Flutter In Action: How to master Algorithms - Graph API - MPT & SPT

Posted by Khushboo Uchat May 19, 2021

After having read through the title if you started reading this article it means **you are ready to dive in the ocean of Algorithms** (If you are sitting near any wall then please do not bang your head !!). If you have not read my previous article then you can find it here. I will strongly recommend to go through it.

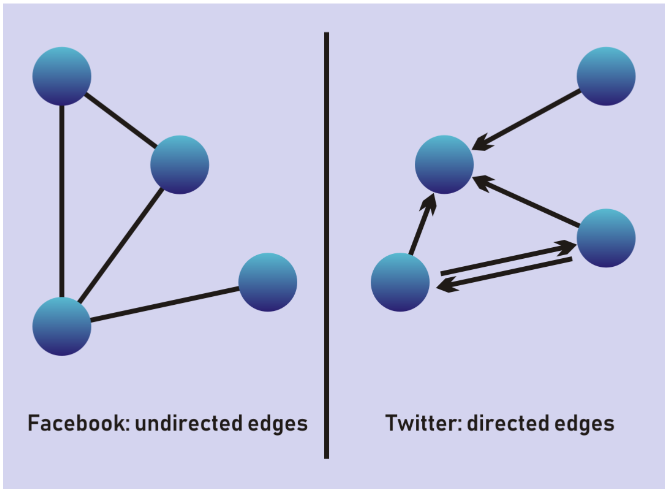
How to master Algorithms - Basics

I will assume that **you are very well verse with** Algorithm basics and different kind of data structures like tree, queue and heaps and **graphs**. Also if any point of time you feel it is too much, I have attached gif images and here is a **challenge for you**. **Just by looking at gif, can you guess the steps of Algorithms or what is it doing ?**

Graph and Graph API :

. It is widely used structure in modern era of technology as well as virtual reality. You will see them everywhere. Graph is nothing but connected dots by lines in very simple language. In technological terms, we will name them vertices connected by edges. Graph is of two types.

Undirected Graphs : vertices connected by edges but edges does not have any direction
Directed Graphs or Diagraphs : vertices connected by edges with edges having direction from one vertex to other. We can see the example of social media in below image which will explain very well how the graphs is useful everywhere.



MST (Minimum Spanning Tree) :

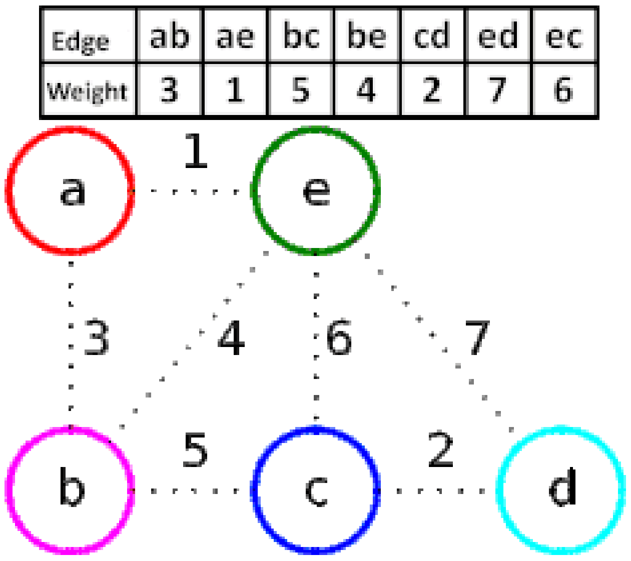
It is powerful algorithm used to solve a variety of huge problems that are important in practical applications nowadays. What is it ultimately ? In this algorithm, The edges have a positive number associated with it called weightage. **Spanning tree is subgraph which has no cycles or self joins in it.** Now we want to find the spanning tree which has minimum weightage. Weightage can be calculated by summing up all the weightage of the edges. But **what are we going to do with finding out MST? We will remove that part of the graph and replace it with the edge of equal weightage.** It has got practical use case in thousand of scenarios ranging from face detection to cancer research.

There are lots of algorithms developed around calculation of MST but discussing all of them is not possible. I will still discuss two of them which is really important to compute MST.

- **Kruskal's Algorithm :**

1. We will sort all the edges by the weight and keep on adding the edges in the tree until it forms a cycle.
2. When the edge is forming the cycle, ignore it.
3. We will keep on doing the same until all edges are considered. At the end the tree will be MST.

