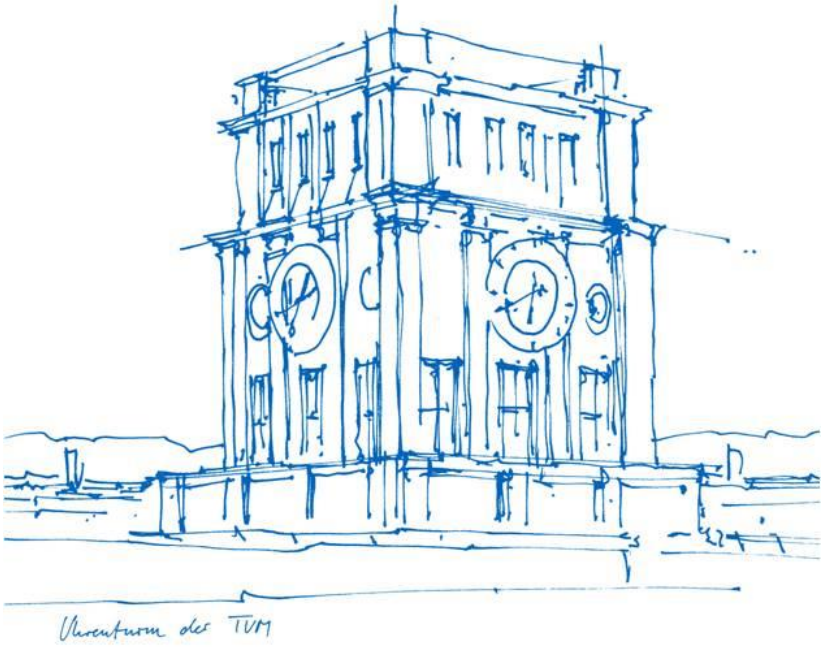
Graph storage: How good is CSR really?

Mahammad Valiyev

Technische Universität München

Informatics

12. December 2017



Agenda

- Introduction
- Implementation
- Evaluation

Graph

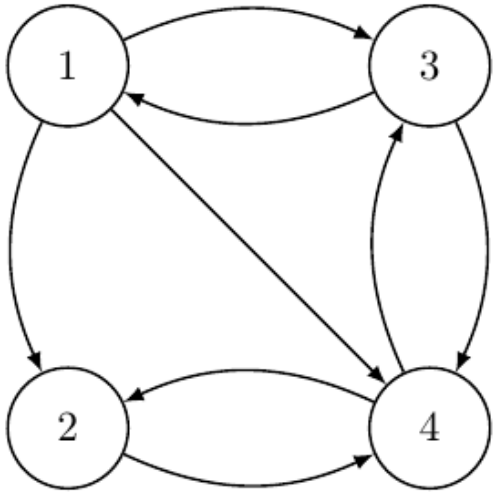
- Vertices
- Edges:
  - Directed/Undirected
  - Weighted/Unweighted

Representations of graph

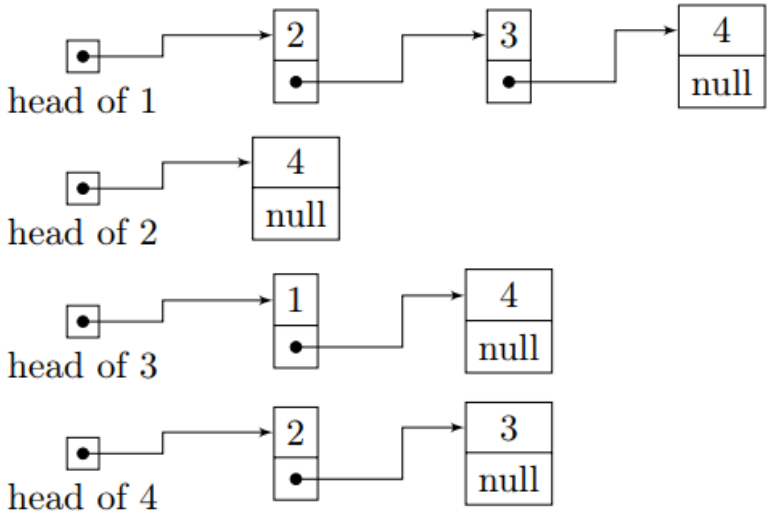
- Adjacency Matrix
- Adjacency List
- Compressed Sparse Row

