Resource Management in Internet of Things by Implementing Kruskal’s and Floyd’s Algorithms

Azar Alizada

Faculty of Engineering and Applied Sciences,
Khazar University, Azerbaijan
Email: agalizada00@gmail.com

Abstract

The idea of the Internet of Things (IoT) has received a lot of interest recently because it has the potential to transform our physical world into a meaningful information-filled digital cyber environment. The IoT devices are more computationally powerful and smaller in resources in terms of size, number, memory, bandwidth and energy consumption. This thesis will examine resource management in IoT. The Internet of Things (IoT) is a network that is made up of real-world items, including sensors, software, and other technologies integrated into them with the primary intention of connecting and exchanging data with other equipment and systems. Resource management is a critical component of IoT, as it ensures that the various devices and sensors are able to communicate and work together efficiently (I. Rašan, 2021). In this thesis, we will customize and implement two algorithms for resource management in IoT: Kruskal's algorithm and Floyd's algorithm. Kruskal's algorithm is an algorithm that looks for the most optimal solution, which is utilized to find the minimum spanning tree of a graph. The algorithm works by considering all the possible routes of the graph and selecting the one with the lowest weight. The minimum spanning tree is added to these possible routes, after which, the procedure is repeated until all the edges have been considered. Floyd's algorithm is an all-pairs shortest-path algorithm that is used to detect the best path in terms of length between all pairs of nodes in a certain graph. The algorithm works by considering all the nodes of the graph, and afterwards calculating the shortest route between every pair of nodes. We will compare and contrast the two algorithms, and evaluate their effectiveness by combining both to optimize resource management in IoT. As a result, Floyd’s and Kruskal’s algorithms are customized to detect and eliminate loops within devices to improve resource management in IoT devices. By using Floyd algorithm before Kruskal’s implementation it can be determined if there is a cycle in the IoT resources. Floyd algorithm-discovered cycles can be eliminated by applying the Kruskal algorithm. This method allows fewer cables, tracks, and other

components to be used in IoT devices. It can improve the efficiency and management of IoT resources by reducing costs, time, effort, and other factors.

Keywords: Sensor, embedded systems, network, cloud computing, technologies

Introduction

One of the most promising technologies of the present decade is the Internet of Things, defined as using intelligent, self-configuring things connected to a network and exchanging data by perceiving, responding to events, and interacting with the environment. Furthermore, if they have sensing technology and onboard computing components, unmanned mobile devices like drones have integrated into the overall "objects" participating in the IoT. These sensing and computational capabilities enhance network embedded intelligence in the recently introduced mobile Internet of Things and allow for the realization of complex tasks in a highly distributed manner, balancing load across infrastructure and significantly increasing communications' energy efficiency (Agrawal, Shashank and Dario Vieira, 2019). Mobile IoT networks are the consequence of their primary ability to move through space and time, which gives IoT monitoring requirements a new level of freedom.

In the article, first of all, the definition of the Internet of Things, its usage areas, its architecture and wireless communication technologies used in this field are explained. Then, with the existing studies, privacy and security analysis in the Internet of Things is done. Attacks against objects; physical, network and software cyber-attacks. Communication protocols used in this area are examined in terms of privacy and security and possible security gaps are determined. It describes the basic precautions that should be taken against security vulnerabilities. In addition, a new method is proposed to ensure the management of the nodes produced by the manufacturers from a single point, as well as to increase the security in the addition of new nodes to the existing networks, which is one of the most critical parts of information security on the Internet of Things, and to contribute to closing the security gaps that may occur. The proposed method is called cloud. For this method, a cloud application is first developed, and the nodes are managed from a single point. However, when a new node will join an existing network, the network coordinator will connect to the cloud system, will obtain the security information of the node to join and will verify whether the node is malicious or not. If the authentication is successful, the node joins the network. In case of failure to verify, the node cannot join the network.

To summarize all points talked about, IoT related products and gadgets produce an enormous amount of data that needs to be stored and easily accessed at any given moment which is made easier and cheaper via the use of the internet to connect to a cloud storage device. These cloud storage devices store the necessary data while communicating with the gadgets. However, cloud services should also provide a robust security system to combat against any cyber attacks which may lead to a handful of unwanted scenarios such as data being stolen and kept as ransom, or the attacker shutting down the service temporarily due to which the gadgets would not function. The cloud storage services must also provide a nimble way of storing and accessing data for efficiency purposes, the faster this process becomes the more accurate a device or a gadget may operate giving it the possibility to also analyze the data at a much more quick and accurate way. In addition, having a central network where all of the gadgets work is going to make it more straightforward to interact and manage the devices and all of its functionality (Evans, Martin, 2017). Moreover it will give the ability to interconnect different sensors and gadgets with each other to make certain actions easier.

