# Shortest Path Search on the Database

And more ...

Georepublic

**“The various religions are like different roads converging on the same point.**

**What difference does it make if we follow different routes, provided we arrive at the same destination?”**

Mahatma Gandhi

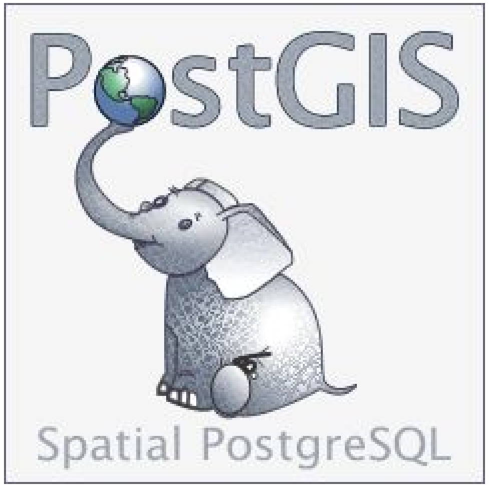
MKGandhi

# What is *pgRouting*?

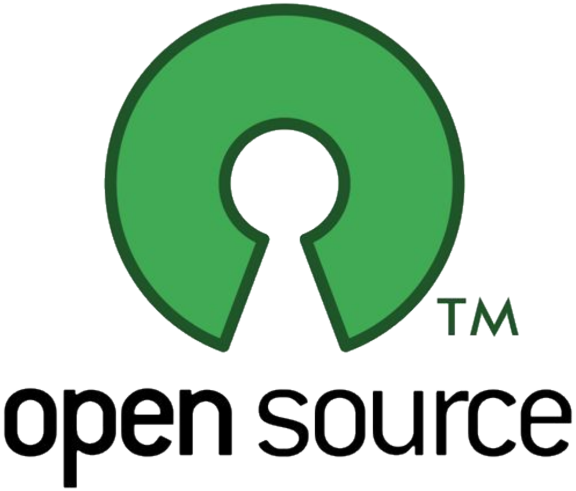
Georepublic

# a Library

# an extension



# An open source project



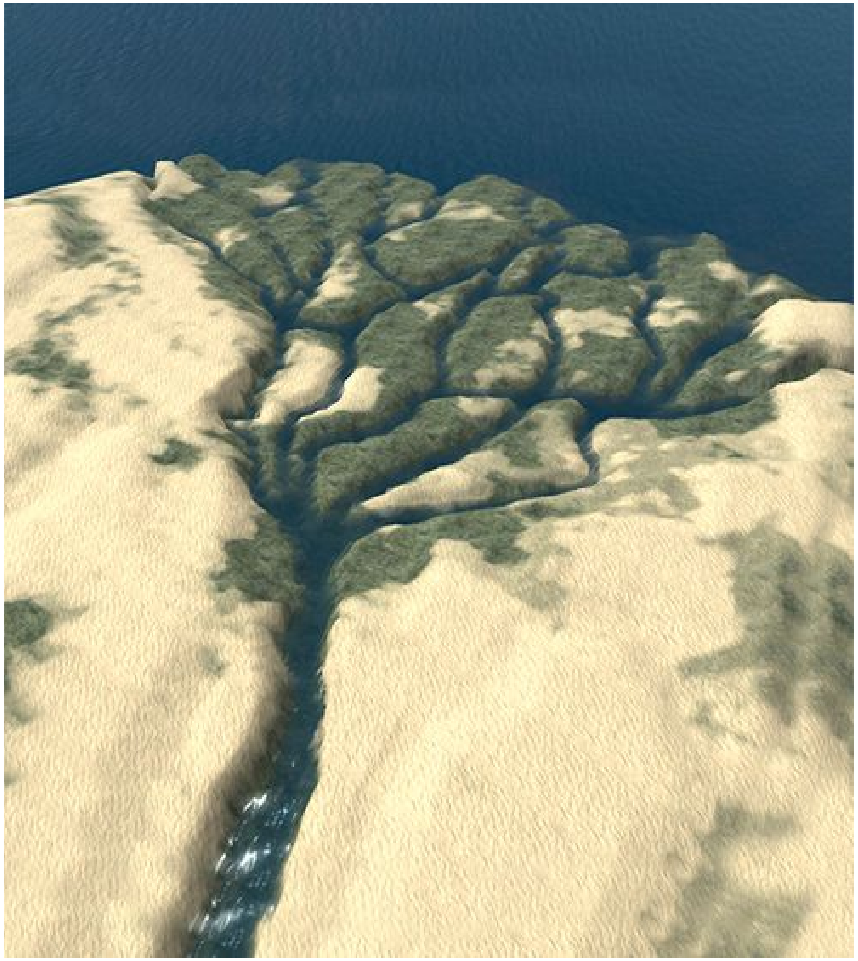
# A community project

# NOT

I am not a front end!

But, I can be used to create one.