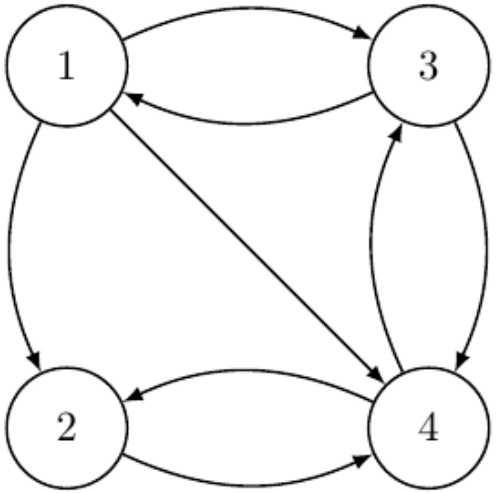
Adjacency Matrix

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

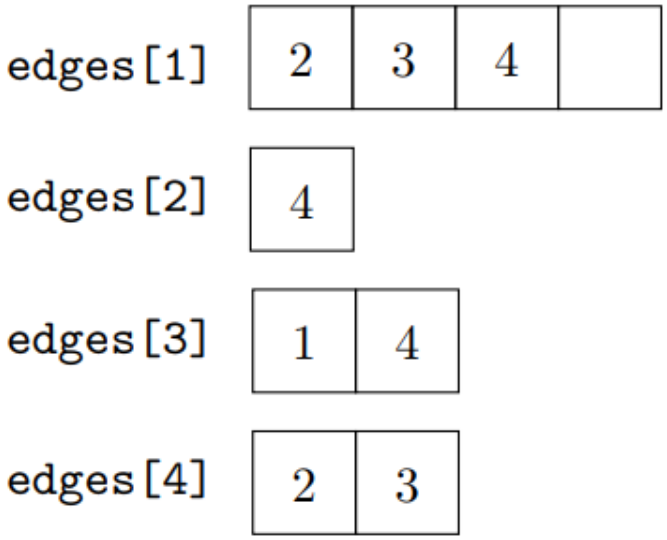


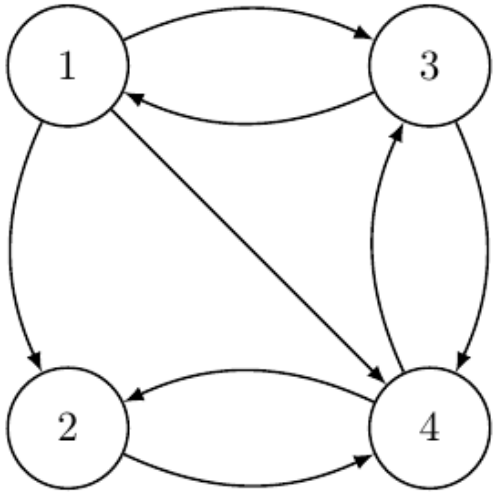
Adjacency List using std::list



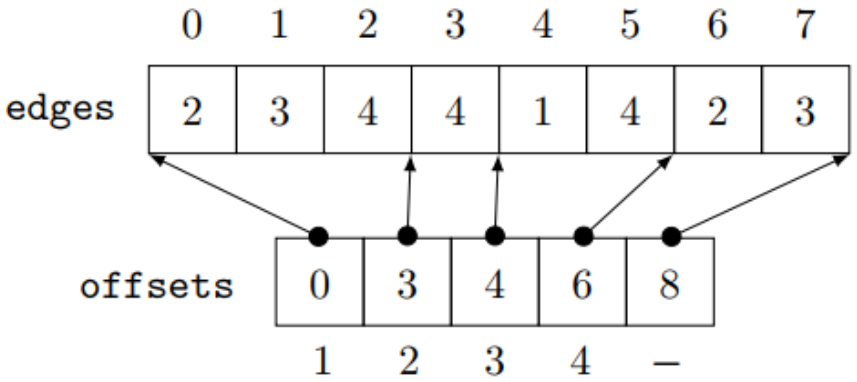


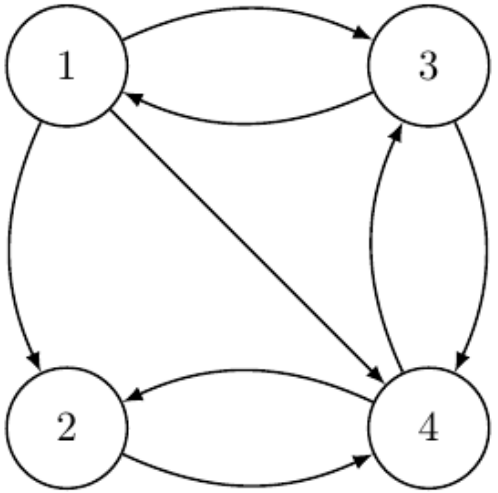
Adjacency List using std::vector





Compressed Sparse Row





Implementation

- Graph Containers:
  - Compressed Sparse Row:
    - Simple Update
    - Light Update
  - Adjacency List:
    - Implemented with std::list
    - Implemented with std::vector

