Algorithm Snippets

Atul Dhigra
Rutgers University
Email: atul.dhingra@rutgers.edu

Pritish Sahu
Rutgers University
Email: pritish.sahu@rutgers.edu

Raghav Vashisht
Rutgers University
Email: raghav.vashisht@rutgers.edu

***Abstract*—Data structures and algorithms are an important toolset not only for computer scientists but people from different academic backgrounds to achieve faster and quicker implementation of their work. We therefore, are working on making the algorithms more interactive and intuitive in form of web-based client and instructional videos. These algorithm snippets will be pseudo code driven. We will animate step wise solution which making it more intuitive for a lay person to understand. Algorithm Visualization is very difficult to portray intuitively for advanced algorithms and there has not been much work in this direction. We will design the framework in such a way that it makes understanding an algorithm very easy and intuitive. In our work we have focused our attention on Prim's and Kruskal's implementation of Minimum-Spanning Tree algorithms(MST)**

***Index Terms*—Algorithm Snippets, Minimum-Spanning Tree algorithms(MST), Prim, Kruskal**

I. Project Description

Algorithm animation is the process of abstracting a program's data, operations, and semantics, and then creating a dynamic visualization of those abstractions[1]. It can be used for instructional purposes for better intuition of the working of an algorithm(MST) algorithms. In our project we are developing an interactive web-based algorithm animations. In our project we worked on deterministic algorithm animation and snippets. This framework gives an unique opportunity to learn an advanced algorithm in an easy ways and spending less time. In our work, we focus our project to show the mechanism behind Minimum Spanning Tree which is feasible in 5 weeks. There are very few algorithmic learning resources which use animation to show the nuanced working of algorithms. We attempt to extrapolate along once such framework, TANGO [1]. As compared to TANGO, we concentrate on higher level algorithms. In our work we focus our attention on Prim's and Kruskal's implementation of MST.

The main stumbling block is to visualize the content and make it interactive. We have decided to divide our work into categories, coding the front end, integrating the front and back end and implementing the algorithm.

A. Stage1 - Requirement Gathering Stage

It is widely believed that a high level of interaction increases the effectiveness of the education. In our project we focus our attention on creating a web based framework for interactive learning of algorithms. Presently we are focusing our attention on MST algorithms.

- We will deliver an interactive web-based platform for algorithm snippets. We also wish to store the recordings of the algorithm snippets as instructional videos so as to provide an interactive way for users to provide their own data for algorithm animation. We will also include the running time analysis and space complexity in form of graphical representation.
- This setup can be useful to various kinds of users, including tutors for instructional aid, students for educational purposes. This could develop into a good foray into Algorithms for beginners.
- There are a lot other users that can benefit from online tutorials. Other users may include off-campus students and people in industry who can refer the algorithm snippets for better understanding of algorithmic implementation.
- The scope of the project requires at least 5 weeks to work on data gathering stage, implementation, testing phase. The work has been divided among the team members; Atul will work on technical detailing, preparing documentation and front end. Pritish and Raghav will focus primarily on back-end coding for implementation of the deliverables.

B. Stage 2 -The Design Phase

- We are working with a web based Algorithm learning tool that utilizes Algorithm Snippets for displaying and animating the working of an algorithm. Presently we are focusing our attention on MST algorithm, concentrating on Prim's and Kruskal's implementation of MST.
  In our work we will implement a User Interface in form of a web-page, where the user selects the version of MST algorithm. This animates the algorithm-snippets(pseudo code) and corresponding selection of the minimum edge length on the graph. We will also provide an 'Animator Control' that will enable the user to play/pause the animation, step forward, step backward, slow animation and fast animation.
- Here we present a high level abstraction of the system flow diagram.

- High Level pseudo code for the system is presented as follows.
  1: System is initialized by loading HTML pages that contain links to the two versions of MST algorithms, i.e Prim's and Kruskals.

