Algorithms and Data Structures 2
CS 1501

Fall 2022

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming Deadlines
  - Lab 8: next Monday 11/14 @ 11:59 pm
  - Homework 8: next Monday 11/14 @ 11:59 pm

Previous lecture

- Minimum Spanning Tree (MST) problem
  - Prim’s MST algorithm
    - running time analysis of the Best Edges implementation
    - an implementation that uses a heap
  - Kruskal’s MST algorithm

This Lecture

- Weighted Shortest Paths problem
    - Dijkstra’s single-source shortest paths algorithm

Muddiest Points

- **Q: Please review an example of eager prims and kruskals again**
- Sure!

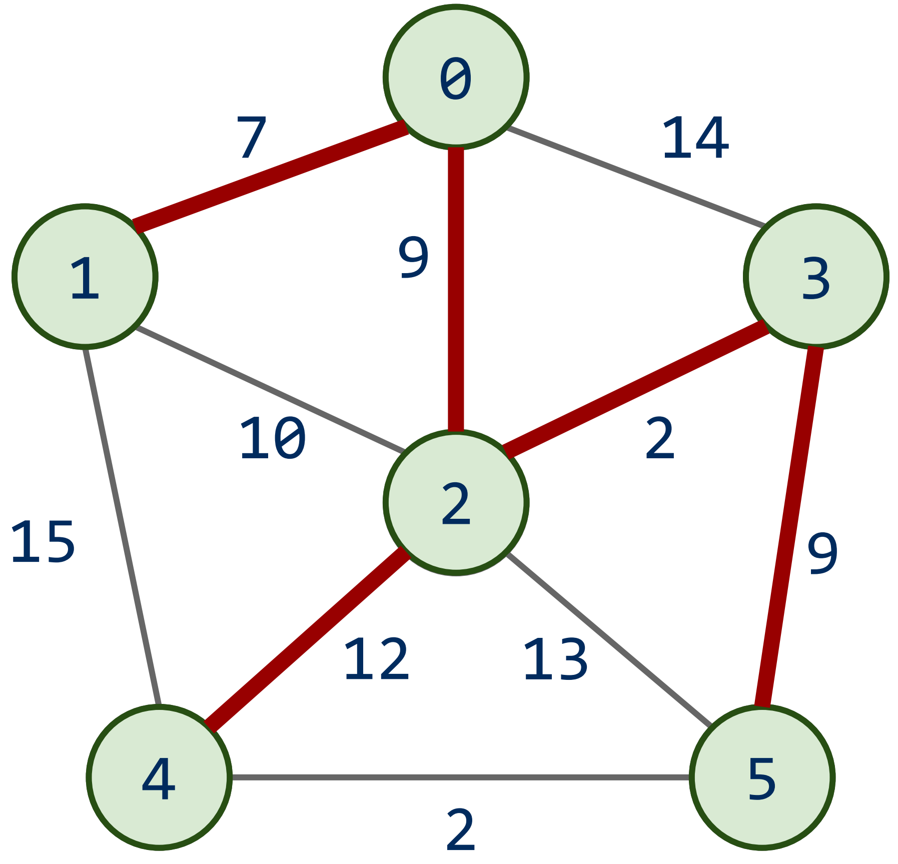
Problem of the Day: Weighted Shortest Paths

- Input:
    - A road network
        - Road segments and intersections
        - Road segments are labeled by travel time
            - From length and maximum speed
            - How do we get max speed?
    - Starting address and destination address
- Output:
    - A shortest path from source to destination

Dijkstra's algorithm

- Set a distance value of Double.POSITIVE_INFINITY for all vertices
- distance[start] = 0
- Set cur = start
- While destination is not visited:
  - For each unvisited neighbor x of cur:
    - Compute tentative distance from start to x through cur
      - distance[cur] + weight of edge between cur and x
    - Update distance[x] if tentative distance < distance[x]
  - Mark cur as visited
  - Let cur be the unvisited vertex with the smallest tentative distance from start