Due to the enormous benefits that IoT provides on a daily basis, companies have sunk a massive amount of wealth on further developing the gadgets present in the field. This means that resource management has not received as much attention and is not up to date with the technology currently available. This leads to inefficiency which can cost a lot of money, not to mention the fact that better management could mean that servers for cloud based services could be made more compact while operating at the same if not better efficiency which would further lead into better space management for tech companies. Providing a solution for the above mentioned points would not only improve the quality of the services provided, but also solve most, if not all of the problems currently present in the resource management area of IoT.

Material and Method

Resource management in IoT

In order to enhance resource management in IoT devices, Floyd's and Kruskal's methods can be modified to locate and dispose of the loops within devices. It is possible to identify whether there is a cycle in the IoT resources by applying the Floyd method prior to the deployment of Kruskal’s method. The Kruskal method can be used to get rid of cycles found by the Floyd algorithm. Less wires, tracks, and other parts can be used in IoT devices thanks to this technique. By lowering expenses, spending less time and effort, and other variables, it can increase the management and efficiency of IoT resources. Resources and devices are dispersed throughout the IoT environment at an increasing rate. A vast variety of protocols, data formats, and physical sensing resources need to be managed in order to exploit the full potential of the deployed devices (Weber, Rolf H. and Romana Weber, 2020). This raises the issue of how effectively the resources of the devices can be controlled and provisioned.

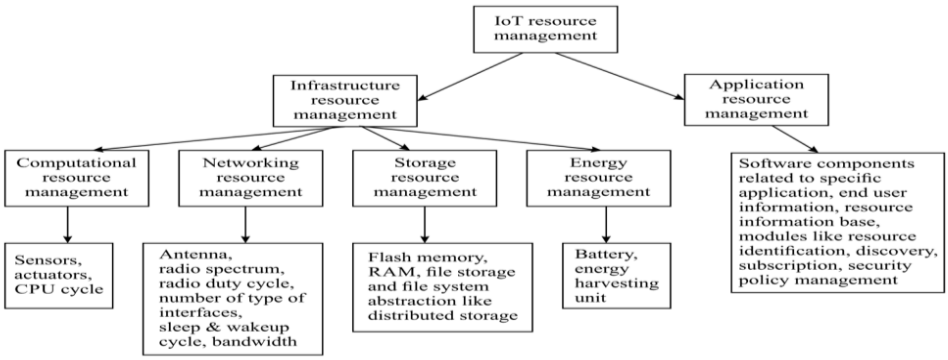

Figure 1. Types of IoT resources.

Source: Bhajantri, Lokesh B, and Gangadharaiah S. "A Comprehensive Survey on Resource Management in Internet of Things." Journal of Telecommunications and Information Technology, vol. 4, no. 2020, 2020, pp. 27–43., doi:10.26636/jtit.2020.145220.

In order to address issues with IoT resource management, this study takes advantage of an algorithmic method that makes use of the Floyd and Kruskal algorithms. Kruskal's method is used to determine the shortest route between any two locations in a linked weighted graph. In contrast, Floyd's cycle detection method is used to spot pointless cycles. In order to address issues with IoT resource management, this study offers an algorithmic method that makes use of both of the above mentioned algorithms. These two methods have been combined in order to discover and eliminate loops inside certain devices, which will improve resource management in IoT devices. The quantity of cables, traces, and other components required in IoT devices can be decreased using this technique. Applying this strategy will result in savings in time, money, and effort, which could improve resource management in the IoT.

Each of these approaches has its own advantages and disadvantages. In this thesis, we will survey the state of the art in resource management in IoT, with a focus on the challenges that need to be addressed in order to achieve efficient resource utilization. We will discuss in the next chapter some of the perspectives that have been proposed to display these obstacles by implementing Kruskal’s and Floyd’s algorithms.

Implemented Algorithms

Floyd’s Cycle Detection Algorithm

In computer science, a data graph is one that is used to train a model (cycle). If a pointer going on a road at two different speeds can display the same value here, then there is one similar to Floyd's Cycle Detection Algorithm here. It is the algorithm used to determine whether cycles exist in computer science graphs used to model data. This method, also known as Floyd's Cycle Detection Algorithm, identifies whether a loop is present if two speed pointers moving along a path might both show the same value. The Floyd's cycle finding method was developed by computer scientist Robert W. Floyd and introduced in 1967.

