Wordle is NP-hard

Bernardo Subercaseaux

January 10, 2022

1 Introduction

Wordle (https://www.powerlanguage.co.uk/wordle/) is a one player word game that combines principles of the classic games Mastermind and Hangman, both of which have been studied from a complexity perspective [1, 2, 3, 4]. Wordle is played over a dictionary of words D, for example that of the English language, that is fully known to the player. As a word length is fixed for a game of Wordle, we assume all words in D have the same length k. The player, who we will denote the *guesser*, wishes to discover a secret word $w \in D$ through a series of at most ℓ guesses $p_1, p_2, \ldots, p_\ell$, all of which must be words belonging to D. Whenever a guess p is made, the guesser receives some information about the relation between p and the secret word w: she is notified of every position i such that $p[i] = w[i]$, and also of every position i such that $p[i]$ is present in the word w but not in position i. A game of Wordle is illustrated in Figure 1, using color green to denote positions where $p[i] = w[i]$ and yellow for positions such that $p[i] \in w, p[i] \neq w[i]$. The guesser is said to win if one of its guesses p_i is exactly equal to w, and she loses if no such match occurs after ℓ guesses.

A careful reader might notice that we are not being fully precise with respect to some corner cases; what if a guess contains two occurrences of a letter ℓ, which appears only once in w? does the guesser receive information that is any different from the case where ℓ appears twice in w? We conveniently skip such considerations as our proof is independent of them.

2 Computational Hardness

First, let us define the computational problem associated to Wordle as a decision problem.

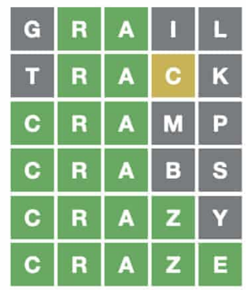


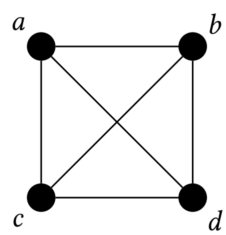
Figure 1: Illustration of a game of Wordle in English, where the secret word is `CRAZE`, and symbols in a guess that match the secret word are colored green, while symbols that appear in the secret word in a different position are colored yellow.

PROBLEM :	Wordle
INPUT :	(D, ℓ) with D a finite dictionary over an alphabet Σ in which every word has length k, and an integer $\ell \geq 1$, the maximum number of guesses.
OUTPUT :	Yes, if the guesser can guarantee a win the associated game of Wordle, and No otherwise.

Theorem 1. Wordle *is* NP*-hard.*

The proof is almost analogous to that of coNP-hardness of EvilHangman by Barbay and Subercaseaux [1]. We will reduce from Dominating Set on 3-regular graphs, which was proved to be NP-hard by Kikuno et al. [5].

Proof of Theorem 1. Let (G, d) be an input for Dominating Set, with G a 3-regular graph and d an integer such that we need to decide whether G contains a dominating set of size at most d. We will work over word length $k = 4$. The alphabet Σ will be the vertex set of G. We build a dictionary D_G from G as follows: for every vertex $v \in G$, create a word w_v whose first symbol is v, and whose three remaining symbols are the neighbors of v. As we have not yet specified an ordering for the neighbors of v in the word w_v, many encodings for D_G are possible. We will need encodings to satisfy an additional property; that for each symbol $v \in V(G)$ and index $i \in \{1, 2, 3, 4\}$, there is exactly one word $w \in D_G$ such that $w[i] = v$. An encoding D_G that satisfies this property will be called a proper encoding. An encoding and a proper encoding for K_4 are illustrated in Figure 2a. We will use



1. w_a = abcd
2. w_b = bacd
3. w_c = cabd
4. w_d = dabc

1. w_a = abcd
2. w_b = badc
3. w_c = cdba
4. w_d = dcab

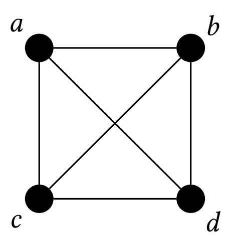
(a) The graph G to encode (b) An encoding of G (c) A proper encoding of G

Figure 2: Example of encodings for K_4

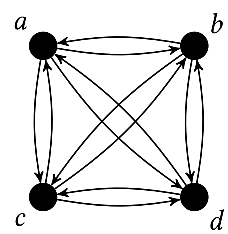
that a proper encoding of G can be computed in polynomial time, which is proved by Barbay and Subercaseaux [1], and also included in this article as Lemma 1 for completeness. Now, assume a proper encoding D_G has been computed. We claim that $(D_G, d+3)$ is a positive instance of Wordle if, and only if, (G, d) is a positive instance of Dominating Set. For the forward direction, assume $(D_G, d+3)$ is a positive instance of Wordle. This implies that there is a guessing strategy that guarantees knowing w after $d+3$ guesses, which in turn implies that there is a strategy that guarantees obtaining at least one symbol of w in the first d guesses, as in the worst case 3 guesses are required to deduce the word from a particular symbol, using the fact that the encoding D_G is proper. As every guess w can be uniquely associated to a vertex $v \in V(G)$ by mapping it to its first symbol, we have that there is a sequence S of d vertices such that any vertex v in $V(G)$ is either in S or has a vertex of S in its corresponding word w_v, meaning v is in the neighborhood of a vertex in S. This implies S is a dominating set in G of size d. For the backward direction, assume G has a dominating set S of size at most d and the secret word is w_u for some $u \in V(G)$. Now consider the sequence of guesses $w_v, v \in S$. If u is in S then we are done, and otherwise, as S is a dominating set, then u must be a neighbor of somme vertex v^* in S, which implies that after the sequence of guesses we know that u is a neighbor of v^*. As v^* has degree 3, the guesser can use now the sequence of words $w_{u'}$ for $u' \in N(v^*)$, which of course contains w_u as u is a neighbor of v. This guessing strategy has $d+3$ guesses and it is guaranteed to succeed. This concludes the proof. □

