The Traveling Salesman Problem

Theory, Approaches, & Applications

What is the Traveling Salesman Problem?

- A famous and relevant issue involving optimization, subsets, graph theory.

- The main idea of the traveling salesman problem is this:

1. A person is given a starting point on a graph, which represents an area of interest.

2. They are tasked with traveling to each marked location within the graph in the shortest distance, or fastest time possible.

3. As programmers, we typically like to solve these problems with algorithms, which may be applicable depending on the situation (The simplicity of the graph).

4. However, there is no truly efficient algorithm to correctly solve this problem every time (other than comparing every possible route, which could be in the thousands for larger problems!)

So then, How does one usually solve the "Traveling Salesman Problem?"

That's Easy... Just Use Heuristics!

Heuristics are any approach to problem solving that uses practical methods, but is not guaranteed to be rational. So you also have to THINK ABOUT IT.

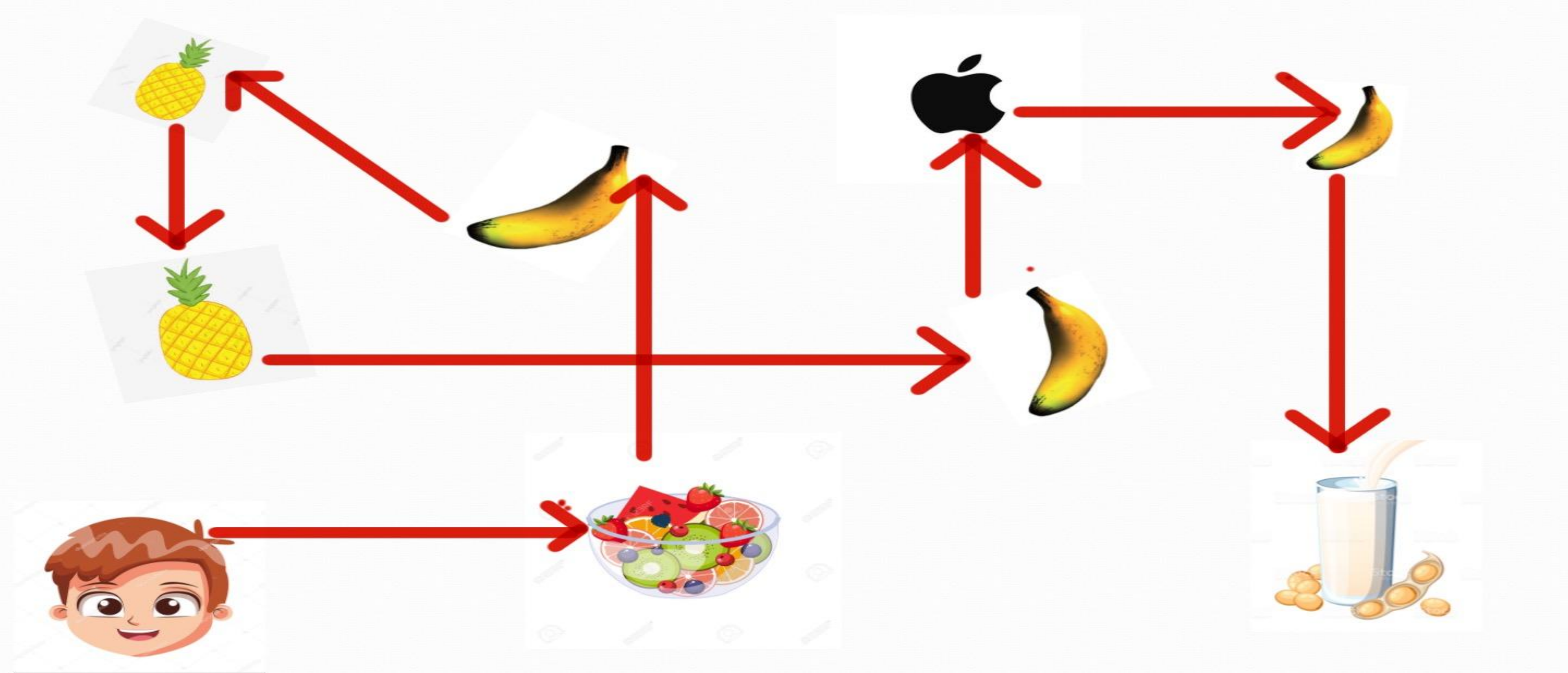
The Three Main Heuristics Are...

