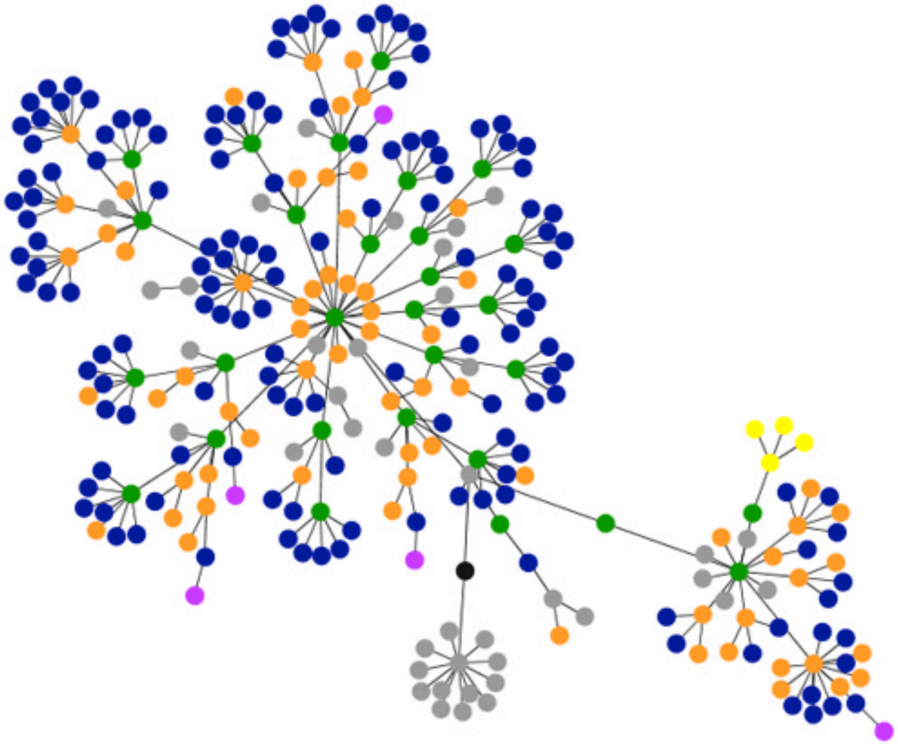


Minimum Spanning Tree

Fall 2013 - KAUST

Group Members

Name	ID
1. Mona Alshahrani.	124047
2. Maha Alaslani.	128694
3. Omniah Hussain Nagoor.	123923
4. Xiaopeng Xu	129052
5. yuxin chen	129115

+

Outline

- Minimum Spanning Tree (MST)
  1. Definition
  2. Applications
  3. Example
- Algorithms:
  1. Kruskal's Algorithm
     - Pseudocode
  2. Prim's Algorithm
     - Pseudocode
- Experiments
- Conclusions

MST Definition

- Input :

  Undirected graph G = (V,E)

