**CS1200: Intro. to Algorithms and their Limitations** Anshu & Vadhan

# Problem Set 5

*Harvard SEAS - Fall 2024* *Due: Wed Oct. 23, 2024 (11:59pm)*

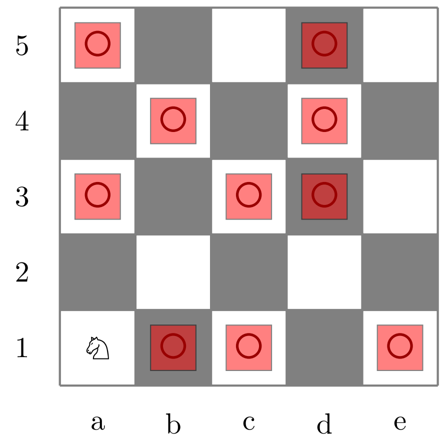
**Your name:**
**Collaborators:**
**No. of late days used on previous psets:**
**No. of late days used after including this pset:**

*Remember to mark your pages on Gradescope properly, or points will be taken off. Additionally, if your handwriting isn't particularly neat please submit written proofs, which are much easier to grade for the TFs.*

1. (Solving Games) Consider playing a solo game on an $n \times n$ chessboard. You have one piece, a chess knight, which starts in the lower-left corner, and your goal is to reach any of the other three corners in as few moves as possible. Like a usual chess knight, in one move, you can move to any position that is two squares away in a horizontal direction and one square away in a vertical direction, or two squares away in a vertical direction and one square away in a horizontal direction. There is a catch, however: some squares have visible landmines, so you cannot move to them (since you do not want set off an explosion).

   (a) Give an algorithm that achieves the above goal in time $O(n^2)$ by reduction to either the ShortestWalks problem or the SingleSourceShortestPaths problem. The algorithm should output $\perp$ if no sequence of moves can take you to any of the other three corners when started in the lower-left corner. (Note: if reducing to SingleSourceShortestPaths, we haven't defined abstractly what it means to reduce a computational problem to a data structure problem, but you may construct and use a SingleSourceShortestPaths data structure on an appropriately constructed graph.)

   (b) Carry out your algorithm on the $5 \times 5$ board shown below, listing both the frontier vertices and the predecessor relationships at each stage of BFS (the red square symbols are landmines).



2. (Maximal Independent Sets) Let $G = (V, E)$ be a graph. A set $S \subseteq V$ is a *maximal* independent set if we cannot add any vertices to $S$ while it remains an independent set. That is, for every vertex $v \in V \setminus S$, $S \cup \{v\}$ is not an independent set.

   (a) Show that given a graph $G$, a maximal independent set can be found in time $O(n + m)$. Note that this is in sharp contrast to *maximum-size* independent sets, for which we do not know any subexponential-time algorithms. (Hint: be greedy.)

   (b) Show that if $G$ is 3-colorable, then it has a 3-coloring $f$ in which the set of vertices of color 2 (i.e. $f^{-1}(2)$) is a maximal independent set.

   (c) It is known that every graph $G$ has at most $3^{n/3}$ maximal independent sets, and there is an algorithm (the Bron-Kerbosch algorithm) that enumerates all of the maximal independent sets in time $O(3^{n/3})$. Use this fact to conclude that 3-coloring can be solved in time $O((n + m) \cdot 3^{n/3}) = O(1.44^n)$, improving the runtime of $O(1.89^n)$ from SRE4.

3. (Exponential-Time Coloring) In the Github repository for PS5, we have given you basic data structures for graphs (in adjacency list representation) and colorings, an implementation of the Exhaustive-Search $k$-Coloring algorithm, an implementation of the Bron-Kerbosch algorithm, and a variety of test cases (graphs) for coloring algorithms.

   (a) Implement the $O(n + m)$-time algorithm for 2-coloring that we covered in class in the function `bfs_2_coloring`, verifying its correctness by running `python3 -m ps5_tests 2`. Your implementation of BFS should follow the presentation and notation that we used in class (with the loop over distance $d$ and the sets $F$ and $S$), which may be different than presentation of BFS in other sources (online or in the optional textbooks).

   (b) Implement the $O((n+m) \cdot 3^{n/3})$-time algorithm for 3-coloring (MaximalIS + BFS) from Problem 2 above in the function `iset_bfs_3_coloring`, also verifying its correctness by running `python3 -m ps5_tests 3`.

   (c) Compare the efficiency of Exhaustive-Search 3-coloring and the $O((n + m) \cdot 3^{n/3})$-time algorithm. Specifically, identify and write down the largest instance size $n$ each algorithm is able to solve (within a time limit you specify, e.g. 1 second) and the smallest instance size $n$ each algorithm is unable to solve (again within that same time limit).

   In addition to these numeric values, please provide a brief explanation of why these results make sense, based on your knowledge of both the algorithms' runtime and how each algorithm goes about finding a coloring. For this part, there is no need to go through every combination of parameters; feel free to give just the largest and smallest instances each algorithm can solve and speak generally as to why one algorithm performs better than the other. More instructions can be found in `ps5_experiments`.

4. (Reflection): Take one homework problem you have worked on this semester that you struggled to understand and solve, and explain how the struggle itself was valuable. In the context of this question, describe the struggle and how you overcame the struggle. You might also discuss whether struggling built aspects of character in you (e.g. endurance, self-confidence, competence to solve new problems) and how these virtues might benefit you in later ventures.

   *Note: As with the previous psets, you may include your answer in your PDF submission, but the answer should ultimately go into a separate Gradescope submission form.*

*Quick note on grading: Good responses are usually about a paragraph, with something like 7 or 8 sentences. Most importantly, please make sure your answer is specific to this class and your experiences in it. If your answer could have been edited lightly to apply to another class at Harvard, points will be taken off.*

5. Once you're done with this problem set, please fill out this survey so that we can gather students' thoughts on the problem set, and the class in general. It's not required, but we really appreciate all responses!