- Intuition (Guess and Check)
- Nearest neighbor
- Space Filling Curve (SFC)

Let's Take The Same Problem and Solve it Four Different Ways:

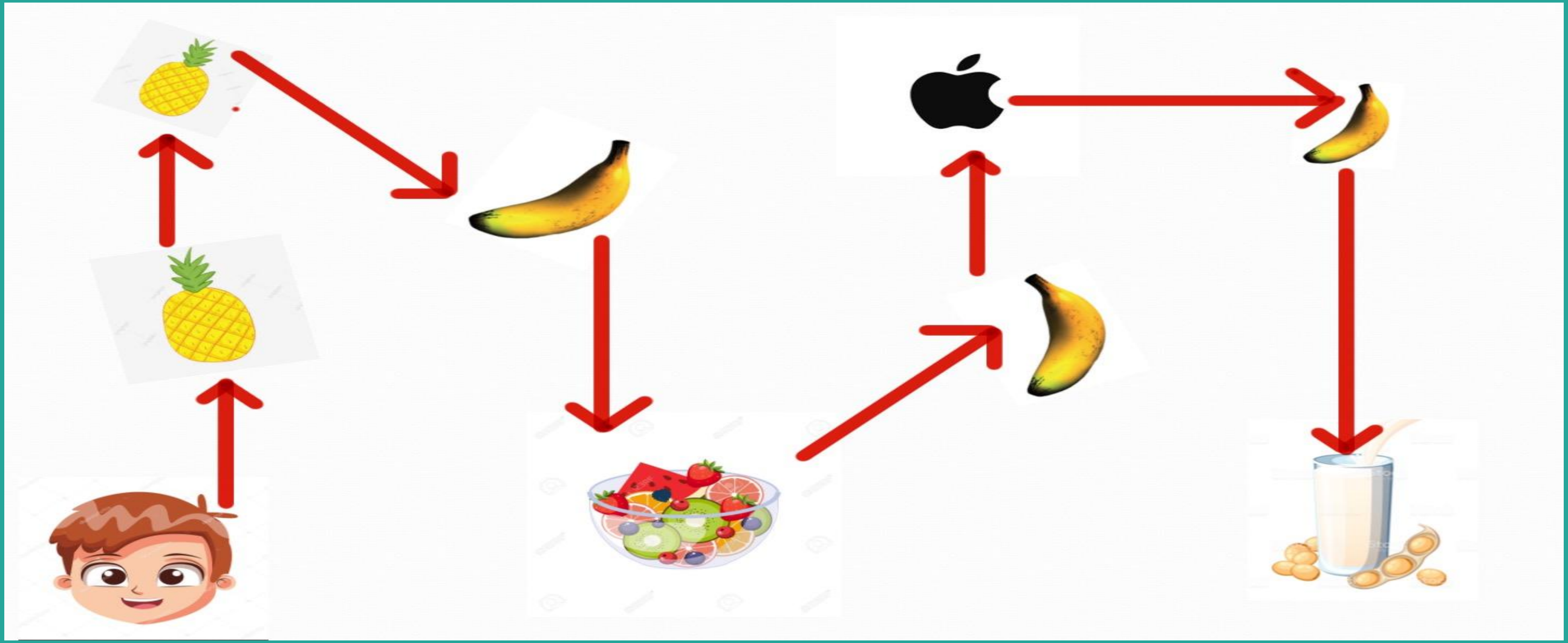
- Someone is at home and is oversleeping, since he is under nationwide quarantine due to COVID-19.
- They miraculously wakes up immediately before a Zoom meeting and realize that they have about three and a half minutes to do eat breakfast.
- Unfortunately, the ingredients have been scattered throughout his kitchen and now they must get to each one as quickly as possible so they are not late to class.
- Note:It does not matter which task in the room they gets too first, so long as he gets its done as efficiently as possible.

Heuristic #1: Intuition: To do this, just connect all the locations with a straight line.