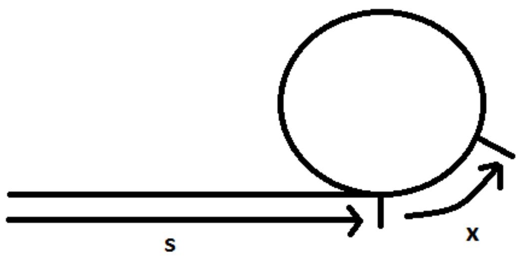

Figure 2. Floyd's Algorithm

Source: https://github.com/epomp447/Floyd-Warshall-Algorithm-Java-

The cycle's beginning is symbolized by the letter S.

S+x stands for, where s is the offset from the cycle's beginning where the two pointers meet.

The quicker pointer may have traveled an additional distance marked as s+x from the offset x location with the cycle after the slow pointer had to travel s+x. It returned to the same state and location with the offset x cycle thanks to the additional s+x distance (https://github.com/epomp447/Floyd-Warshall-Algorithm-Java-).

This simply specifies that the pointer will be from the cycle's start-position, offset 0, if the distance s+x moved from the specified offset accompanies the pointer to the offset x, and if x moved less from that specific position, or so s.

Floyd’s cycle detection algorithm pseudo code:

```
function CycleExists(IoTDevices) {

if(IoTDevices is empty)
return 'No cycle found.';

IoTDevicesSlowPointer = IoTDevices;
IoTDevicesFastPointer = IoTDevices;
while(IoTDevicesFastPointer != null && IoTDevicesFastPointer.nextIoTDevice !=
null)
{
IoTDevicesSlowPointer = IoTDevicesSlowPointer.nextIoTDevice;
IoTDevicesFastPointer = IoTDevicesFastPointer.nextIoTDevice. nextIoTDevice;

if(IoTDevicesSlowPointer == IoTDevicesFastPointer)
return 'Cycle detected.';
}
return 'No cycle found in IoT resources.';
}
```

Kruskal’s Algorithm

Kruskal's algorithm is an algorithm that looks for the most optimal solution, which is utilized to find the minimum spanning tree of a graph. The algorithm works by considering all the possible routes of the graph and selecting the one with the lowest weight. The minimum spanning tree is added to these possible routes, after which, the procedure is repeated until all the edges have been considered.

The detailed steps of the Kruskal method are shown below. The subset being grown is represented by the blue edges. The algorithm considers each edge in the order of its weight. Numbers next to the edges denote their weight. The edge that is being taken into consideration is indicated by an arrow at each level of the algorithm. If the edge connects two distinct forest trees,, the two trees are combined and the edge is added to the forest. If the edge creates a cycle, it is disregarded. As can be seen from the image, in step four (d), the edge with four weights is disregarded because it produces a cycle.

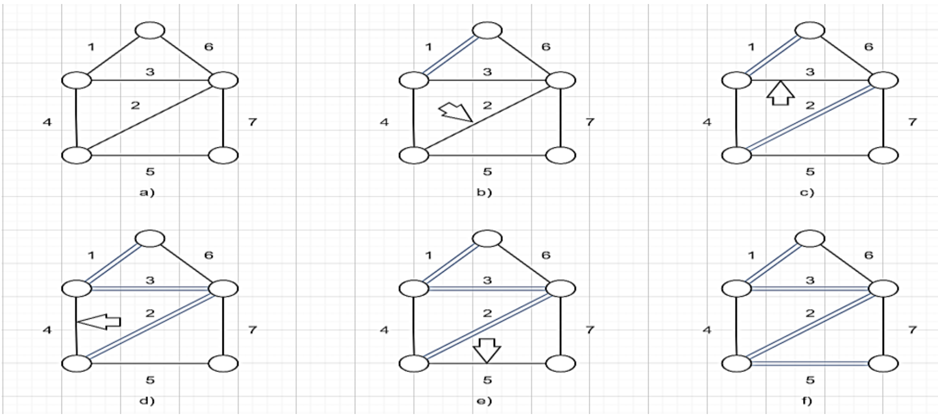

Figure 3. Occurrence of a cycle.

Finally, Floyd algorithm-discovered cycles can be eliminated by applying the Kruskal algorithm. Thus, the vertices' best and most efficient connections are discovered. This can help reduce the number of wires, traces, and other components inside IoT devices, which may improve Cost, time, and effort resource management in the Internet of Things.