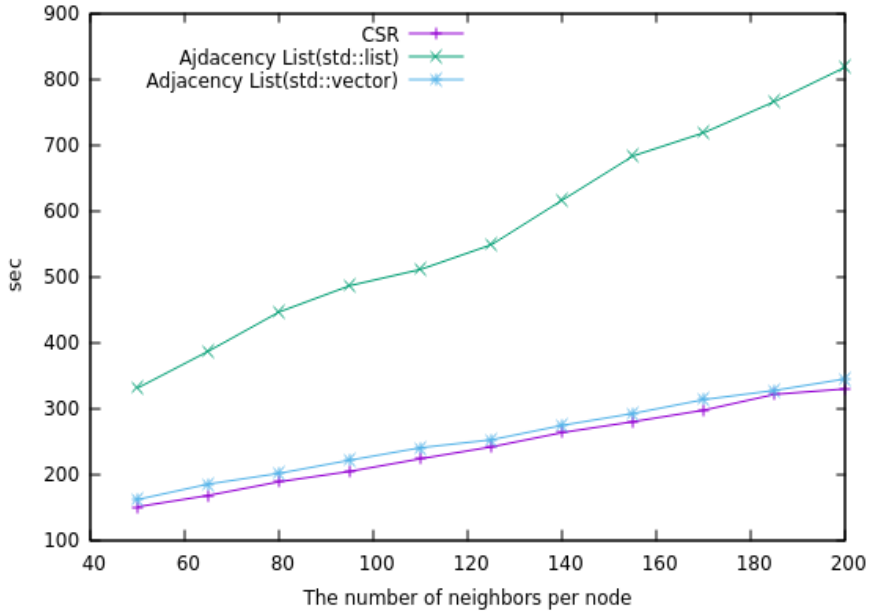
- Algorithms:
  - Depth-First Search
  - Breadth-First Search
  - Dijkstra Algorithm

Evaluation: Platform

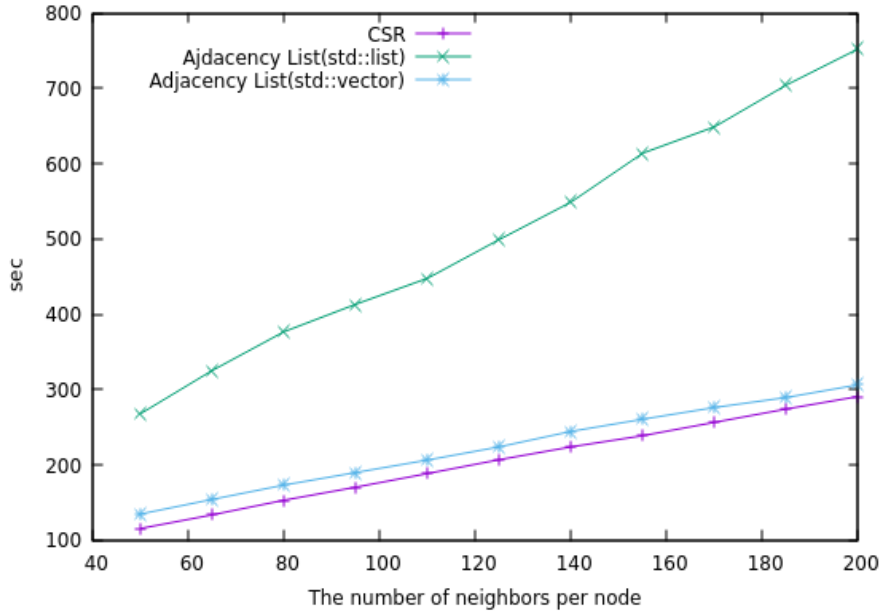
- OS: Ubuntu
- Processor: Intel(R) Core(TM) i7-3930K
- Frequency: 3.20GHz
- Memory: 64 Gb

