ASYMPTOTICALLY FASTER CIRCUIT TOPOLOGIES

Michael You*
youmichaelc@gmail.com

April 8, 2021

ABSTRACT

We are familiar with using the Turing paradigm to create computers, representing information in binary bits. However, there are more natural ways to solve problems without using bits. For example, if you have a graph, why not just find a natural way to represent it so the computation can be done better? That's the motivation of the paper. We will explore designing more natural circuit topologies to solve problems, and analyze their computational complexity. We will find that for sorting, we can come up with an algorithm that is faster than existing algorithms, and shortest paths in graphs that we can perform asymptotically better than existing algorithms. We will then explore the possibility of extending more natural circuit topologies to solve other problems faster, and applications of using this circuit topology technique to solve algorithmic problems in general.

Keywords Computational Complexity · Circuits · Diodes · Graphs

1 Introduction

Turing machines

efficient for information storage, but is it the best way to compute things? Best way to represent everything?

There can be more natural ways to represent problems.

2 Definitions

Definition 1. An **ideal diode** is a circuit element that has infinite current when on, and 0 current when off. The threshold voltage[2] for when it is on and off is called V_{D} and is measured from the $+$ to the $-$ terminal.

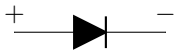


Figure 1: A diode, with the positive and negative terminals marked.

Property 1. For a chain of diodes $D_1, D_2, \ldots, D_n$ in series, the turn on voltage for the chain is

$$V_D = \sum_{i=1}^{n} V_{D_i} \tag{1}$$

*Github: mikinty

[2]we sometimes call this the "turn-on voltage"

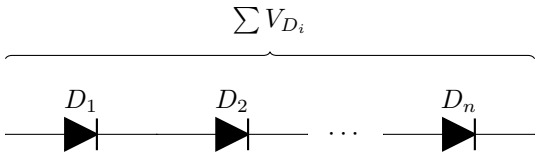


Figure 2: Diode chain

Definition 2. A **voltage source** is a circuit element that maintains a voltage of V between 2 nodes.

In this paper, we will be using the following element to represent a voltage source.

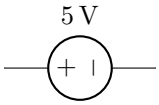


Figure 3: A voltage source

Definition 3. An **ammeter** is a device used to measure the current at some node of a circuit. We will use the notation

$$I(A_i) \tag{2}$$

to describe the current measured by ammeter A_i.

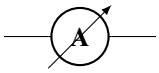


Figure 4: An ammeter

Definition 4. A **switch** is a device that can be set to 2 configurations, either closed, which behaves as a wire in a circuit, or open, which will behave as an open circuit, so no current can flow through.

Figure 5: An open switch

3 Circuit Topologies

Here, we will explore two circuit topologies, their computational complexity in the problem they solve, and their costs.

- Sorting numbers
- Shortest path in a graph

3.1 Sorting

We define the sorting problem to be

Given a list of n numbers,

The fastest algorithms for sorting numbers are

- **QuickSort:** $O(n \log n)$
- **Radix Sort:** $O(b(n+k))$

We will present a circuit that runs in $O(n)$ time and has $O(n)$ space complexity.

3.1.1 Circuit

Given an array with numbers

$$[x_1, x_2, x_3, \ldots, x_n],$$

Get diodes D_i of turn-on voltage $V_{D_i} = x_i$, and construct the following circuit:

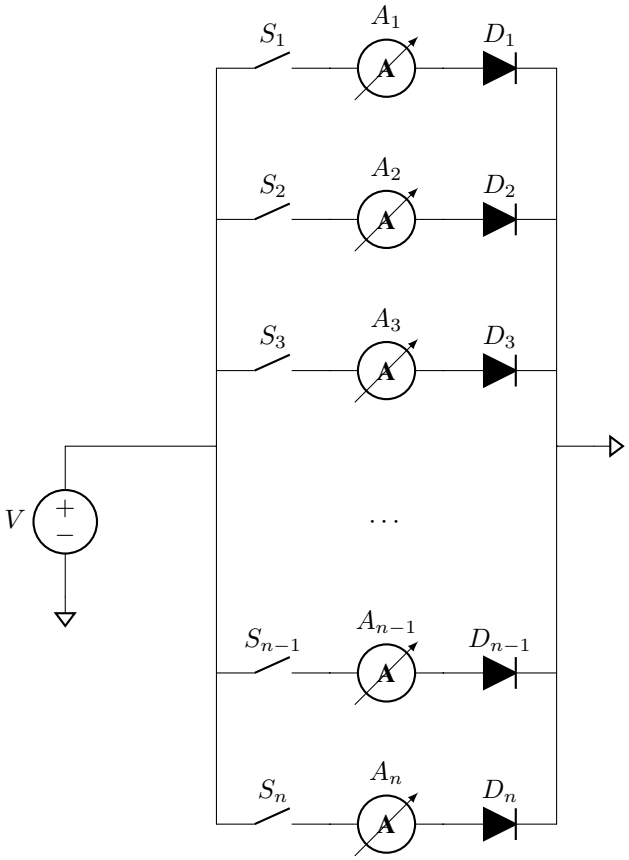


Figure 6: Circuit for sorting numbers

3.1.2 Algorithm

```
SORT:
  // Build circuit
  Get a voltage source V, with nodes A at the positive, B at the negative
  for i = 1 : n:
    Build a series circuit of a switch S_i, ammeter A_i, and diode D_i
    and put it between A, B

  // Sort
  queue Q = []
  set V = V_max
  while ∃S_i that is on:
    for i such that I(A_i) > 0:
      turn off S_i
      Q.push(V_i)

  return Q
```

We will now prove this algorithm is correct and then prove its time and space complexity.

First, we will prove the following lemma,

Lemma 1. If there are diodes $D_1, D_2, \ldots, D_n$ in parallel, and a voltage of $V \geq \max(V_{D_1}, V_{D_2}, \ldots, V_{D_n})$ is applied across these diodes, the voltage across the diodes is $V_{\text{min}} = \min(V_{D_1}, V_{D_2}, \ldots, V_{D_n})$, and only diodes D_i such that $V_{D_i} = V_{\text{min}}$ are on.

Proof. We will prove this statement by contradiction.

Suppose some other D_k where $V_{D_k} > V_{\text{min}}$ is on. Then the voltage across the diodes is V_{D_k}. But since we know that $V_{D_k} > V_{\text{min}}$, it must be the case that some D_i with $V_{D_i} = V_{\text{min}}$ is also on. However, if D_i is on, then the voltage across the diodes is $V_{D_i} < V_{D_k}$, and therefore D_k cannot be on. We have reached a contradiction, and therefore only diodes D_I with $V_{D_i} = V_{\text{min}}$ can be on. □

Now, the proof for the sorting algorithm is simple,

Theorem 1. *Algorithm 3.1.2 correctly sorts diodes $D_1, D_2, \ldots, D_n$ from smallest to greatest threshold voltage V_{D_i}.*

Proof. Once we have built our circuit, since $V = V_{\text{max}}$, we know at least one diode is turned on, as long as $\exists S_i$ that is on. For the diodes that are on, or have $I(A_i) > 0$, they will only be the diodes that are on, and these diodes, by Lemma 3.1.2, are the ones with the smallest threshold voltage in the set of diodes.

Now, we can show this sorting algorithm works.

- Initially, the queue Q is empty, so it is trivially sorted
- On an arbitrary iteration with Q sorted, we only add diodes with V_{D_i} the smallest in the set of remaining diodes with their switch on. When we add these diodes to the queue, V_{D_i} must be larger than the previous diode V_{D_j} added, since when D_j was added, D_i was not on. Therefore, Q remains sorted in each iteration.
- On the last iteration, we are left with diodes that have a larger threshold voltage than all diodes currently in Q, so adding these last diodes to Q still keeps Q sorted.

□

3.1.3 Complexity

Theorem 2. *Algorithm 3.1.2 runs in $O(n)$ time and uses $O(n)$ space.*

Proof. Building the circuit takes 1 operation for putting the voltage source, and $3n$ operations for attaching the switch, ammeter, and diode in parallel.

This circuit contains

$$1 + 3n = 3n + 1 \in O(n) \tag{3}$$

components, so it is $O(n)$ space.

Then, to run the algorithm, we are just turning off S_i one by one, n times, and each time adding V_{D_i} to our queue, which is a total of 2 operations.

Therefore, the total number of operations is

$$1 + 3n + 2n = 5n + 1 \in O(n). \tag{4}$$

□

Notice that although the complexity is $O(n)$, which matches radix sort, the constant $kn \in O(n)$ is for a $k \approx 2$, which outperforms radix sort.

3.2 Shortest Path

A natural extension to the sorting circuit is to create graphs out of diodes, and then look for shortest paths.

The shortest path problem on a weighted graph is given a graph, what is the shortest path from V_i to V_j. Finding the shortest path between every pair of vertices can be solved with the Floyd-Warshall algorithm in $O(|V|^3)$ time, with $O(|V|^2)$ space complexity.

We will present a circuit that uses $O(|V|)$ space complexity, along with an algorithm that runs in $O(|V|^2)$ time.