Kruskal algorithm pseudo code:

```
   marked = true
Integer color[1000]
Input: Number of IoT Devices(total)
  FOR total number of IoT Resources(N):
    insert(locationX)
    insert(locationY)
    insert(Weight)
    graph[locationX][locationY] = marked
    weightGraph[locationX][locationY] = Weight
Output:
  FOR each node:
    DFS(currentIoTDevice)
```

```
    MSA(currentIoTDevice) //Regraphs the inputted graph so that there are no
cycles
    MSA(Integer currentIoTDevice){
  FOR N number of IoT Resources:
    IF graph[currentIoTDevice][Nth IoT Resource] is equal to true & used[Nth
IoT Resource] is not equal to true:
      MSAgraph[currentIoTDevice][Nth IoT Resource] = 1;
      fWeight(currentIoTDevice][Nth IoT Resource] =
weightGraph[currentIoTDevice][Nth IoT Resource];
      MSA(i);
  }
  DFS(Integer currentIoTDevice){
    FOR N number of IoT Resources,:
      IF graph[currentIoTDevice][Nth IoT Resource] is an edge connecting the
currentIoTDevice to the IoT Resource i:
        IF color[Nth IoT Resource] is not left completely:
          IF color[Nth IoT Resource] has been left but not completely:
            cycle is true
          DFS(Nth IoT Resource);
    color[currentIoTDevice] is completely left
}
print graph[N][N]
print MSAgraph[N][N]
print fWeight[N][N]
```

Customization of Kruskal's and Floyd's Algorithms

This study describes an algorithmic solution that employs the Floyd and Kruskal algorithms to address IoT resource management concerns. To enhance resource management in IoT devices, these two techniques have been integrated and used to find and remove loops inside specific devices. The only difference between the Kruskal's algorithm implementations used in this experiment is the method used to sort the graph's edges. As a result, we examine the time required by each sorting method and use it to determine which factor dominates the algorithm's total running time. By using Floyd algorithm before Kruskal's implementation it can be determined if there is a cycle in the given graph.

- Two pointers: fastPointer and slowPointer are initialized for looping through the nodes.

- The slowPointer advances one position, but the fastPointer advances two.
- The fastPointer moves twice as quickly as slowPointer. It travels twice as far as the slowPointer does.
- A loop is present if both pointers cross at some point. If the fastPointer reaches the end position then no loop exists.
- If a loop is found then by using Kruskal's algorithm the edges that create the loop can be removed.

Floyd's and Kruskal's methods can be modified to locate and dispose of the loops within devices. It is possible to identify whether there is a cycle in the IoT resources by applying the Floyd method prior to the deployment of Kruskal's method. The Kruskal method can be used to get rid of cycles found by the Floyd algorithm. In order to address issues with IoT resource management, this study takes advantage of an algorithmic method that makes use of the Floyd and Kruskal algorithms. Kruskal's method is used to determine the shortest route between any two locations in a linked weighted graph. In contrast, Floyd's cycle detection method is used to spot pointless cycles. . Less wires, tracks, and other parts can be used in IoT devices thanks to this technique. By lowering expenses, spending less time and effort, and other variables, it can increase the management and efficiency of IoT resources. Resources and devices are dispersed throughout the IoT environment at an increasing rate (Atzori, Luigi, Antonio Iera, and Giacomo Morabito, 2020). In order to address issues with IoT resource management, this study offers an algorithmic method that makes use of both of the above mentioned algorithms. These two methods have been combined in order to discover and eliminate loops inside certain devices, which will improve resource management in IoT devices. The quantity of cables, traces, and other components required in IoT devices can be decreased using this technique. Applying this strategy will result in savings in time, money, and effort, which could improve resource management in the IoT.

Flowchart for customization of Kruskal's and Floyd's Algorithms is given below:

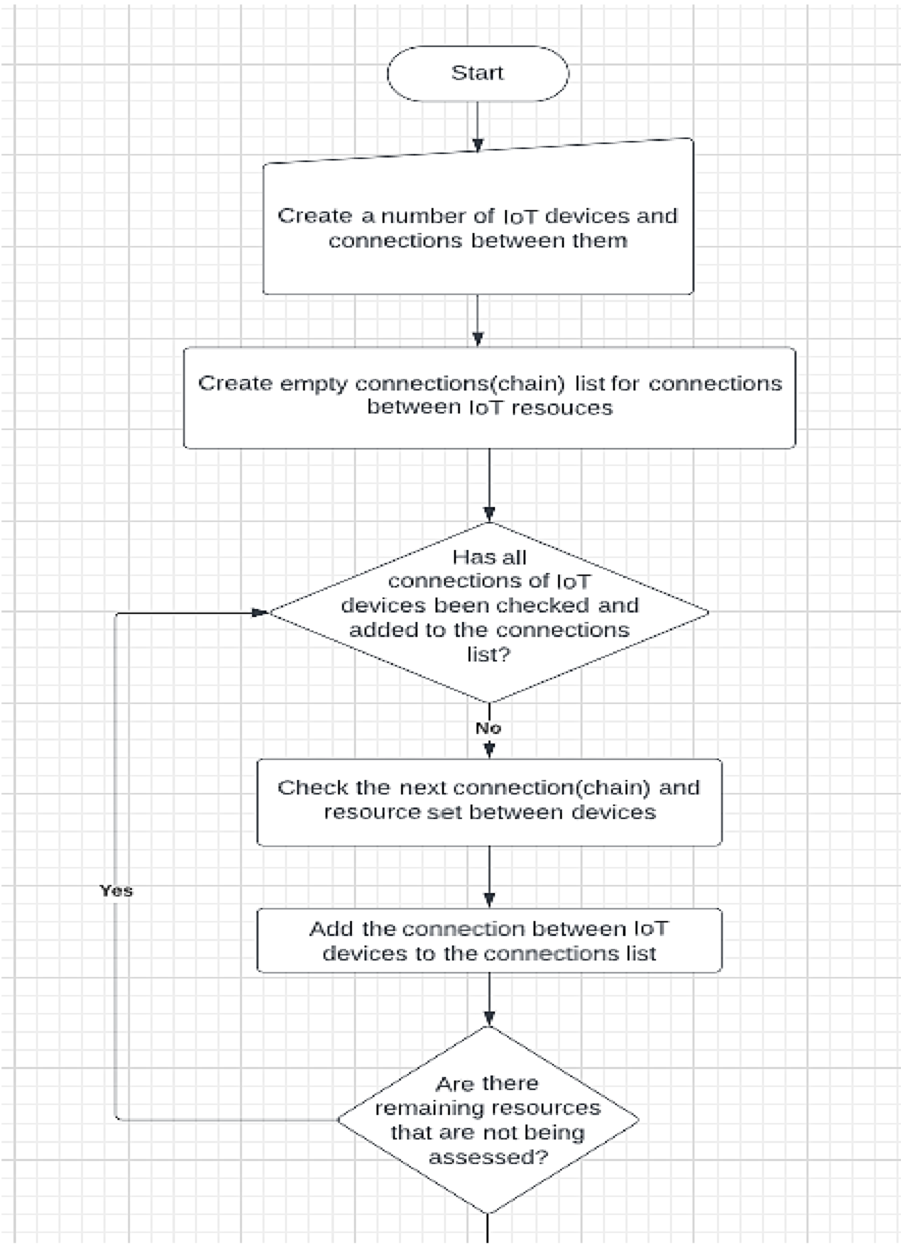

Diagram continues with the following page

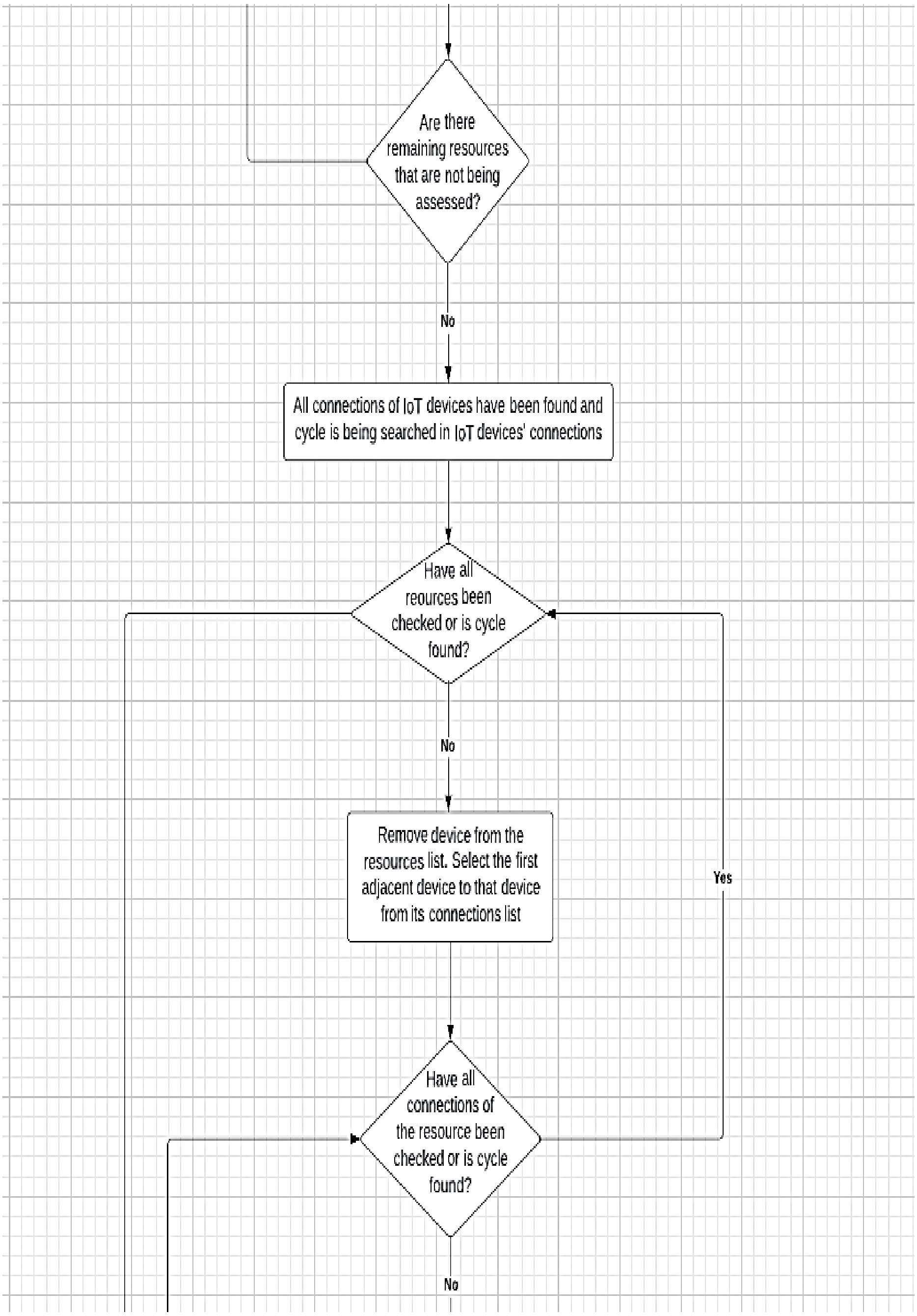

Figure 4. Flowchart for cycle detection and deletion by implementing Kruskal algorithm after finding cycle by Floyd algorithm

Simulation of the Customized Algorithms with Scenario

This image shows an example of an input made by a user. There are 32 nodes in this example, the first digit inputted on each line shows the node from which the connection between the nodes is made(edge(s)), and the second value on each line is the node that the edge is directed towards/connected to. The third numeral represents the weights of each edge from one vertice to another. Hence combinations of these edges between these vertices create a directed graph, which in turn can be used to derive different sorts of information or outcomes.

```
32
1 17 2
2 18 3
3 19 4
4 20 5
5 21 6
6 22 7
7 23 8
8 24 9
9 25 1
10 26 2
11 27 3
12 28 4
13 29 5
14 30 6
15 31 7
16 32 8
17 16 9
18 15 1
19 14 2
20 13 3
21 12 4
22 11 5
23 10 6
24 9 7
25 8 8
26 7 9
27 6 1
28 5 2
29 4 3
30 3 4
31 2 5
32 1 6
4 6 7
5 7 8
6 8 9
22 23 1
19 15 2
31 5 3
10 3 4
27 16 5
```

Figure 5. Inputted Graph of Example as a list

The 4 images shown on the next pages are the inputs entered by the user in a weighted graph form, multidimensional array form, the output of the input without cycles and comparison table of the task execution. The first multidimensional graph is the graph that had been input in picture 5. The second multidimensional figure is the output of the input graph without the edges that cause cycles.

The image below is the input entered by the user converted into a directed graph which is much easier for the user to understand. In this specific example, there are 8 cycles and 32 nodes.