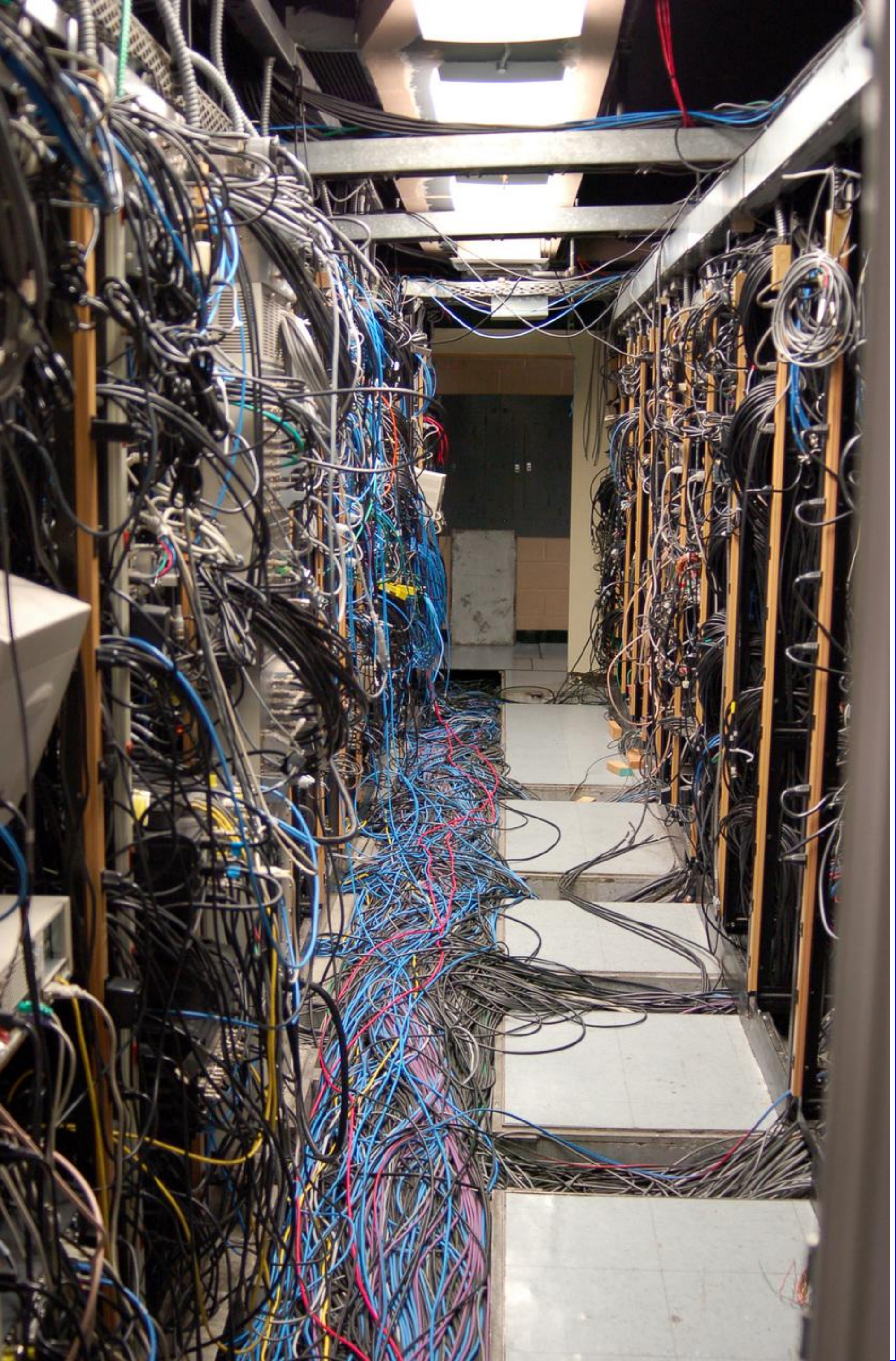
# ALL ABOUT THAT GRAPH



# RIVERS

# RELATIONSHIPS