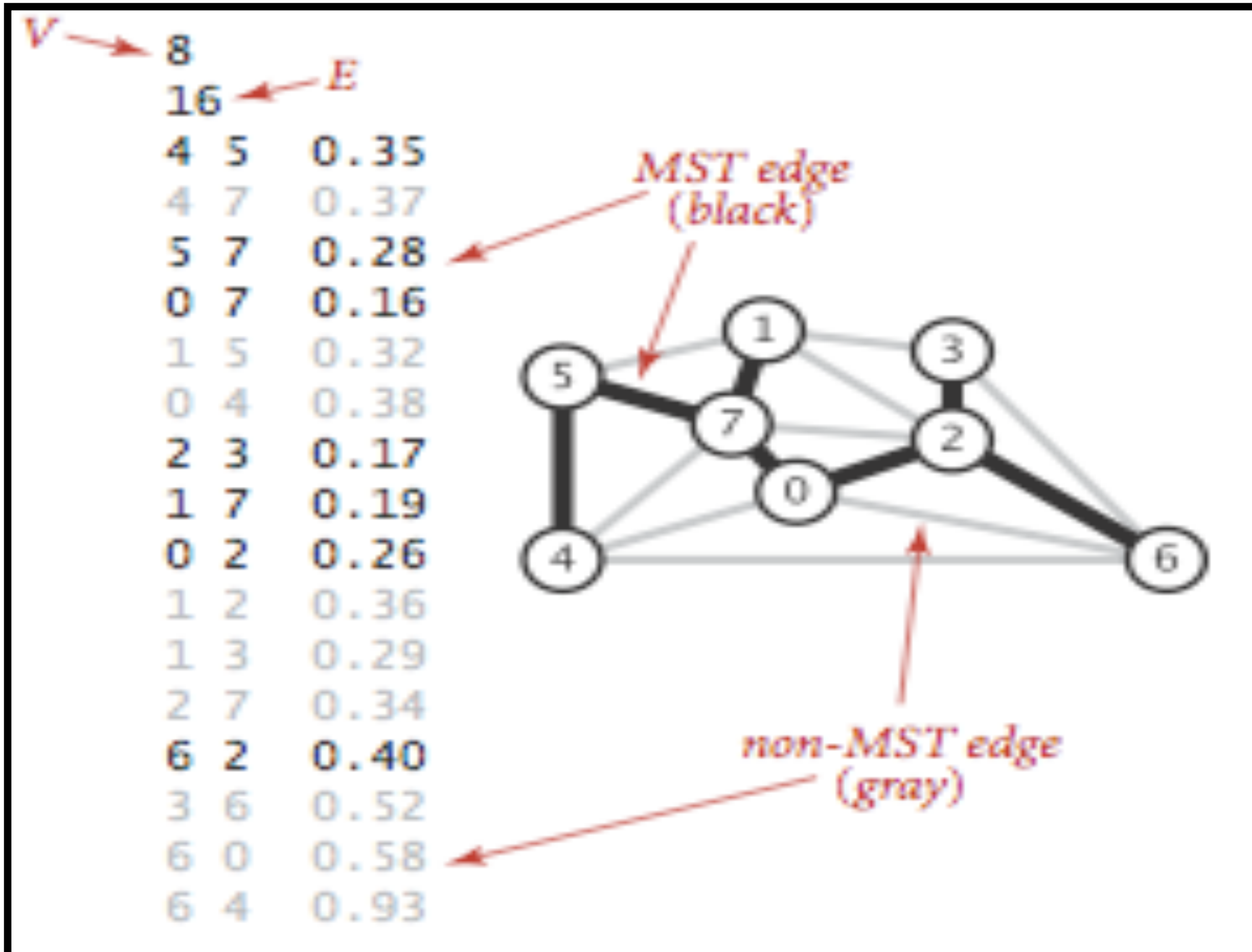
  Each edge (u,v) $\in E$ has a weight w(u,v) .

- Output :

  a tree T = (V, E'), where $E' \subseteq$ E, with the minimum cost

$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

+



V
8
16
E
4 5 0.35
4 7 0.37
5 7 0.28
0 7 0.16
1 5 0.32
0 4 0.38
2 3 0.17
1 7 0.19
0 2 0.26
1 2 0.36
1 3 0.29
2 7 0.34
6 2 0.40
3 6 0.52
6 0 0.58
6 4 0.93
MST edge (black)
non-MST edge (gray)
0
1
2
3
4
5
6
7

+

MST Applications

- Computer networks
- Transportations networks
- Water supply networks
- Electrical grids
- Image segmentation and more

Kruskal's Algorithm

- Kruskal's algorithm qualifies as a greedy algorithm.
- At each step it adds to the forest an edge of least possible weight.
- It uses a disjoint-set data structure to maintain several disjoint sets of elements.