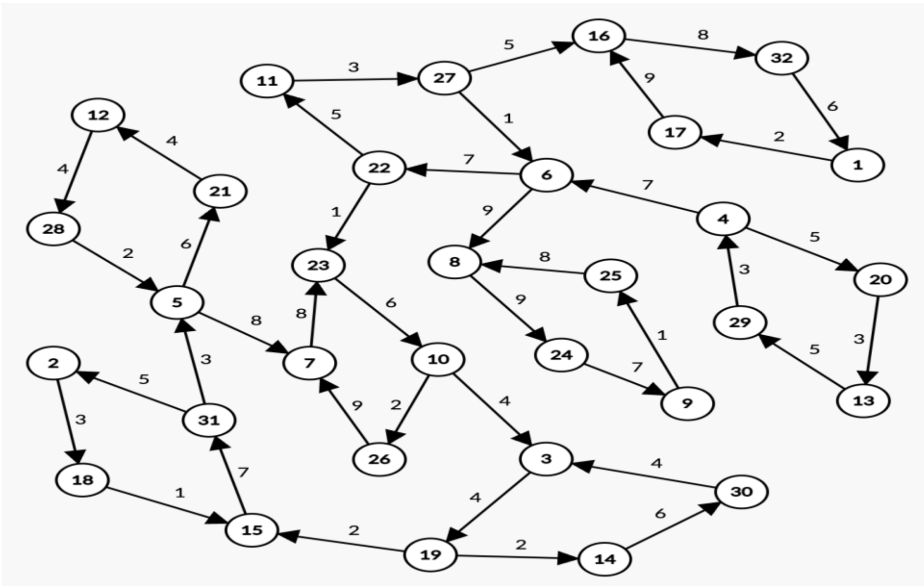

Figure 6. Directed graph

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 5 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 4 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 3 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 9 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 8 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 2 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 3 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 4 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 5 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 6 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 7 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8
2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 4 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 5 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 6 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 7 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 8 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 9 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 8 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 9 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 2 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 3 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 4 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 5 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 7. Multidimensional Array Form of Input List

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 2 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 3 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 4 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 5 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 6 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 7 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8
2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 4 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 5 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 6 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 7 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 8 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 9 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 8. Outputted Graph of the Given Example without the Cycles

Table 1. statistical parameters like duration, etc.

Duration	Min Durati...	Avg Durati...	Std Dev D...	Max Durati...
133.62 ms	5.58 µs	177.22 µs	3.85 ms	105.72 ms
133.62 ms	5.58 µs	177.22 µs	3.85 ms	105.72 ms
5.78 ms	7.46 µs	11.19 µs	4.13 µs	65.17 µs
1.76 ms	5.58 µs	29.77 µs	44.40 µs	238.17 µs
5.31 ms	19.67 µs	90.03 µs	112.42 µs	580.42 µs
9.25 ms	40.96 µs	156.83 µs	153.06 µs	788.62 µs
1.70 ms	12.58 µs	53.09 µs	134.86 µs	780.00 µs
1.11 ms	42.88 µs	58.59 µs	15.26 µs	109.50 µs
105.89 ms	28.96 µs	21.18 ms	47.26 ms	105.72 ms
407.62 µs	407.62 µs	407.62 µs	–	407.62 µs
694.96 µs	694.96 µs	694.96 µs	–	694.96 µs
787.88 µs	787.88 µs	787.88 µs	–	787.88 µs
927.04 µs	927.04 µs	927.04 µs	–	927.04 µs

The first one from the left is the duration of the task, and the second from the left shows the minimum time it should take to solve a task like this. Avg duration shows how much time it would usually take to solve a task like this, Std Dev is the library used, and Max duration means the maximum amount of time needed to run the program. In all test cases, the results came back in less than 150 ms

Results and Discussion

In result, as it seems in the above example all user inputs have been tested and calculation time of all inputs has been shown in the tables for given scenario. For better understanding the inputs and test results have been converted into graphs. By using Kruskal's and Floyd's algorithms the cycles inside graphs have been found and depicted in the images, output is obtained without cycles which were given in input. As a result, utilizing this method can help decrease the number repetitions/cycles in terms of wires, traces, and other components inside IoT devices, which may boost IoT resource management in terms of cost, time, and effort.

In the IoT, there are progressively more resources and devices being distributed into the environment. In order to take advantage of the deployed devices, a wide range of protocols, data formats, and physical sensing resources must be controlled (Wilkinson, Glenn, 2019). This study introduced an algorithmic solution that employs the Floyd and Kruskal algorithms to address IoT resource management concerns. The shortest path in a connected weighted graph between any two points is found using Kruskal's approach. On the other side, Floyd's cycle detection technique is employed to identify unnecessary cycles. This study describes an algorithmic solution that employs the Floyd and Kruskal algorithms to address IoT resource management concerns. To enhance resource management in IoT devices, these two techniques have been integrated and used to find and remove loops inside specific devices. This method allows for a reduction in the number of wires, traces, and other components used in IoT devices.

Graph theory, a branch of discrete mathematics, introduces the concept of Kruskal's algorithm (Lokesh B. Bhajantri and Gangadharaiah, 2022). This technique finds the shortest path between any two points in a linked weighted graph. This method converts a given graph into a forest by considering each node as a separate tree. Floyd's cycle detection algorithm, often known as the Hare-Tortoise algorithm, is a pointer algorithm that traverses the sequence using just two pointers and two distinct speeds. The loop in a linked list can be found using this approach. It employs two pointers, one of which moves twice as quickly as the other. In this research these two algorithms have been combined and employed to detect and eliminate loops inside given devices in order to improve resource management in IoT devices. By using this technique, the quantity of wires, traces, and other components inside IoT devices can be reduced.