It is one example of Greedy Algorithm (Our greed is increasing at each step so it is called Greedy Algorithm). ***This algorithm is good for understanding but in practice it might not be easy to visualize with billions of edges.***



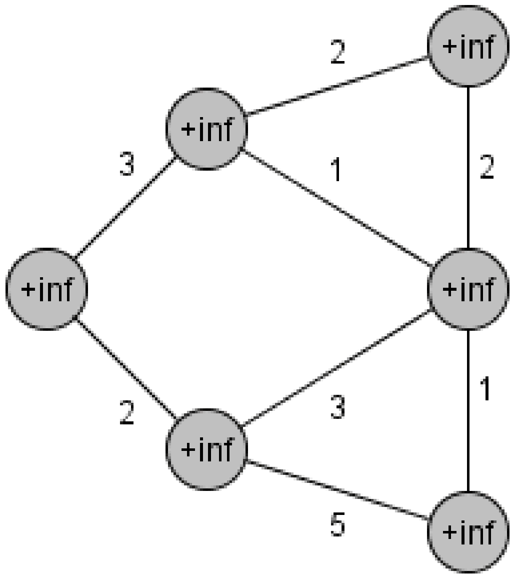
Edge	ab	ae	bc	be	cd	ed	ec
Weight	3	1	5	4	2	7	6

- **Prim's Algorithm :**

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.

2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.
3. Repeat step 2 (until all vertices are in the tree).

In practice, modified version of this algorithm is used and mostly Binary Heap is used for this algorithm in practice to find MST.



Shortest Path :

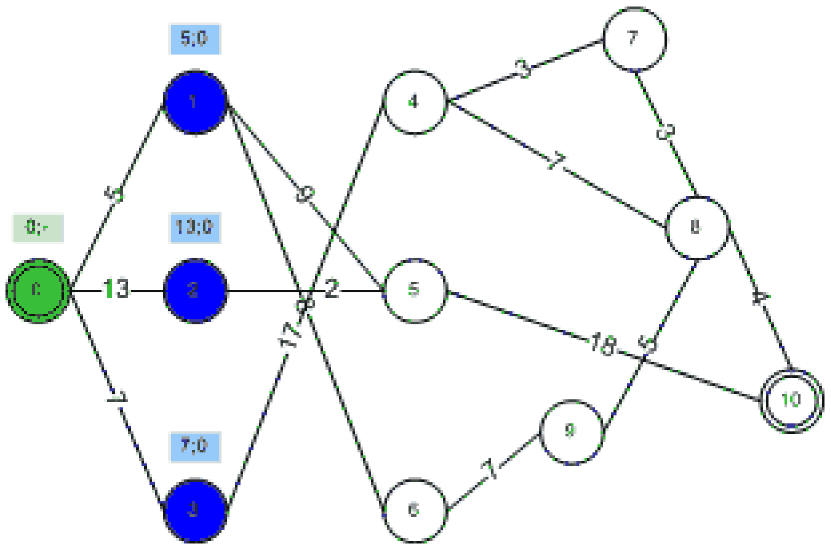
One of the most common problem we encounter in our life as well as coding. So it will be easy to understand for all of us as everyone wants to find shortcut to save the efforts (it can be time or money or both in some cases). In the same way, to find the shortest path in graph there are algorithms developed which comes rescue when it comes to graph. (For life you have to write your own algorithms to find shortcut... too much philosophy .. spare me for that please). Here we will restrict ourselves to weighted graph API.

- **Dijkstra's Algorithm**

Here given any vertex as source, we will find the shortest path for every other vertex and form the tree known as shortest path tree. We will assume that every edge has positive number as weightage.

1. 1. Take one vertex and mark all other vertex as unvisited. Assign infinity as shortest distance for all the other vertices
   2. Calculate the distance for every vertex from the source by traversing on the edges by weightage. This process is called relaxation of the edges.
   3. Compare the value you got as distance with the assigned value for the destination vertex and replace the value with shortest one for the destination vertex
   4. When we are done with all unvisited vertices then mark the source as visited. It will not be considered again. Now you can take one of the destination vertex as source.