Kruskal's Algorithm 2/2

Step 0: Set A=Ø and S=E, where E is the set of all edges.
Step 1: Sort the edges in non-decreasing order

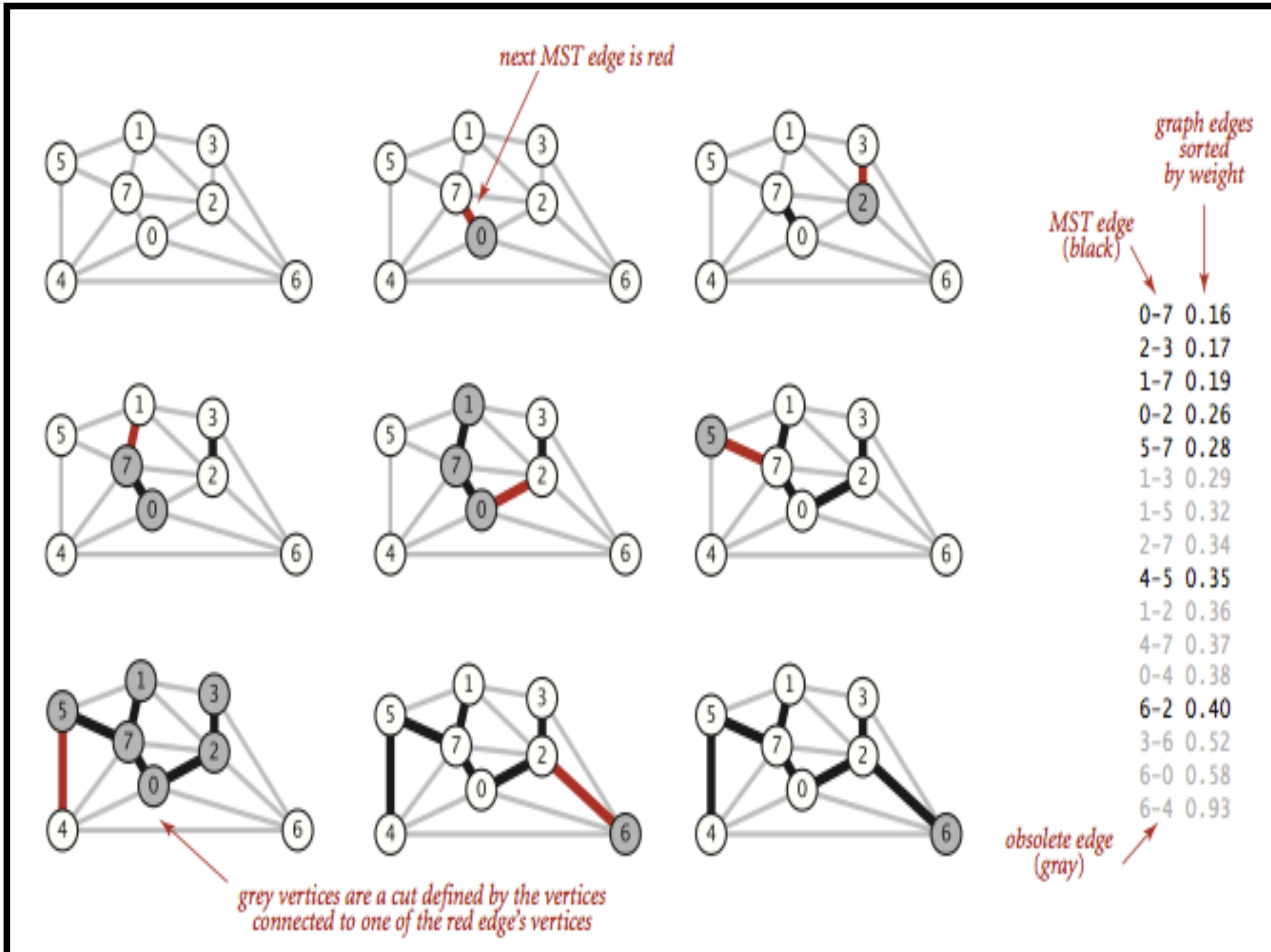
Step 2: choose e {e: minimum We ∈ S }

check that adding e to A doesn't create cycles.

If "Yes", remove e from set S.
If "No", move e from set S to set A.

Step 3: If S=Ø, output the minimal spanning. Else go to Step 1.