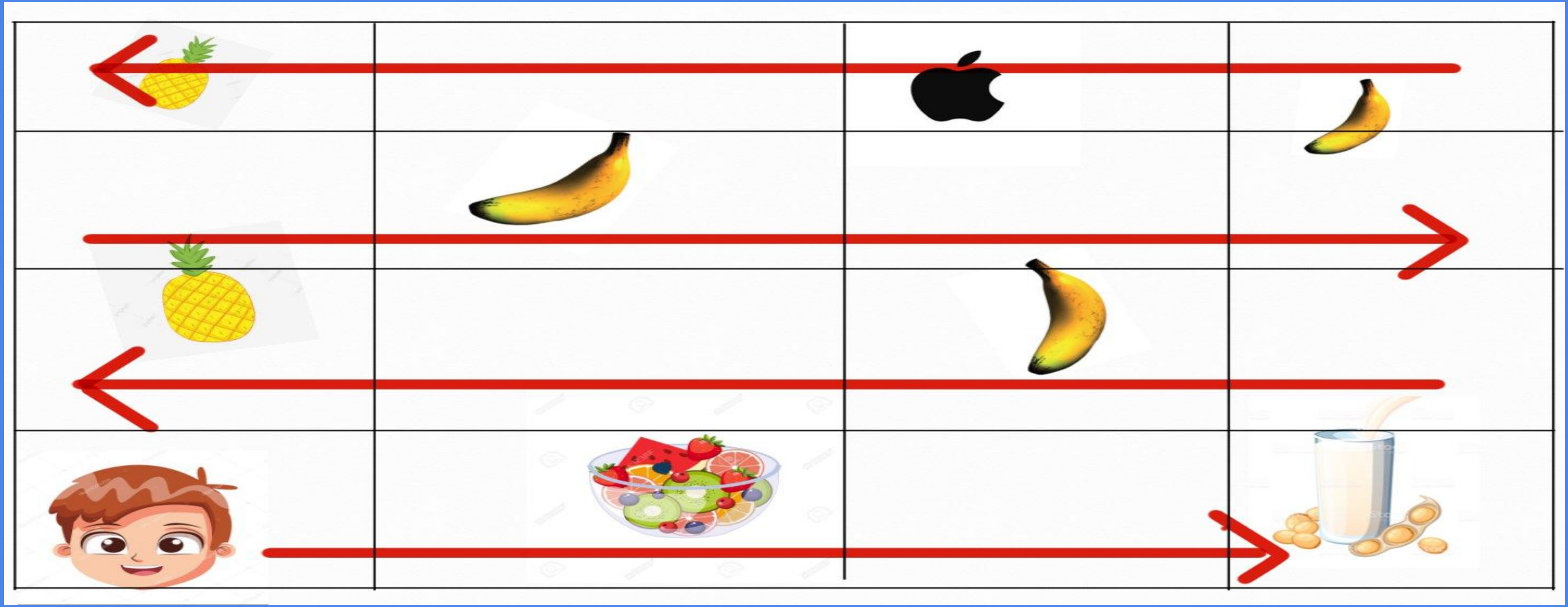
Heuristic #2: Nearest Neighbor

- Start from any point, and then choose the neighbor that is closest until every destination is reached.
- The efficiency of this approach largely depends on where you start.



Heuristic #3: Space Filling Curve (SFC)

Steps: - Put all of your points into a graph and then travel to every section of the graph as efficiently as possible. Then highlight the paths where your curves touched the desired locations.



TSP In Computer Science:

- Computers will not implement the same techniques as people will.
- Instead, we can implement an algorithm that utilizes recursion.
- What is the Method then?
- First, some basic definitions on graph theory.

Definitions & Objectives:

1. Directed Graph: Made up of a set of vertices connected by edges, where the edges have a direction associated with them.
2. Vertices: Places in the graph one must travel to.
3. Objective 1: We must travel through all of the vertices at least once, and then return to our original destination.
4. The cost for travel must be as little as possible (extra variables).

