Chapter 1

Shortest Odd Path

1.1 Intuition

Now that we have tried out some algorithms for Shortest Odd Walk, we are finally ready to add the restriction that each vertex is used at most once, and thus solve Shortest Odd Path. The algorithm we are about to present is heavily based on Ulrich Derig's algorithm from 1985, though with some improvements.

1.1.1 Reduction to Shortest Alternating Path

Consider first another related problem:

Shortest Alternating Path
Input: A weighted graph $G := (V, E)$, two vertices $s, t \in V$, and a set $F \subseteq E$
Output: the shortest s-t-path in G where every other edge used is in F.

Derig observed that Shortest Odd Path can be reduced to a special case of Shortest Alternating Path, by constructing what we will refer to as a *mirror graph*.

Definition 1.1.1 (Mirror graph)**.** Let $G = (V, E)$ be a graph, and $s, t \in V$ be two vertices. We construct a new graph $H \supset G$, where for each vertex $u \in V \setminus \{s, t\}$ we add a 'mirror' vertex u', and a connecting 'mirror' edge between them. The vertices in $V(H)$ that are also in $V(G)$ are referred to as the 'real' vertices, and the newly added vertices are referred to as the 'mirror' vertices. In addition, for any vertex $u \in V(H) \setminus \{s, t\}$, real or not, we define $mirror(u)$ as u's mirror on the other side. We usually label mirror vertices with an $'$ at the end of the real counterpart's label. For example, if G is the graph in Figure 1.1, then Figure 1.2 would be its corresponding mirror graph H.

Our reduction from Shortest Odd Path to Shortest Alternating Path follows:

1. Let (G, s, t) be an instance of Shortest Odd Path.
2. Construct H as the mirror graph of G, and let F be the set of mirror edges in H. Now (H, s, t, F) is an instance of Shortest Alternating Path.
3. Let P' be the shortest alternating path of (H, s, t, F), if one exists. If none exist, then we do not have any odd s-t-paths in G either.

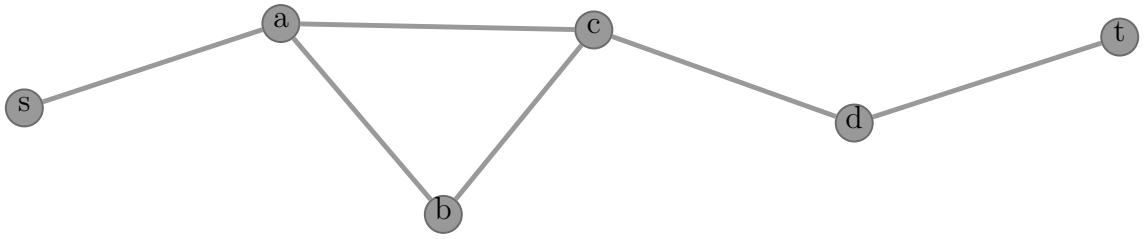

Figure 1.1: Our input graph G, for Shortest Odd Path

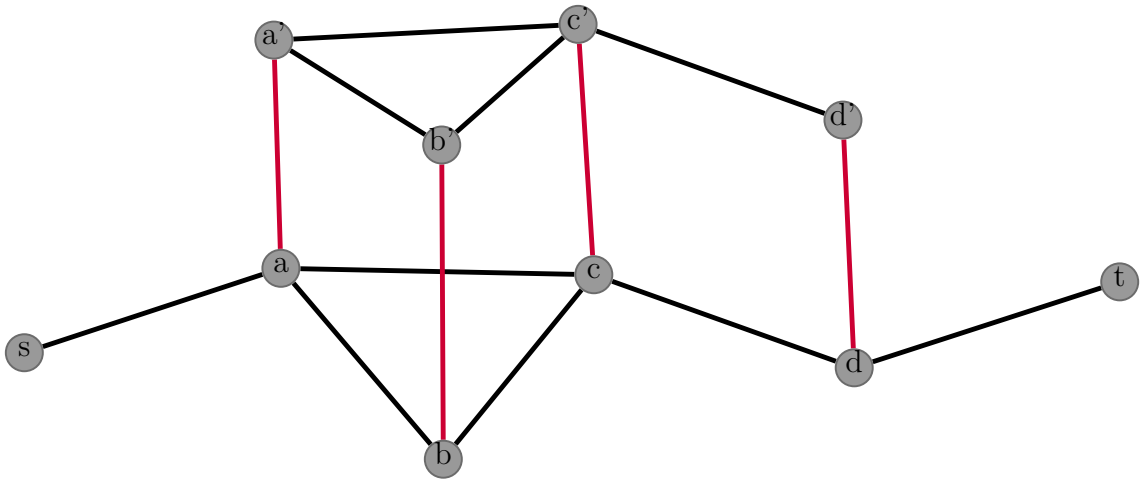

Figure 1.2: The mirror graph H of G, with mirror edges marked in red and mirror vertices labeled with an $'$

4. Construct P by filtering out mirror edges from P', and for each edge $(u', v') \in E(H) \setminus (F \cup E(G))$ from the mirror side of H we replace it by the corresponding edge $(u, v) \in E(G)$ from the real side.

5. Now P is the shortest odd s-t-path in G.

For example, if our input G for Shortest Odd Path is Figure 1.1, then H and F could look like Figure 1.2. One of the two possible alternating paths is $P' := [(s, a), (a, a'), (a', b'), (b', b), (b, c), (c, c'), (c', d'), (d', d), (d, t)]$. When we filter out mirror edges and replace edges from the mirror side with their real counterparts, we end up with $P := [(s, a), (a, b), (b, c), (c, d), (d, t)]$, which is one of the two possible odd paths of G.

TODO dette kan kanskje formuleres bedre?

To see why the reduction works, simply observe that for each step we take in the graph, we have to go to the other side of the mirror. If we take another step, we get back to the same side again. It is only when we reach the target vertex t that we do not have

to go to the other side. Therefore, to reach a neighbour of t, we must have used an even number of mirror edges and an even number of non-mirror edges, and when we take the last step to reach t we have used an odd number of edges and thus found an odd path. If this alternating s-t-path in H is the shortest such path, then the corresponding path in G must also be the shortest odd s-t-path in G. The reduction works also in the weighted case, as long as each edge (u', v') on the mirror side get the same weight as their real counterpart, and all the mirror edges get the same (usually 0) weight.

TODO refer to Derig for fewer and more convoluted details

1.1.2 Adapting previous work

TODO refer to M.O. Ball and U. Derigs, An analysis of alternate strategies for implementing matching algorithms

TODO

Bibliography

Appendix A

Generated code from Protocol buffers

Code Listing A.1: Source code of something

```
1 System.out.println("Hello Mars");
```