+



next MST edge is red
graph edges sorted by weight
MST edge (black)
0-7 0.16
2-3 0.17
1-7 0.19
0-2 0.26
5-7 0.28
1-3 0.29
1-5 0.32
2-7 0.34
4-5 0.35
1-2 0.36
4-7 0.37
0-4 0.38
6-2 0.40
3-6 0.52
6-0 0.58
6-4 0.93
obsolete edge (gray)
grey vertices are a cut defined by the vertices connected to one of the red edge's vertices

Prim’s Algorithm 1/2

- Prim’s algorithm is also qualifies as a greedy algorithm.
- At each step it finds a subset of the edges that forms a tree in which the total weight of all the edges in the tree is minimized.

Prim’s Algorithm 2/2

- **Step 0:** Choose any vertex v; set $S = \{v\}$ and $A=\emptyset$
- **Step 1:** Find a lightest edge such that one endpoint is in S and the other is in $V\setminus S$. Add this edge to A and its other endpoint to S.
- **Step 2:** If $V\setminus S = \emptyset$, output minimum spanning tree. Else go to Step 1.

Kruskal’s Pseudocode

```
MST-KRUSKAL (G,w)
A=Ø
for each vertex  v ∈ G.V
    make-set(v)
Sort the edges of G.E into nondecreasing order by weight
For each edge (u,v) ∈ G.E, taken in nondecreasing order by weight
    If FIND – SET (u)≠ FIND – SET (v)
        A = A  U {(u,v)}
        UNION (u,v)
Return A
```

Complexity Analysis 1/2

Kruskal's:

- Using disjoint-set
  - $O(|V|)$ *make-set* operations
  - $O(|E| \log |E|)$ sort
  - $O(|E|)$ *find-set* and *union* operations
  - Total running time : $O(|E| \log |V|)$

Prim's Pseudocode

```
MST-PRIM (G,w,r)
    for each u ∈ G.V
        u.key = ∞
        u.π= NIL
    r.key =0
    Q=G.V
    While Q ≠ Ø
        u=EXTRACT-MIN(Q)
        For each v ∈ G.Adj[u]
            If v ∈ Q and w (u,v) < v.key
                v.π=u
                v.key=w(u,v)
```

Complexity Analysis 2/2

Prim's:

- O(|V|) Extract-Min operation
- Extract-Min is O(log |V|)
- O(|E|) Decrease key operations
- Decrease key is O(|E| log |V|)
- Total running O(|E| Log |V|)

Experiments:

- Generating random graphs
- Implementing Prim's Adjacency
- Implementing Prim's Priority Queue
- Implementing Kruskal Priority Queue

Generating graphs

G(V,E) set V=3,E=3

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

adj [3] [3]

Generating graphs

G(V,E) set V=3,E=3

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

V={1,2,3}

InSet={1}

OutSet={2,3}

Pick up node 1 in InSet,
Pick up node 2 in OutSet

Randomly Generate corresponding weight w(1,2)=1

Generating graphs:

G(V,E) set V=3,E=3

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

V={1,2,3}
InSet={1,2}
OutSet={3}

Pick up node 1 in InSet,
Pick up node 3 in OutSet

Generate corresponding weight w(1,3)=2

Generating graphs:

G(V,E) set V=3,E=3

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix}$$

V={1,2,3}
InSet={1,2,3}
OutSet={}

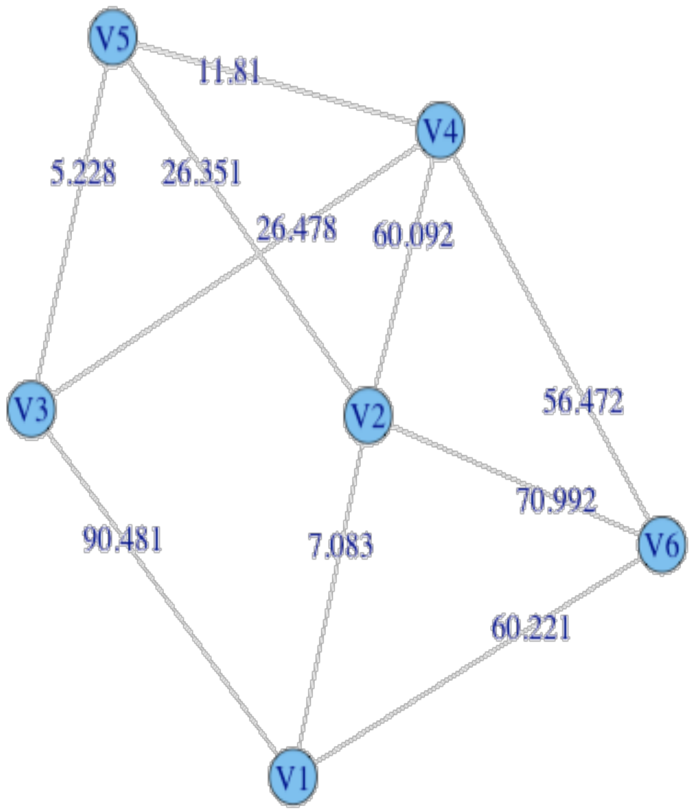
Pick up node 2 and node 3 in InSet

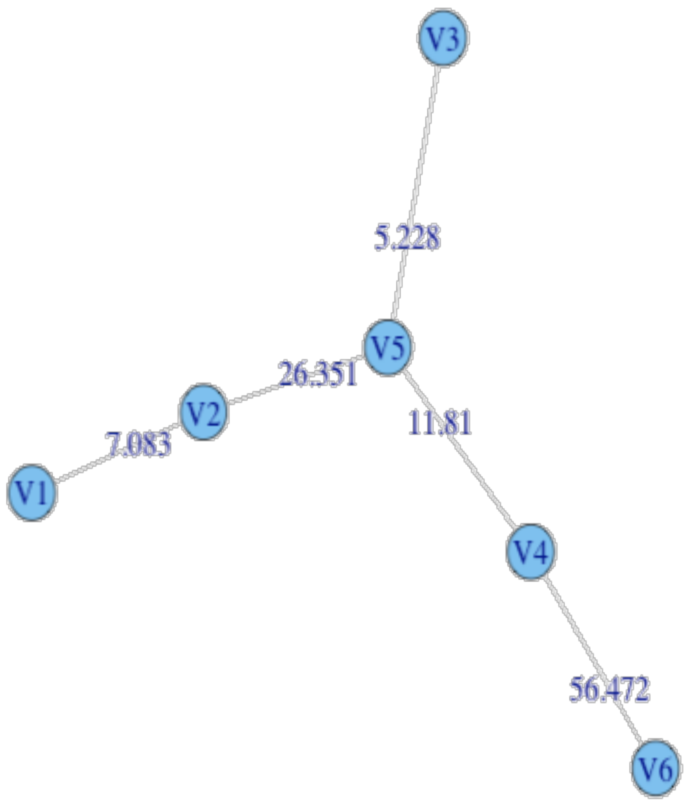
Generate corresponding weight w(2,3)=3

Experiments Results:

- Validating MST
- Running Time Calculation.
- Comparison of Running Time

Validating MST



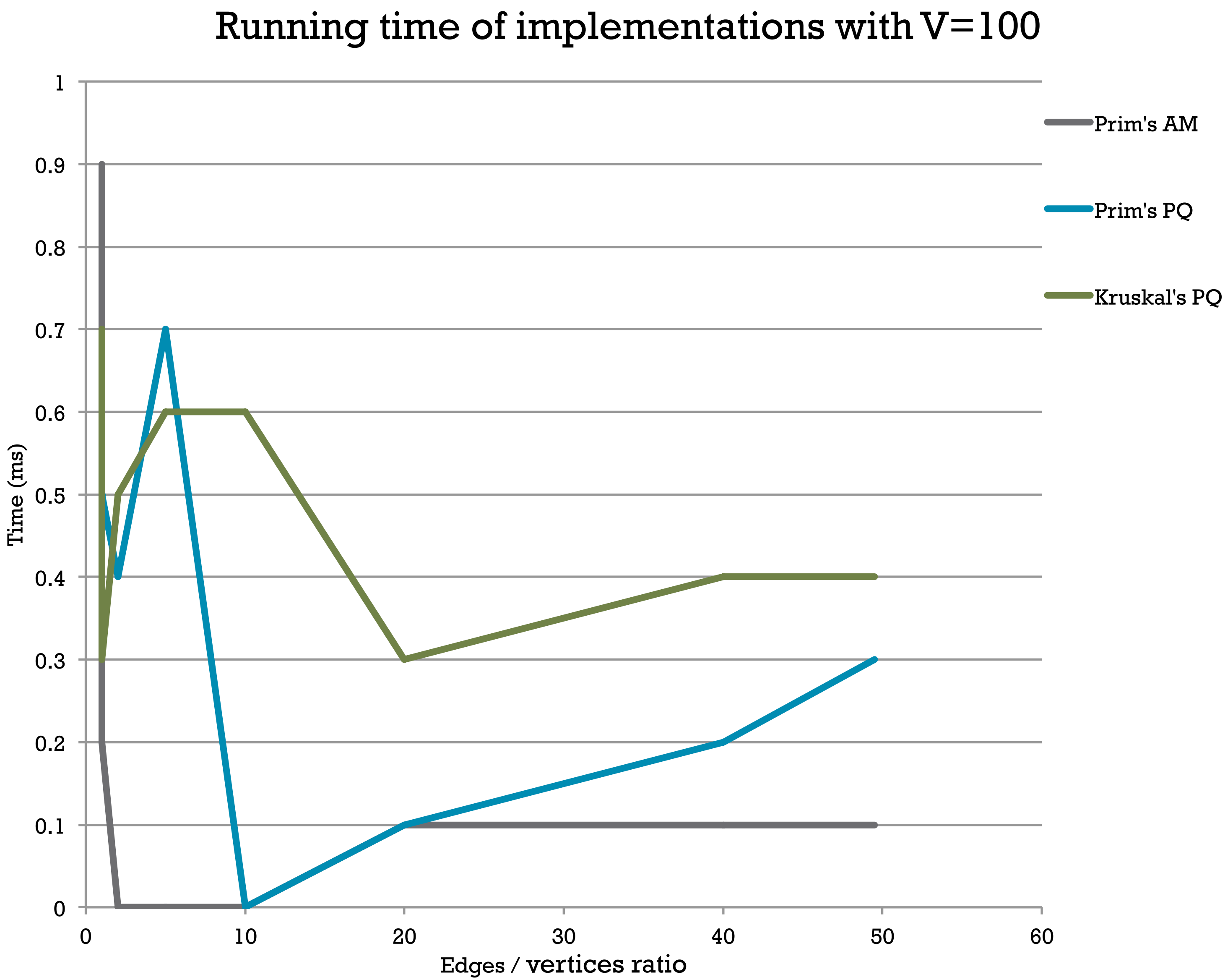


Validating MST

Algorithm & Data Structure	Prim’s PQ	Prim's AM	Kruskal's PQ
Weight of MST	106.94501	106.94501	106.94501

