FAST SORTING AND SHORTEST PATH ALGORITHMS WITH DIODE CIRCUITS

Michael You*
youmichaelc@gmail.com

July 10, 2022

ABSTRACT

We present a set of solutions to sorting numbers and finding all shortest paths in a directed, weighted graph, by using diodes, instead of traditional Turing computational model methods. We see that the solutions are able to produce good time complexity results, derive a solution for $O(n)$ time and space for sorting n numbers and $O(|V|^2)$ time and $O(|V| + |E|)$ space for finding the shortest path between all pairs of vertices in a graph. However, the cost of producing diodes with large weights is potentially an exponentially expensive resource relative to input sizes, which may lend to why we are able to solve problems so quickly.

Keywords Computational Complexity · Circuits · Diodes · Graphs · Sorting · Shortest Path

1 Introduction

Most computer science algorithms rely on the Turing Machine model of computation (Turing et al. [1936]), where we encode information with bits, and we can read or write information in some sort of memory. For practical purposes, we use Von Neumann architectures for computers (von Neumann [1993]), which use instruction sets to tell CPUs how to manipulate data in order to solve problems. While these methods have proven empirically to be able to solve a vast variety of problems relatively efficiently, there is a question of "are current computation methods the fastest way to represent and solve our problem?"

The idea for this paper is that we present a more naturally-motivated way to represent problems, in our case sorting and shortest path, with diodes, and see how we can solve them more quickly than with traditional Turing Machine style computation.

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.

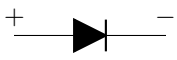


Figure 1: A diode, with the positive and negative terminals marked.

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`, graduate of Carnegie Mellon University, M.S. in ECE

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

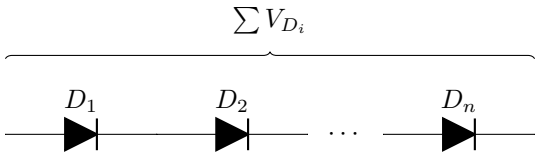


Figure 2: Diode chain

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.

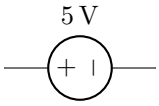


Figure 3: A voltage source of $5\,\mathrm{V}$

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.

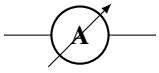


Figure 4: An ammeter

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 (left) and a closed switch (right)

3 Circuit Topologies

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

3.1 Sorting

Definition 5. The sorting problem is when we are given a list of n numbers, and we need to output the numbers in increasing order.

The fastest algorithms for sorting numbers are

- **QuickSort:** $O(n \log n)$, used in practice because of quickness and optimization (Hoare [1962]).
- **Radix Sort:** $O(b(n+k))$, despite linear efficiency, requires fixed size keys and a way of breaking down keys like it does for numbers. Therefore not used in practice as much (Davis [1992]).

We will present a circuit that allows us to sort n numbers in $O(n)$ time and has $O(n)$ space complexity. We will see later that we have some of the limitations of radix sort, namely that we are constraining ourselves to numbers, but we have a more general solution because the algorithm doesn't use any other assumptions.

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:

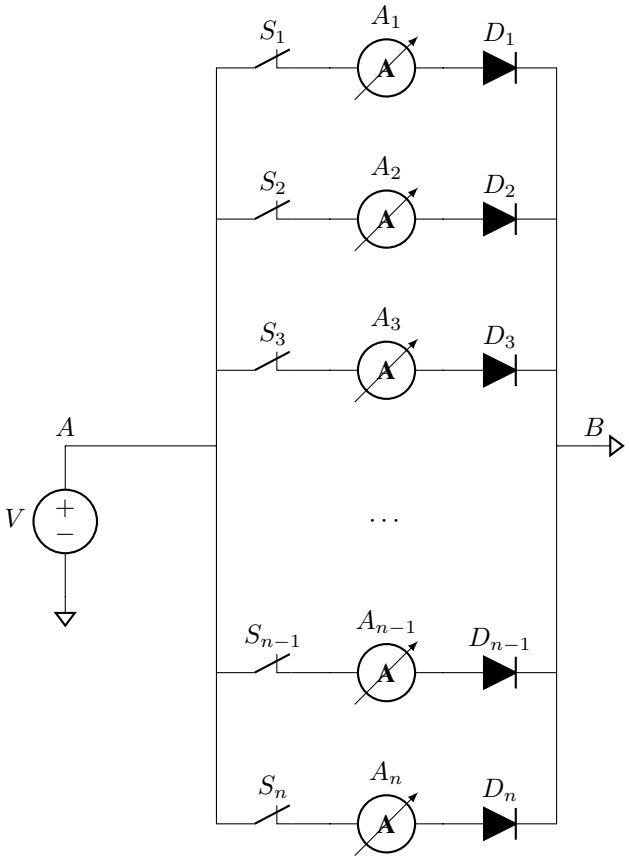


Figure 6: Circuit for sorting numbers

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 starting at A, with a switch S_i closed, ammeter A_i, and diode D_i
    and ending at B

  // Sort
  queue Q = []
  set V = V_max
  while ∃S_i that is on:
    for i such that I(A_i) > 0:
      open 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 regarding a property we will call “superparallel comparison”,

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 by induction.

- 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.

This algorithm will terminate, since every iteration takes at least one diode from the set of diodes that have their switch S_i on, and pushes that voltage into the queue. □

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 in Graphs

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

Definition 6. The shortest path problem on a weighted, directed graph is given a graph G, we want to find the shortest path from V_i to V_j, for any two vertices $V_i, V_j \in G$.