Directed Graph: Example

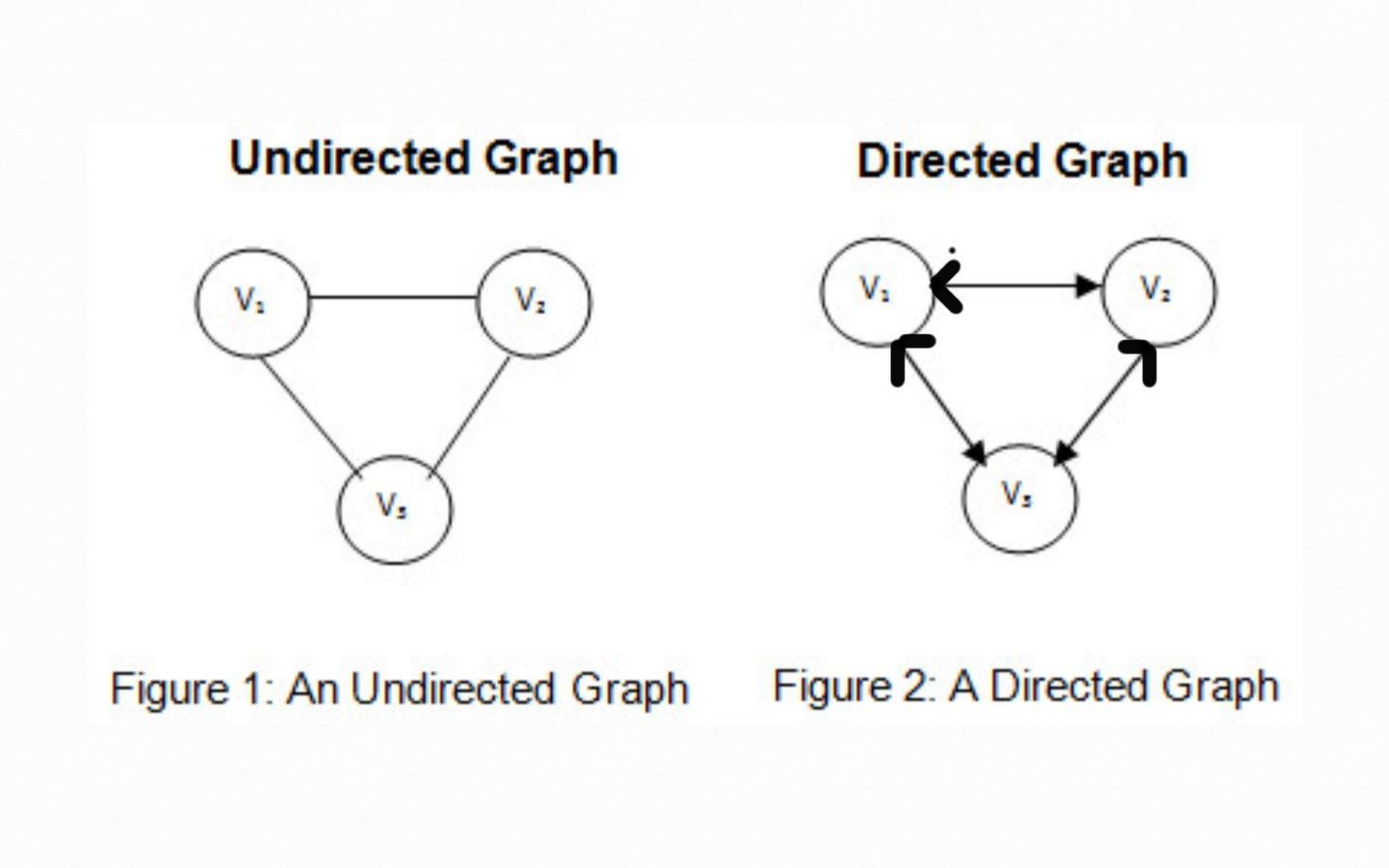


Figure 1: An Undirected Graph

Figure 2: A Directed Graph

Formula and Implementation of TSP:

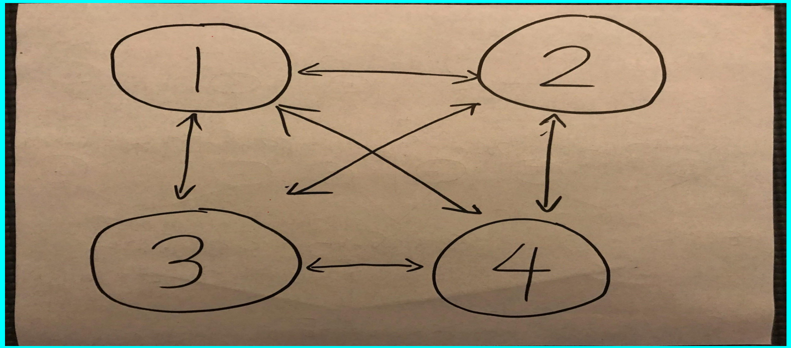
The Algorithm is Generally This:

g(i,s) = minimum{ C (iK) + g(K, S- {K})}

Example:

g(1, {2,3,4}) = min {2,3,4} { C (iK) + g(K, {2,3,4}- {K})}

Note: Once s (remaining set is exhausted), you would make the function input your starting point into parameter ‘s’ (through recursion!)



Notes:

I = The vertex which you are starting from.

s = remaining vertices in your set to travel to.