Lemma 1. *Given a 3-regular graph G, one can compute a proper encoding for it in polynomial time.*

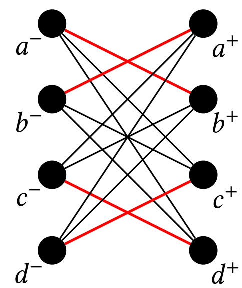
Proof. Let $G = (V, E)$ be a 3-regular graph, we start by considering the digraph



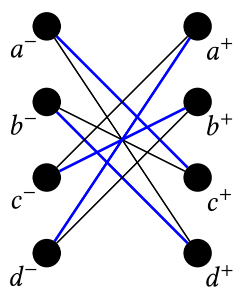
(a) The graph G to encode



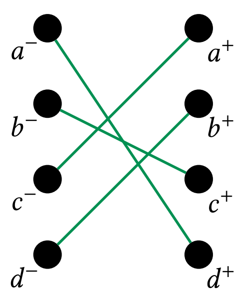
(b) Its associated digraph D



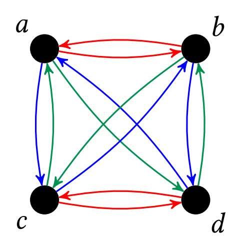
(c) Associated bipartite graph B, with a perfect matching



(d) A second perfect matching



(e) Third perfect matching



(f) A proper edge coloring of D based on the matchings

Figure 3: Illustration for the proof of Lemma 1 on K_4. Note that the encoding resulting of subfigure (f) corresponds to the one presented in Figure 2c.

$D = (V', E')$ associated to G where $V' = V$ and E' contains the pairs (u, v) and (v, u) if there was an edge between nodes u and v in G. We claim that if there exists a way to color edges in D with {`red`, `blue`, `green`} such that every vertex has (i) incoming edges of each different color, and (ii) outgoing edges of each different color, then we can produce a proper encoding based on that. Here's how to do it: if vertex u has a red outgoing edge to v, a green outgoing edge to w and a blue outgoing edge to x, then we can encode it as `uvwx`. Note that the color of an edge $u \to v$ determines in which position is v going to appear in the encoding of u, and therefore condition (i) over v ensures that the label of v appears in every position, while condition (ii) over u ensures that no more than one vertex is assigned position p on the encoding of u.

In order to find such an edge coloring, we create the undirected bipartite graph $B = (V'', E'')$, where for every vertex $v \in V$, we put two vertices v^+ and v^- in V'', and for every edge (u, v) in E' we put the edges (u^+, v^-) and (u^-, v^+). The partition of B is then, of course, the set of vertices $(\cdot)^+$ and the set of vertices $(\cdot)^-$. Note that B is also a 3-regular graph, as every vertex v with neighbors u, w and x in the original graph, it is associated vertex u^+ is connected with v^-, w^- and x^- in B, and u^- will be connected to u^+, w^+ and x^+.

As a direct consequence of Hall's Marriage Theorem [6], every regular bipartite graph has a perfect matching. Such a perfect matching can be computed in polynomial time using for example the Hopcroft-Karp algorithm [7]. Let M be the set of edges of a perfect matching computed that way. We can color every edge in M with red. Now, if we remove from E'' all the edges of M, the bipartite graph is 2-regular, as each node has lost exactly one neighbor. By using Hall's theorem again, we can get a new perfect matching M', whose edges we color with blue. If we now remove all the edges of M'', we get a 1-regular graph, which is itself a perfect matching, and whose edges we color with green. This is enough to get the required coloring in the graph D, just by coloring every edge (u, v) with the same color of the edge (u^-, v^+).

□

References

[1] Jérémy Barbay and Bernardo Subercaseaux. 2020. The computational complexity of evil hangman. In *10th International Conference on Fun with Algorithms (FUN 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[2] Jeff Stuckman and Guo-Qiang Zhang. 2005. Mastermind is np-complete. (2005). arXiv: `cs/0512049 [cs.CC]`.

[3] Giovanni Viglietta. 2012. Hardness of mastermind. In *Fun with Algorithms*. Evangelos Kranakis, Danny Krizanc, and Flaminia Luccio, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 368–378. ISBN: 978-3-642-30347-0.

[4] Michael T. Goodrich. 2009. On the algorithmic complexity of the mastermind game with black-peg results. *Information Processing Letters*, 109, 13, 675âĂŞ678. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2009.02.021. http://dx.doi.org/10.1016/j.ipl.2009.02.021.

[5] Tohru Kikuno, Noriyoshi Yoshida, and Yoshiaki Kakuda. 1980. The np-completeness of the dominating set problem in cubic planer graphs. *IEICE TRANSACTIONS (1976-1990)*, 63, 6, 443–444.

[6] P. Hall. 1935. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10, 1, (January 1935), 26–30. DOI: 10.1112/jlms/s1-10.37.26. https://doi.org/10.1112/jlms/s1-10.37.26.

[7] John E. Hopcroft and Richard M. Karp. 1973. An \$n^5/2\$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2, 4, (December 1973), 225–231. DOI: 10.1137/0202019. https://doi.org/10.1137/0202019.