tile is numbered as 0.
3.

Integers $d_i(s), d_j(s) \in \{-1, 0, 1\}$, respectively denoting the vertical and horizontal movement of the empty tile during the transition from step s to step s + 1.

4.

For all steps $s \in [0, K]$, a Boolean goal(s) denoting whether the puzzle has been solved at step s or at an earlier step.

5.

Integers $empty_i(s) \in [0, N-1]$ and $empty_j(s) \in [0, N-1]$, respectively denoting the vertical and horizontal coordinates of the empty tile at step s.

6.

$(num(s, i, j) = 0) \Leftrightarrow (empty_i(s) = i \wedge empty_j(s) = j)$. This synchronizes the two ways of tracking the location of the empty tile.

7.

For all steps $s \in [0, K] : (steps \leq s) \rightarrow (goal(s) = 1)$

8

For all steps $s \in [0, max - 1] : d_i(s) = 0 \vee d_j(s) = 0$. This enforces that the player only makes orthogonal moves.

9.

$goal(s) = 1 \Leftrightarrow (d_i(s) = 0 \wedge d_j(s) = 0)$. This enforces that no more moves are made once the puzzle has been solved.

10.

$(d_i(s) = 0 \wedge d_j(s) = 0) \rightarrow (empty_i(s) = empty_i(s+1) \wedge empty_j(s) = empty_j(s+1))$. This enforces that, if one has not moved, the empty tile should not have moved.

11.

For the moves $(i, j) \in \{(1, 0), (-1, 0), (0, 1), (0, -1)\} : (d_i(s) = i \wedge d_j(s) = j) \rightarrow (empty_i(s) + i \geq 0 \wedge empty_i(s) + i \leq N \wedge empty_j(s) + j \geq 0 \wedge empty_j(s) + j \leq N \wedge empty_i(s) + i = empty_i(s+1) \wedge empty_j(s) + j = empty_j(s+1))$. This enforces that the player only makes valid moves and that the empty tile actually moves.

12.

$\neg((empty_i(s) = i \wedge empty_j(s) = j) \vee (empty_i(s+1) = i \wedge empty_j(s+1) = j)) \rightarrow (num(s, i, j) = num(s+1, i, j))$ .This enforces that, if the tile was not the empty tile during either one of the two steps, its contents should be unchanged.

13.

For all steps $s \in [1, K-1] : d_i(s-1) = 1 \rightarrow d_i(s) != -1, d_i(s-1) = -1 \rightarrow d_i(s) != 1, d_j(s-1) = 1 \rightarrow d_j(s) != -1, d_j(s-1) = -1 \rightarrow d_j(s) != 1$. The number of possible moves at every step reducing the size of the search space.

14.

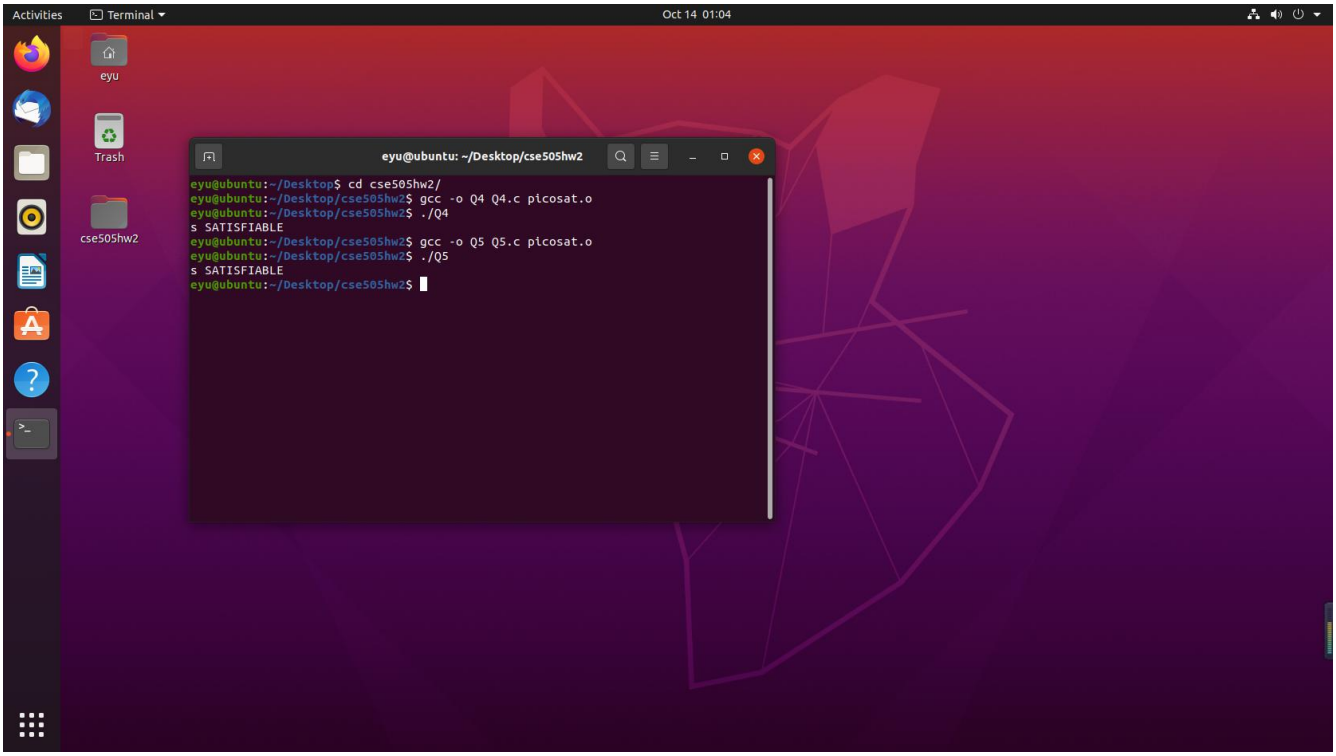
See Q5.c

Test:

```
4    100
15 2 1 12
8 5 6 11
4 9 10 7
3 14 13 0
```

Output:

SATISFIABLE


Activities
Terminal
Oct 14 01:04
eyu
Trash
cse505hw2
eyu@ubuntu: ~/Desktop/cse505hw2
eyu@ubuntu:~/Desktop$ cd cse505hw2/
eyu@ubuntu:~/Desktop/cse505hw2$ gcc -o Q4 Q4.c picosat.o
eyu@ubuntu:~/Desktop/cse505hw2$ ./Q4
s SATISFIABLE
eyu@ubuntu:~/Desktop/cse505hw2$ gcc -o Q5 Q5.c picosat.o
eyu@ubuntu:~/Desktop/cse505hw2$ ./Q5
s SATISFIABLE
eyu@ubuntu:~/Desktop/cse505hw2$