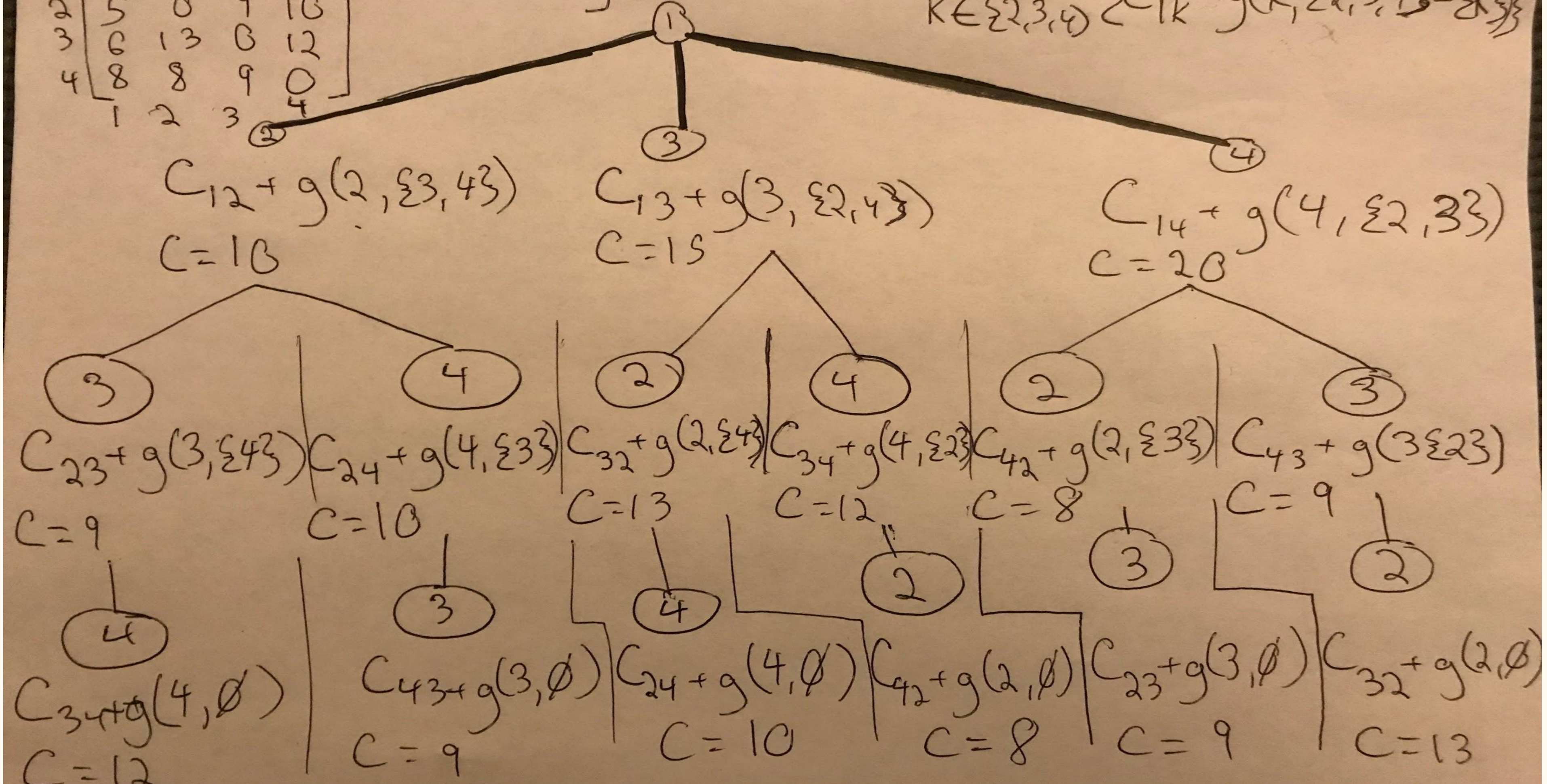
c (iK) = Represents the first element in the set you can travel to from your starting point . In this case, its 2,3, or 4. The C represents cost of travel, which is represented by a set.

G is our function, and it is being used recursively.When you see s - {K}, this means that K is being removed from the set s as a possible place to travel to.

Minimum represents a separate function of comparing each possible route to take and takes the lowest. Its best to visualize this from a tool called a recursive tree!

$$\begin{array}{c|cccc} 1 & 0 & 10 & 15 & 20 \\ 2 & 5 & 0 & 9 & 10 \\ 3 & 6 & 13 & 0 & 12 \\ 4 & 8 & 8 & 9 & 0 \\ \hline & 1 & 2 & 3 & 4 \end{array}$$

$$g(1\{2,3,4\}) = \min_{K\in\{2,3,4\}} \{C_{1k} + g(k, \{2,3,4\} - \{k\})\}$$



Final Thoughts...

- The Recursive algorithm in my opinion isn't usually what's inefficient. Rather, it's the size of the data set we are working with that will be too much for a computer to handle. This happens with the increase of destinations (which could be in the thousands.)
- The Traveling Salesman Problem Is used in a wide array of places, including the realm of computer networking, genome sequencing (Bioinformatics), and gps.
- The Run Time for the Recursive Definition is at most O (n^2 * 2^n), which is not feasible for computers once the number of vertices, or destinations, reaches a certain amount, due to its exponentiality.