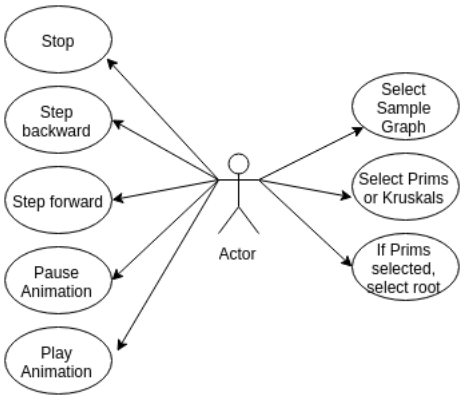

Fig. 1: Flow Diagram

2: If Onclick Prim's or Kruskals, select the respective version of MST and ask user for a starting node, else use a default node.
3: Display step-wise dynamic algorithm snippets with the corresponding selection of the minimum edge selected in that step for the chosen version of MST.
4: If OnClick 'Animation Control' button pressed, perform the corresponding action

- We are working with two versions of MST, Prim's Algorithm and Kruskal's Algorithm. Both these algorithms belong to a set of algorithmic design paradigm called Greedy algorithms. The high level intuition of a MST is to connect points together as cheaply as possible. Prim's version of MST was formulated by Prim in 1957, by Dijkstra in 1959 and earlier in 1930 by Jarnik. Kruskal came up with his version of MST algorithm in 1957. The running time of both these versions of algorithm is $O(|E|log(|V|))$ using Heap data structure for Prim's algorithm and Union Find Data structure for Kruskal's.[2]

Prim's Algorithm
Intuition:Grow tree, one node at a time

Algorithm 1 Prim's Algorithm

1: Initialize X=[s] (s$\in$ V chosen arbitrarily)
2: T=$\emptyset$
3: while X $\neq$ V:
4: let e=(u,v) be cheapest edge of G with u$\in$ X and V$\notin$ X
5: add e to T
6: add V to X

Kruskal's Algorithm
Intuition: Grow trees in parallel and coalesce in the end

- Within the current scope of the project we are working with small set of data that satisfies integrity constraints.

C. Stage 3- The Implementation Stage

- The output from UI is shown below for Kruskal's implementation of MST. In the figure below, each edge is colored blue when a minimum edge weight is detected. The edge is painted yellow when it is selected as a part of MST if and only if it doesn't create a cycle.

Algorithm 2 Kruskal's Algorithm

1: Sort edges in order of increasing cost
2: T=$\emptyset$
3: for i =1 to m
4: if T $\cup$ i has no cycles
5: add it to T
6: return T

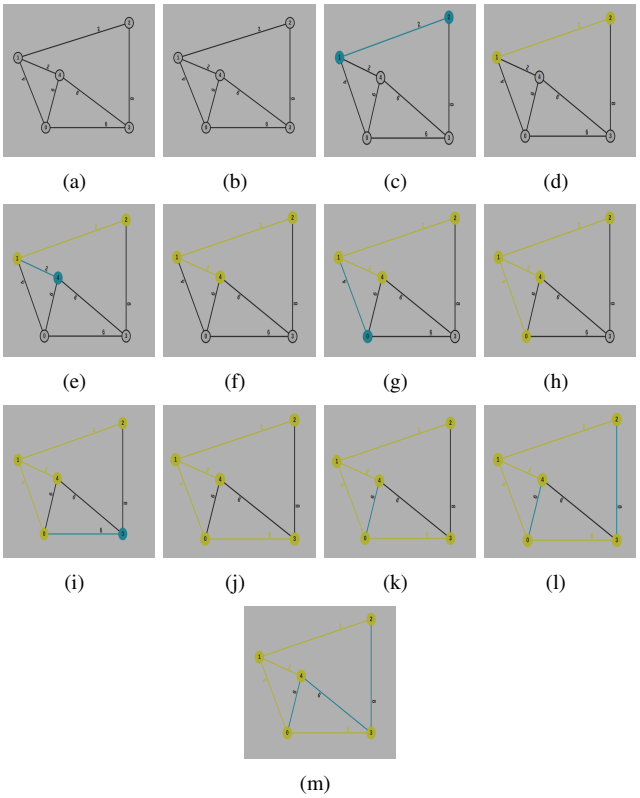

Fig. 2: Kruskal's Implementation

- The output from UI is shown below for Prim's implementation of MST. In the figure below, each node is colored blue when a minimum edge weight is detected. It is checked if a cycle a formed, in that case the node is not selected. In case the edge is feasible for selection, it is painted yellow.