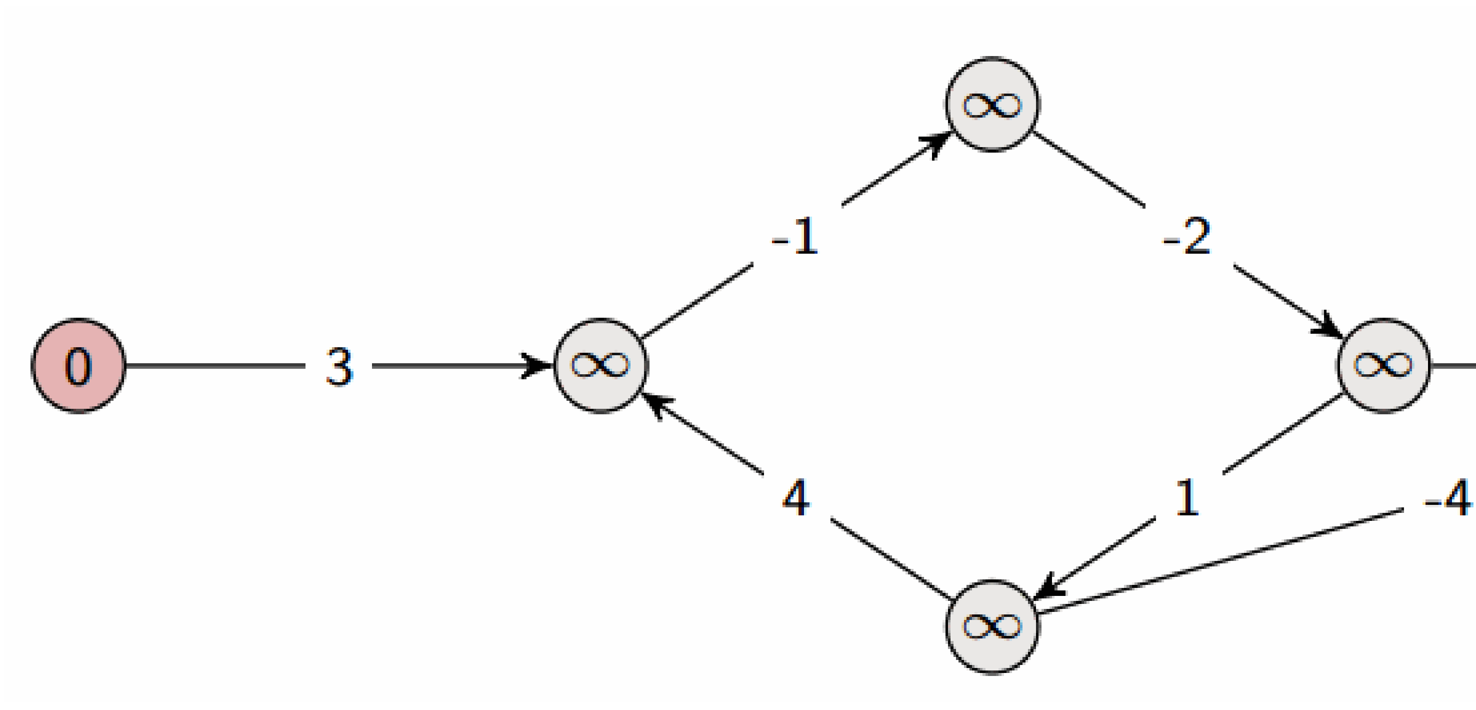
5. Repeat until all vertices are marked as visited or destination node has smallest value compared to all other vertices.



I know it sounds a bit confusing but once you start the process it is just calculate and replace. This algorithm is very useful and lay the foundation on which SPT (shortest path tree) can be calculated.

- **Bellman-Ford Algorithm**

The process is similar to Dijkstra's algorithm but only advantage we got is it repeats the process v-1 times where v is number of vertices. Each phase scans through all edges of the graph, and the algorithm tries to produce relaxation along each edge (a,b) having weight c. Here we will assume that there is no cycle which has all the negative weight. The presence of negative cycle can be detected by separate algorithm and removed. After that this algorithm can be used to detect SPT.



I will stop here as it should be a blog and not a book. If you are bored with graphs then you can count on me. In the next one I will talk about **the simplest thing in our programming world** called **"String"** and after reading the next blog, we will conclude together that it is not simple at all. You can keep guessing what can it be. I will leave this blog like Mistry Series episode. The suspense continues ...
45 Views Tags: #technology

There are no comments on this post