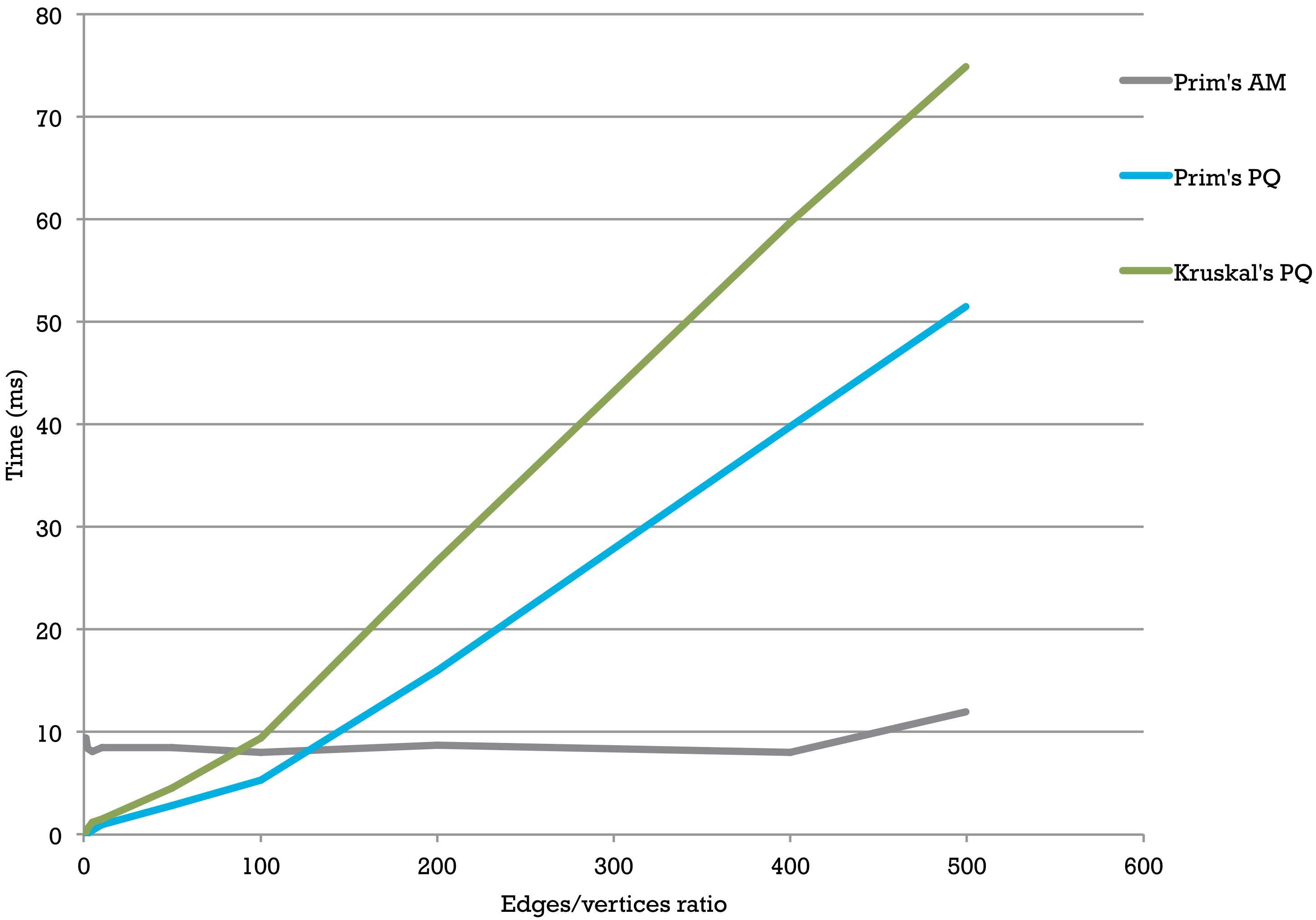
Calculation of Running Time

Node No.	Edge No.'s	Min Edge No.	Max Edge No.
100	99,100,200,500,1000,2000,4000,4950	99	4950
1000	999,1000,2000,5000,10000,50000, 100000,200000,400000,499500	999	499500
10000	9999,10000,20000,50000,100000,500000, 1000000,5000000,10000000,20000000, 40000000,49995000	9999	49995000