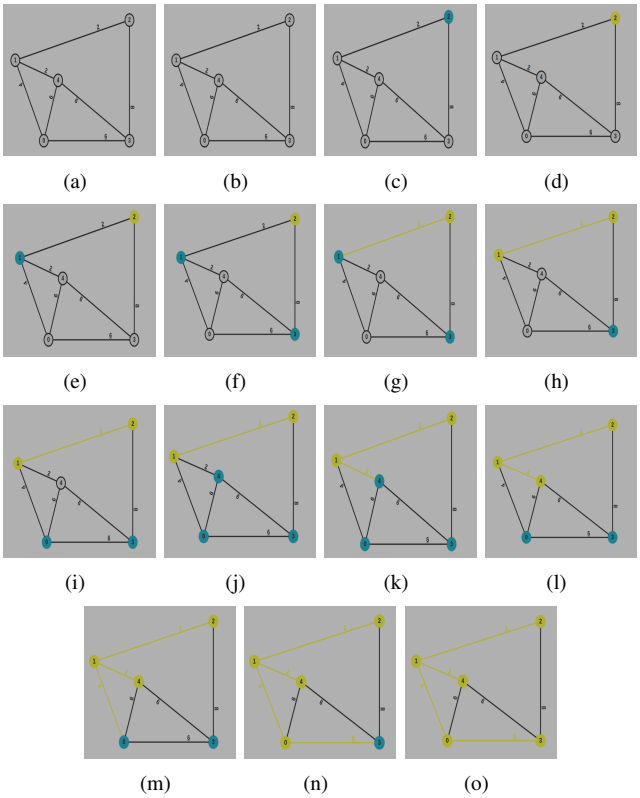

Fig. 3: Prim's Implementation

- Working Code
- Demo and sample findings

D. Stage4- User Interface

We developed a web-based interface for animating algorithm snippets as an learning/instructional aid. We will also be making the videos available for offline use.

- As described in the high level abstraction of the system in Figure 1, the user has will be presented with a web page where he would interact with the system using buttons for choosing graph and the algorithm type. An 'Animator Control' has the functionality of play/pause the animation, step forward, step backward, and stop.
- An error appears if the user selects the Prim's version of MST but, doesn't provide the starting node or the root
- Once the user makes a selection, a graph is animated with the selection of the corresponding minimum weight edge. Simultaneously, algorithm snippets are also animated such that the user is aware which line of the pseudo-code causes which step of the algorithm.
- As the sample graphs are pre-recorded, there are no errors observed in relation to data constraints.
- Figure 5 provides a window into the working structure of the project. There are two drop-down options for selecting different versions of pre-defined graphs, and other drop-down menu is to select the version of MST