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 (Floyd [1962]).

We will present a circuit and corresponding algorithm that runs in $O(|V|^2)$ time and uses $O(|V|+|E|)$ space complexity.

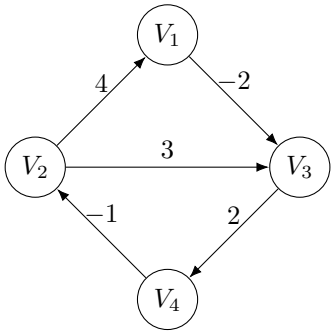
3.2.1 Circuit

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

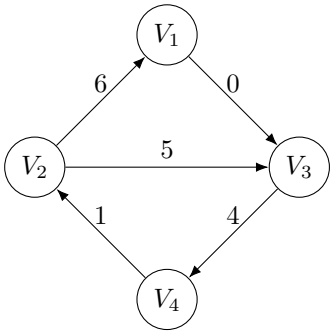
```
BUILD_DIODE_GRAPH(G = (V, E)):
  Normalize G so that there are no negative weights, producing G' = (V, E')

  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
```

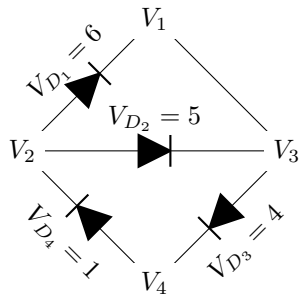
For example, with the following graph, we can construct the corresponding diode graph as follows:



(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
- $|E|$ wires from the ammeter to the device running the algorithm
- $|V|$ 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.

I only tested small graphs and weights, meaning my $V_{\text{max}} \leq 10\,\text{V}$ in all cases.

I tested correctness of my algorithm on a sorting problem with $n = 5$, and on a graph with 6 nodes[3].

For further research, I should compare the efficiency of my circuits to library functions. Since I am using breadboard circuits and hobby-shop components, it may be hard to produce a fair comparison to the very-optimized layout of a CPU, but nonetheless, it would be interesting to see the asymptotic behavior of the algorithm to see if the above analysis are actually correct. Furthermore, it's quite cumbersome to have to create new circuit topologies for every variation of a problem, so there is definitely improvement on building some sort of circuit layout that maybe more generalizable and modular to support a variety of sorting or graph problems easily.

5 Discussion

The biggest question regarding the practicality of the circuits in this paper are about the diodes and physical properties.

Question 1. Do real diodes behave close enough to ideal for the results in this paper to hold?

Although in Section 4 I wasn't able to provide a lot of evidence, especially for larger scales, the exponential $I - V$ curve for diodes likely makes the properties in this paper good enough to hold in real life. The only adjustment is that because a diode is no longer exactly "off" anymore before reaching V_D, meaning the diode will probably still allow

[3] I would really want to post videos of these circuits, but I no longer have them, and would like some funding before attempting to build larger scale circuits

some current through, instead of measuring $I(A_i) < 0$ to determine if a certain diode is on or not, we may instead have to check $I(A_i) < \epsilon$, but this is not a big deal.

Question 2. How do we get diodes of arbitrary threshold voltage?

This is probably the biggest issue with the circuit design in this paper. While I wasn't able to find much about very large turn-on voltage diodes, as it's unlikely anybody would ever need a $100\,000\,\mathrm{V}$ diode, we are likely to believe that a diode with V_D will likely require $O(V_\mathrm{D})$ resources to construct. However, if this is not the case, then the circuits in this paper have some profound implications.

Question 3. What are some of the physical limitations of these circuits, with regard to electrical signal and the speed of light?

Physical limitations include how the electrons travel through wires and in the circuit. 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. However, even with this consideration, we can see that our circuits are all no bigger than the input size, so our time and space complexities are unaffected, even with this electron speed limitation.

5.1 Other Problems to Consider

While I have presented 2 natural problems to solve with these diodes, we beg the question of what other problems can we solve with this superparallel diode comparison idea?

There are some properties of the superparallel comparison that give some interesting properties. For example, if we had a complete binary tree with weights on their edges, and we wanted to know the shortest path from the root to any leaf node, it would normally be a very difficult problem to do with traditional computation, but with superparallel comparison, we can just apply V_max to the root and attach all the leaf nodes together, and be able to find that shortest path among 2^n very fast. Obviously there are the costs of translating the input into the diode way of representing the problem, but the fast comparison seems to at least lessen the burden of comparing many numerical quantities to each other one by one.

Maybe we only really are able to gain time efficiencies on the order of $O(\log n), O(n)$, but depending on the application, that could be a very significant improvement.

References

Alan Mathison Turing et al. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58 (345-363):5, 1936.

J. von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993. doi:10.1109/85.238389.

C. A. R. Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 01 1962. ISSN 0010-4620. doi:10.1093/comjnl/5.1.10. URL `https://doi.org/10.1093/comjnl/5.1.10`.

I. J. Davis. A Fast Radix Sort. *The Computer Journal*, 35(6):636–642, 12 1992. ISSN 0010-4620. doi:10.1093/comjnl/35.6.636. URL `https://doi.org/10.1093/comjnl/35.6.636`.

Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, jun 1962. ISSN 0001-0782. doi:10.1145/367766.368168. URL `https://doi.org/10.1145/367766.368168`.