Dijkstra's example



	Distance	Parent
0	0	--
1	7	0
2	9	0
3	11	2
4	21	2
5	20	3

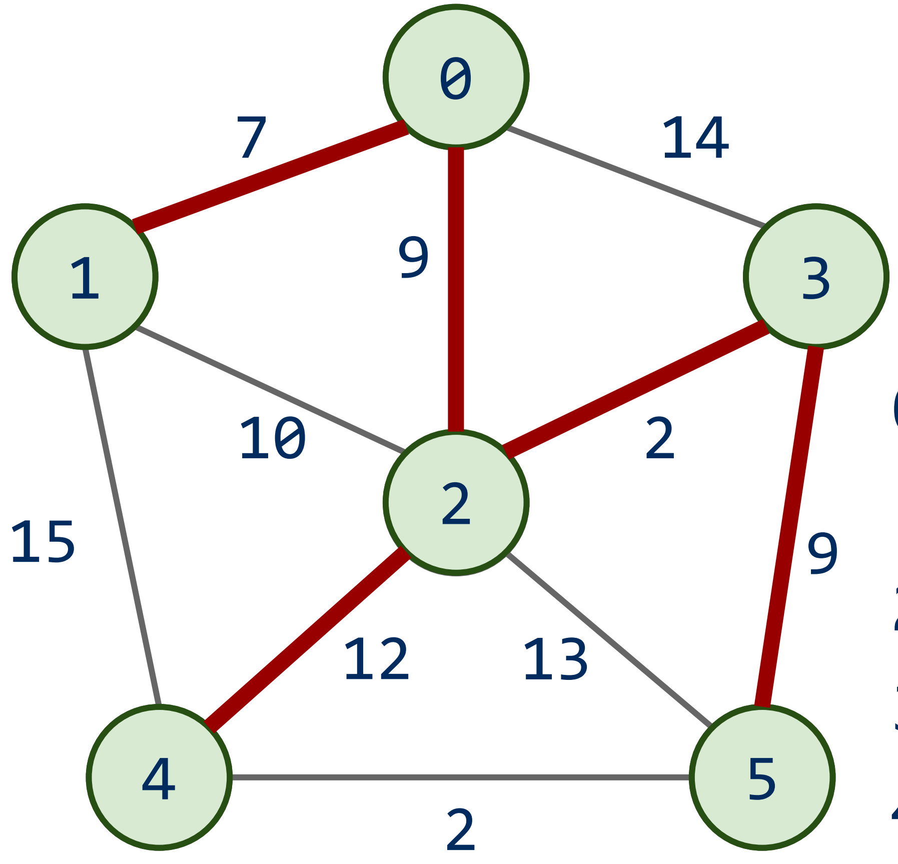
Analysis of Dijkstra’s algorithm

- How to implement?
    - Best path/parent array?
        - Runtime?
    - PQ?
        - Turns out to be very similar to Eager Prims
            - Storing paths instead of edges
        - Runtime?
- Dijkstra’s is correct only when all edge weights >= 0

Bellman-Ford’s algorithm

- Set a distance value of Double.POSITIVE_INFINITY for all vertices
- Initialize a FIFO Q
- distance[start] = 0
- add start to Q
- While Q is not empty:
    - cur = pop a vertex from Q
    - For each neighbor x of cur:
        - Compute tentative distance from start to x through cur
            - distance[cur] + weight of edge between cur and x
        - if tentative distance < distance[x]
            - Update distance[x]
            - add x to Q if not already there

Bellman-Ford's example



FIFO Q:

0
1
2
3
4
5

	Distance	Parent
0	0	--
1	7	0
2	9	0
3	11	2
4	21	2
5	20	3

Analysis of Bellman-Ford’s algorithm

- How to implement?
- Runtime?
- Bellman-Ford’s is correct even when there are negative edge weights in the graph