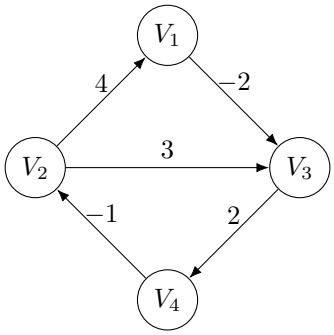
3.2.1 Circuit

To construct the circuit, given a graph $G = (V, E)$,

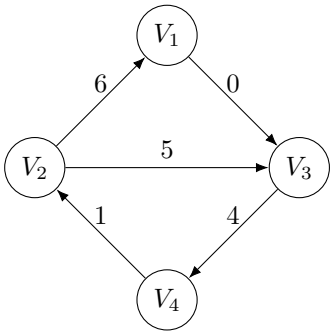
```
BUILD_DIODE_GRAPH(G = (V, E)):
  Normalize G so that there are no negative weights, producing G' = (V, E')

  For e_i = (V_j, V_k, w_i) ∈ E':
    Add A_i, D_i between V_j, V_k such that V_{D_i} = w_i
```

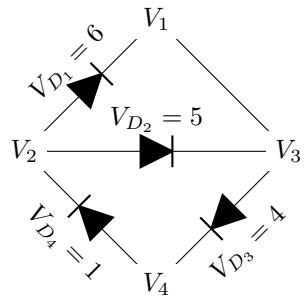
For example, with the following graph, we can construct the corresponding diode graph as follows:



(a) Original graph



(b) Normalized graph



(c) Diode graph, ammeters A_i are in series with D_i, but not shown to reduce clutter in the diagram

3.2.2 Algorithm

```
SHORTEST_PATH_ALL_PAIRS(G):
  BUILD_DIODE_GRAPH(G)

  shortest_paths = {}
  Set V = ∑ V_{D_k}
  For every pair of vertices V_i, V_j:
    Put voltage source V with the positive terminal at V_i and negative terminal at V_j
    The I(A_k) > 0 are the edges e_k that are in the shortest path
    Add this path to shortest_paths(i, j)
```

Theorem 3. *Algorithm 3.2.2 correctly finds the shortest path between every V_i, V_j.*

Proof. There are a set of paths P_k between V_i, V_j, in the form

$$P_k = \left[D_{k_1}, D_{k_2}, \ldots, D_{k_{n_k}}\right]. \tag{5}$$

Using Property 1, when diodes are in series, their turn on voltage is just the sum of all of the diodes' turn on voltages. So we can express the turn-on voltages for all of these P_k paths as

$$V_{\text{turn on}_k} = \sum_{i=1}^{n_k} V_{D_{k_i}} \tag{6}$$

Now, when we apply V_{max} across V_i, V_j, the diode series with the lowest turn-on voltage will turn on. Since diode turn on voltages correspond to the edge lengths on the graph, the smallest turn-on voltage path corresponds to the shortest path.

If there is no such diode series that turns on, then there is no path between V_i, V_j.

Therefore, in every iteration we are finding the shortest path between V_i, V_j. □

3.2.3 Complexity

Theorem 4. *Algorithm 3.2.2 runs in $O(|V|^2)$ time and uses $O(|V| + |E|)$ space complexity.*

Proof. We can find the time and space complexity of the algorithm by considering the construction and the path searching.

Construction. To normalize the weights of G, we just have to add the minimum edge weight in G to all of the edges, which takes $|E|$ operations.

Then, to build the graph, we have to connect

- $|E|$ diodes
- $|E|$ ammeters
- $|E|$ ammeters with wires to the device reading the ammeter
- $|V|$ junctions

Therefore, construction takes

$$|E| + 3|E| + |V| \in O(|V| + |E|) \tag{7}$$

time and space complexity.

Path search. Once we have our graph, in order to find all shortest paths, all we have to do is apply V_{max} across every node pair V_i, V_j. Every application of this max voltage will instantly reveal the shortest path, i.e. $O(1)$ path search. To find every shortest path, we just have to apply this max voltage $|V| \cdot (|V| - 1)$ times, corresponding to the number of V_i, V_j pairs where order matters, but $i \neq j$.

This will take

$$|V|(|V| - 1) \in O(|V|^2) \tag{8}$$

time.

Therefore, combining Equations (7, 8), we see that Algorithm 3.2.2 runs in $O(|V|^2)$ time and uses $O(|V| + |E|)$ space complexity. □

4 Real-Life Implementations of Circuits

To demonstrate that these ideas can work in practice, I built these circuits in real life and tested them out.

Comparison to existing CPUs.

5 Discussion

How do we get diodes of arbitrary threshold voltage? This is unlikely, and can vastly limit the scopes of the problems we'd like to solve.

Physical limitations with electrons traveling through a wire. We have been modeling electron travel through wire as instantaneous, which technically isn't true by physics, since the electrons will move in the wire at the speed of light. Because we have finite speed, we should consider the size of the circuits being built, as they can have an asymptotic complexity effect on their computational efficiency.

5.1 Other Problem

What other problems can we solve with this idea? The idea of this paper is to show that certain "intuitive" circuit topologies can solve problems faster than we might using a Turing computation model, where we represent our problems in a particular data structure, and perform operations on this data structure representation.

Can we do ray tracing?

References