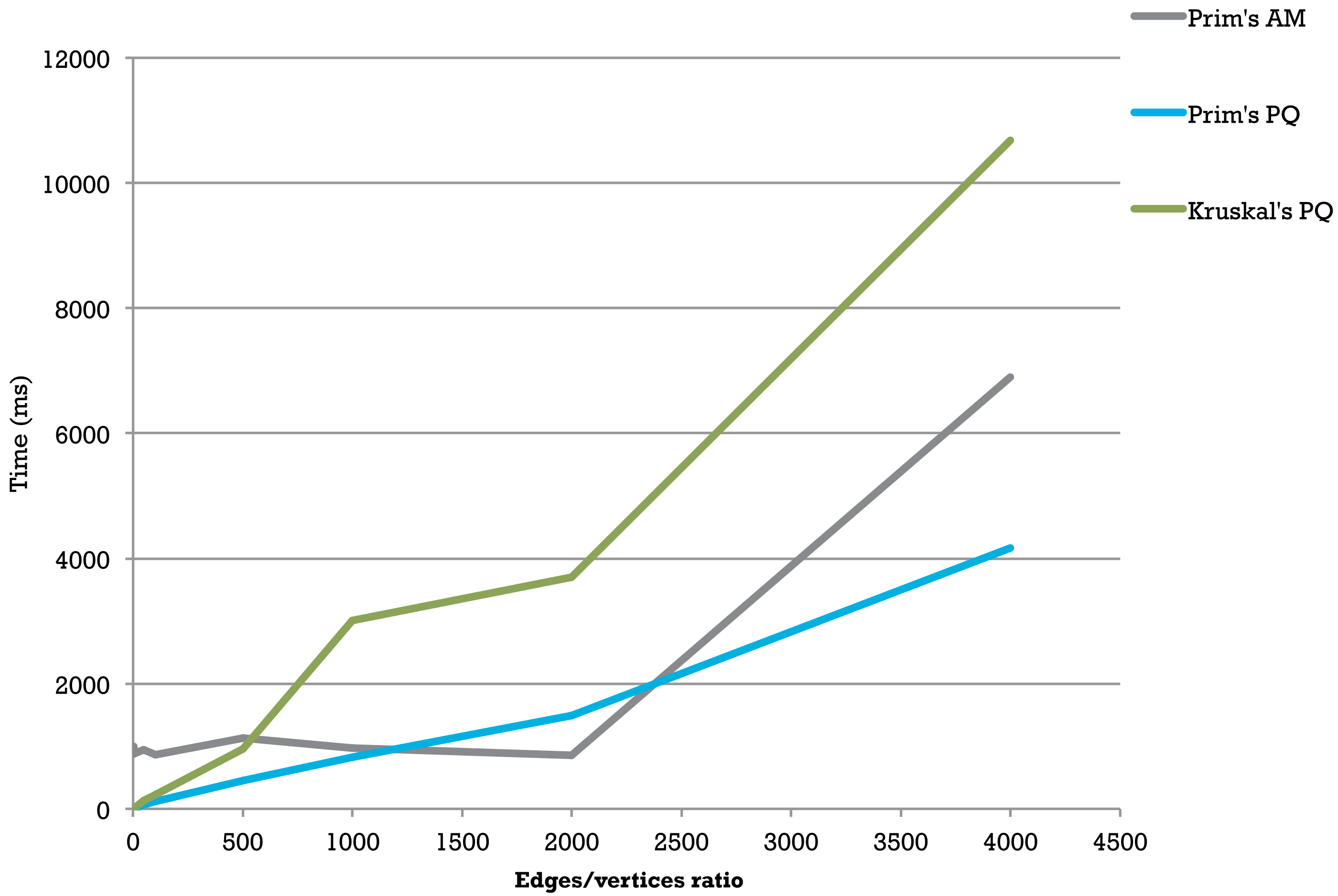

Running time of implementations with V=100
Time (ms)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
Edges / vertices ratio
Prim's AM
Prim's PQ
Kruskal's PQ

+

Running time of implementations with V=1K



Running time of implementations with V=10K



Conclusions

1. When there's only a small number of vertices (such as 100), the performance difference is not explicit.

2. When the V/E is small, the performance of Prim's AM is poor. But with V/E is growing bigger, Prim's AM's running time is not growing quickly. However, Prim's PQ and Kruskal's PQ's running time is growing in proportional to V/E.

3. When the graph is pretty huge, speed of disk read makes a big difference on performance of the algorithms.

+

References

1. Moret, B. M., & Shapiro, H. D. (1991). *An empirical analysis of algorithms for constructing a minimum spanning tree* (pp. 400-411). Springer Berlin Heidelberg.

2. Gabow, H. N., Galil, Z., Spencer, T., & Tarjan, R. E. (1986). Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, *6*(2), 109-122.

3. Cromen,Leiserson,Rivest & Stein. (2009).''Introduction to Algorithms''.MIT Press.Cambridge ,MA.202,pages,631-636

4. Kubica, J. (Performer). Minimum Spanning Trees, Prim's Algorithm [Web Photo]. Retrieved from http://computationaltales.blogspot.com/2011/08/minimum-spanning-trees-prims-algorithm.html

Thanks

- Prof. Mikhail Moshkov
- You

Any Question