- Dataset:
  - # vertices: 1,000,000
  - # neighbors per node differs from 50 to 200
  - Neighbors are selected randomly
  - Directed and weighted

Evaluation

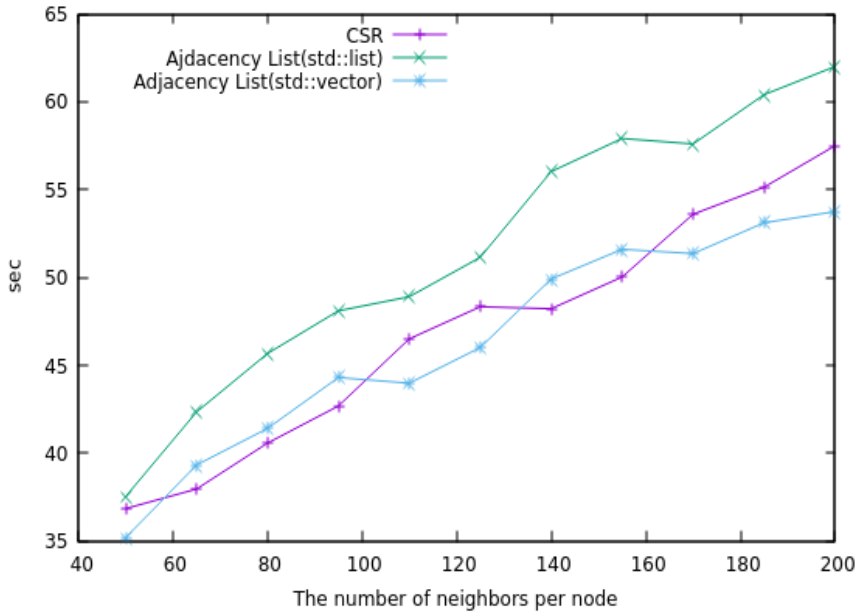


DFS



BFS

Evaluation

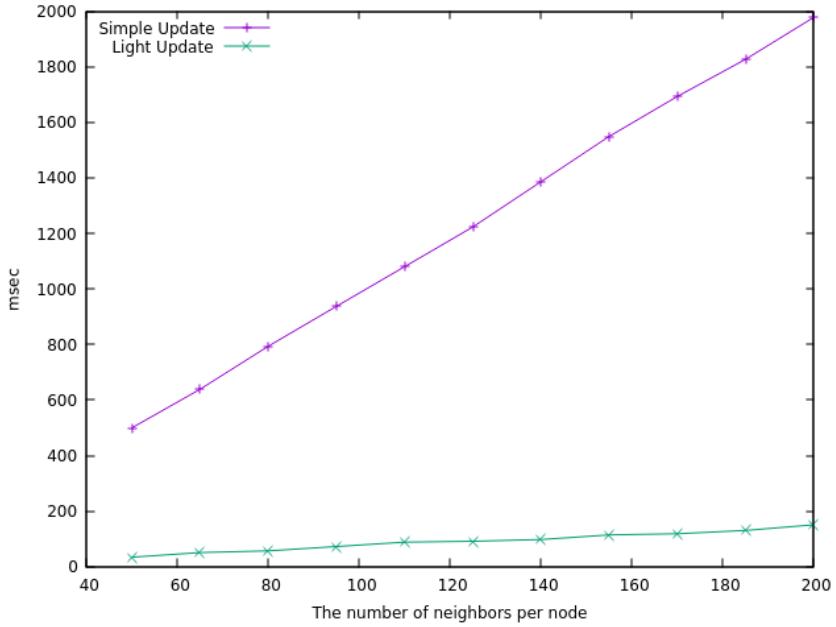


Dijkstra

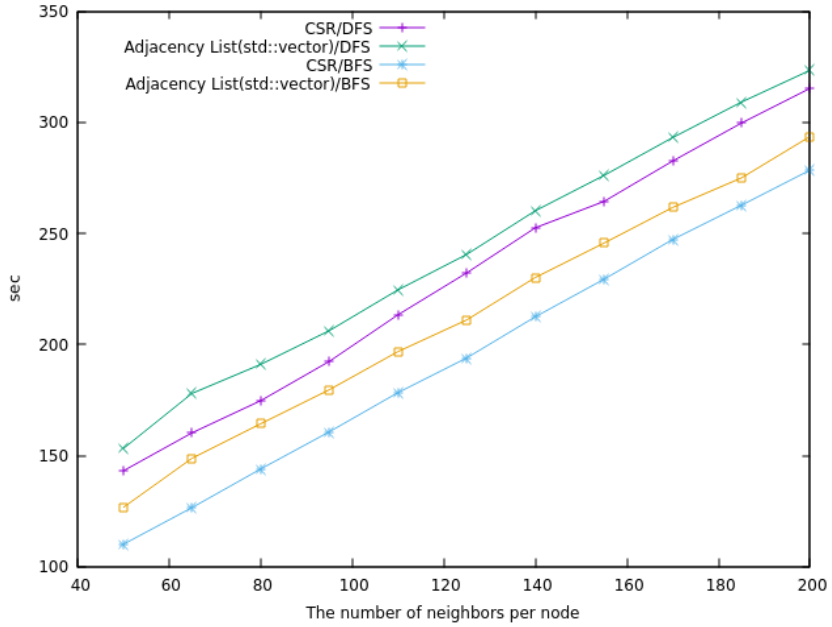
# neighbors per node	mem(std:list)÷mem(CSR)	mem(std:vector)÷mem(CSR)
50	7.33	1.41
65	7.46	2.01
80	7.55	1.66