References

Abou-Zahra, Shadi, Judy Brewer, & Michael Cooper (2019). Web Standards to Enable an Accessible and Inclusive Internet of Things (IoT). The 14th Web for All Conference on The Future of Accessible Work - W4A '17 Proceedings, pages 1-4ACM Press, New York, New York, USA.

Agrawal, Shashank & Dario Vieira (2019). A Survey on Internet of Things, Abakós 1.(2).DOI: 10.5752/P.2316-9451.2013v1n2p7

Al-Fuqaha, Ala, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, & Moussa Ayyash (2019). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials, vol. 17(4), pp. 2347–2376.

Al-Sarawi, Shadi, Mohammed Anbar, Kamal Alieyan, & Mahmood Alzubaidi (2017). Internet of Things (IoT) Communication Protocols: Review, 8th International Conference on Information Technology, 2018. (ICIT). IEEE. pp. 685–90

Atzori, Luigi, Antonio Iera, & Giacomo Morabito (2020) in 2017 8th International Conference on Information Technology, Internet of Things (IoT) Communication Protocols: Review (ICIT). IEEE. Pp. 685–90

Bravo, Jose, Ramon Hervas, Salvador W. Nava, Gabriel Chavira, & Carlos Sanchez. n.d. Towards Natural Interaction by Enabling Technologies: A Near Field Communication Approach. in Constructing Ambient Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg. Pp. 338–51

Centenaro, Marco, Lorenzo Vangelista, Andrea Zanella, & Michele Zorzi (2020). Long-Range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios. IEEE Wireless Communications 23(5):60–67.

Doppler, Klaus, Mika Rinne, Carl Wijting, Cassio Ribeiro, & Klaus Hugl (2019). Device-to-Device Communication as an Underlay to LTE-Advanced Networks. IEEE Communications Magazine 47(12):42–49.

Duan, Kang-Kang & Shuang-Yin Cao (2020). Emerging RFID Technology in Structural Engineering – A Review. Structures 28:2404–14.

Evans, Martin (2017) Arduino in Action.

Geng Wu, S. Talwar, K. Johnsson, N. Himayat, & K. D. Johnson (2017). M2M: From Mobile to Embedded Internet. IEEE Communications Magazine 49(4):36–43.

González, Gustavo Ramírez, Mario Muñoz Organero, & Carlos Delgado Kloos (2008). Early Infrastructure of an Internet of Things in Spaces for Learning. Pp. 381–83

Gubbi, Jayavardhana, Rajkumar Buyya, Slaven Marusic, & Marimuthu Palaniswami (2018). Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. Future Generation Computer Systems 29(7):1645–60.

He, Debiao, Neeraj Kumar, & Jong-Hyouk Lee (2019). Secure Pseudonym-Based Near Field Communication Protocol for the Consumer Internet of Things. IEEE Transactions on Consumer Electronics 61(1):56–62.

Hodges, Steve, Stuart Taylor, Nicolas Villar, James Scott, Dominik Bial, & Patrick Tobias Fischer (2019). Prototyping Connected Devices for the Internet of Things. Computer 46(2):26–34.

Lokesh B. Bhajantri & Gangadharaiah (2022). A Comprehensive Survey on Resource Management in the Internet of Things. Journal of telecommunications and information technology 4/2022

Rašan, I. (2021). Digitalization in the role of humanization or dehumanization of modern society, 44th International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 1460–1465.

Villa-Henriksen, Andrés, Gareth T. C. Edwards, Liisa A. Pesonen, Ole Green, & Claus Aage Grøn Sørensen (2020). Implementation, Applications, Challenges, and Potential of the Internet of Things in Arable Farming. in Biosystems Engineering.191:60–84

Weber, Rolf H. & Romana Weber (2020). Internet of Things. Berlin, Heidelberg: Springer Berlin Heidelberg.

Welbourne, Evan, Leilani Battle, Garret Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, & Gaetano Borriello (2009). RFID Ecosystem Experience: Building the Internet of Things Using RFID. IEEE Internet Computing, 13(3).48–55

Whitmore, Andrew, Anurag Agarwal, & Li Da Xu (2019). The Internet of Things—A Survey of Topics and Trends. Information Systems Frontiers 17(2):261–74

Wilkinson, Glenn (2019). Digital Terrestrial Tracking: The Future of Surveillance. DefCon22

https://github.com/epomp447/Floyd-Warshall-Algorithm-Java-