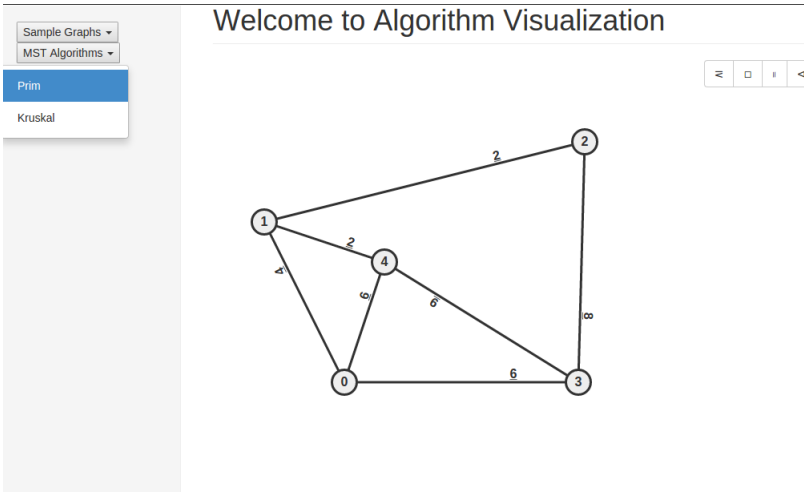

Fig. 4: The User Interface

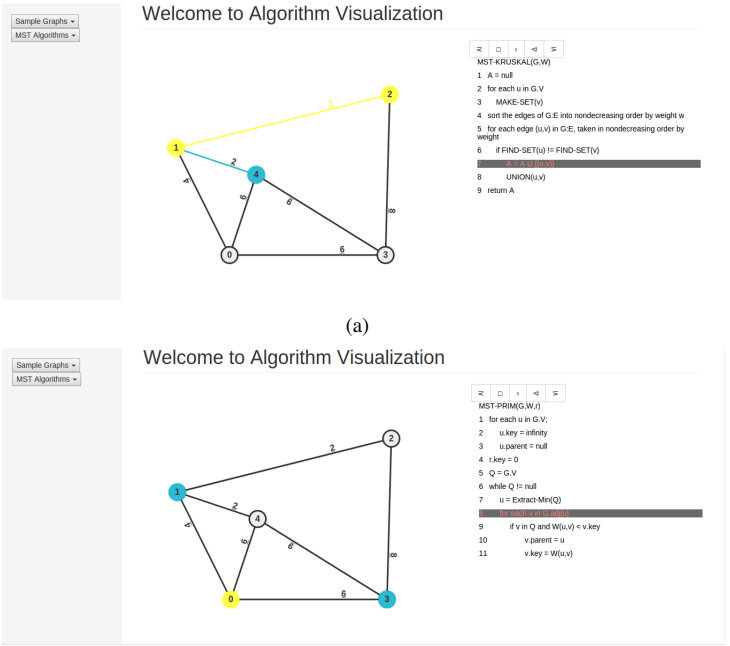

Fig. 5: Navigation through UI

- The two different versions of the interface is to enable a better navigation capability to the user to switch between the different version of MST.
- The user needs to select a graph initially and a version of MST to begin the animation as shown in figure 5.
- Figure 5 shows the two different navigation paths through the UI. Figure 5a shows the UI when the user selects Kruskal's version of MST. Figure 5b shows the visualization of Prim's algorithm for MST.
- An error message pops up, when a Prim's version of MST is selected without providing the starting node or the root for running the algorithm.

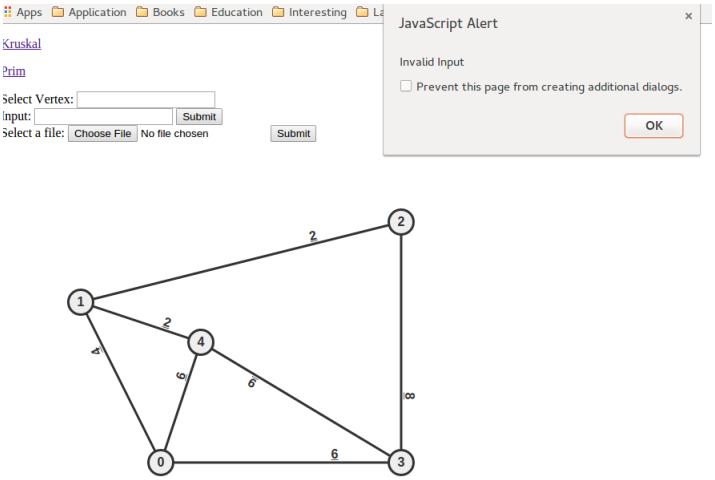

Fig. 6: Error message with Prim's Implementation

- Once the user clicks on a sample graph, and choose a version of MST, either Prim's or Kruskal's the algorithm snippets animate in correspondence to the selection of minimum weight edge on the graph.
- The interface mechanism are shown in figure 4 and figure 5.

REFERENCES

[1]] John T, Stasko, *TANGO : A framework and system for algorithm animation* . Computer, 23(9) :27-39, September 1990

[2] J. Eisner. *"State-of-the-art algorithms for minimum spanning trees: A tutorial discussion"*, University of Pennsylvania, 1997