95	7.62	1.42
110	7.67	1.23
125	7.7	1.15
140	7.73	1.85
155	7.76	1.68
170	7.78	1.54
185	7.79	1.42
200	7.81	1.31

Memory Consumption ratio

Evaluation



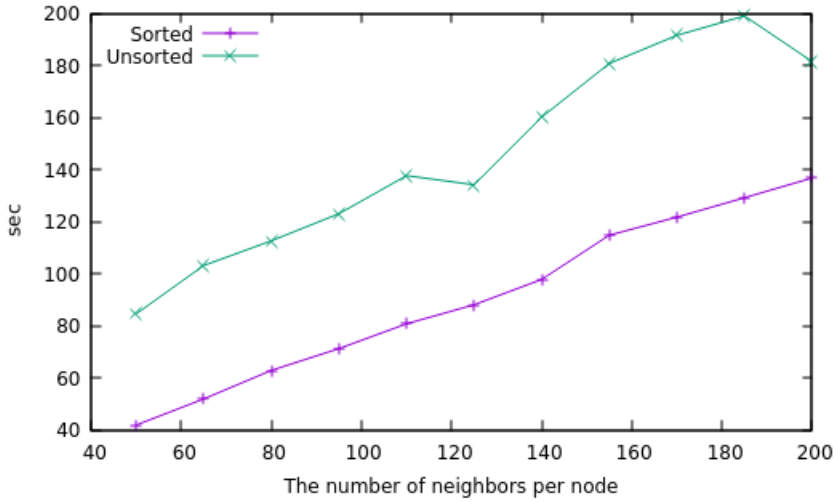
Update time



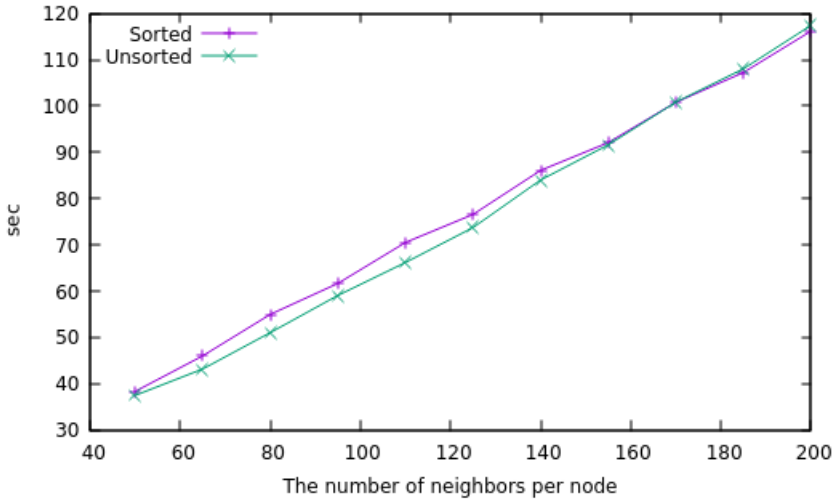
Real World Example

Evaluation

- Dataset:
  - # nodes: 1,000,000
  - # neighbors per node differs between 50 and 200
  - Neighbors have closer ids



DFS



BFS

Thanks for your attention