Georepublic



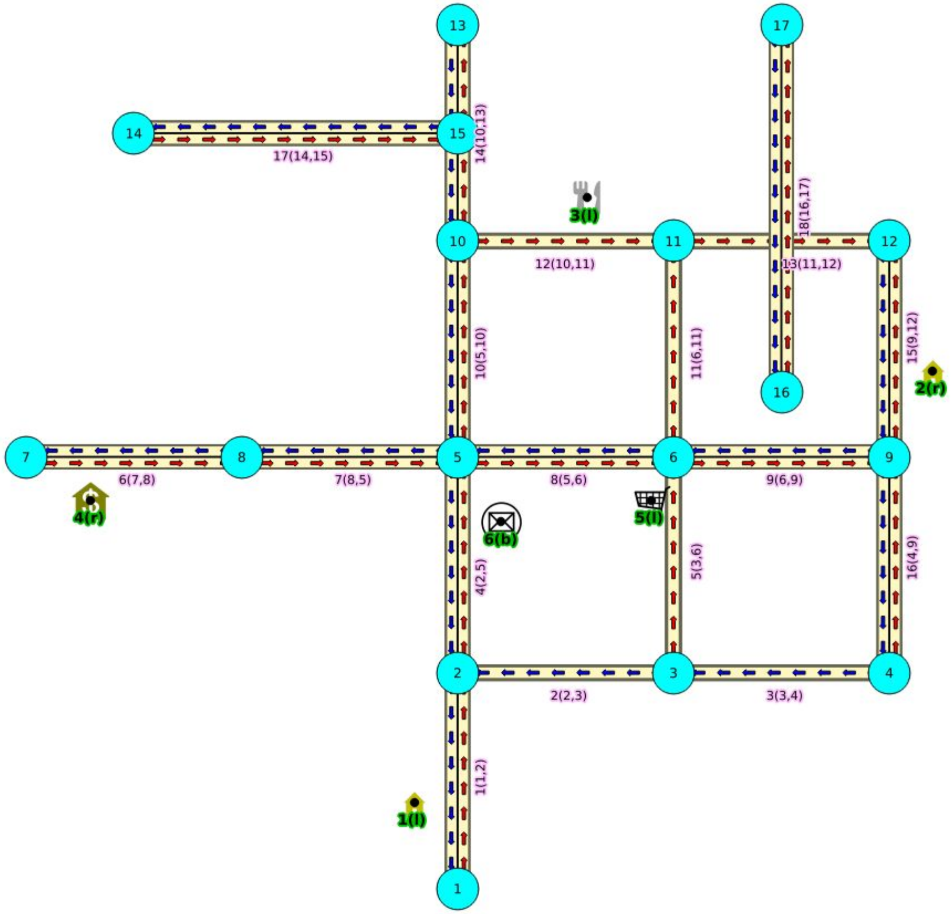
# COMMUNICATIONS

Georepublic

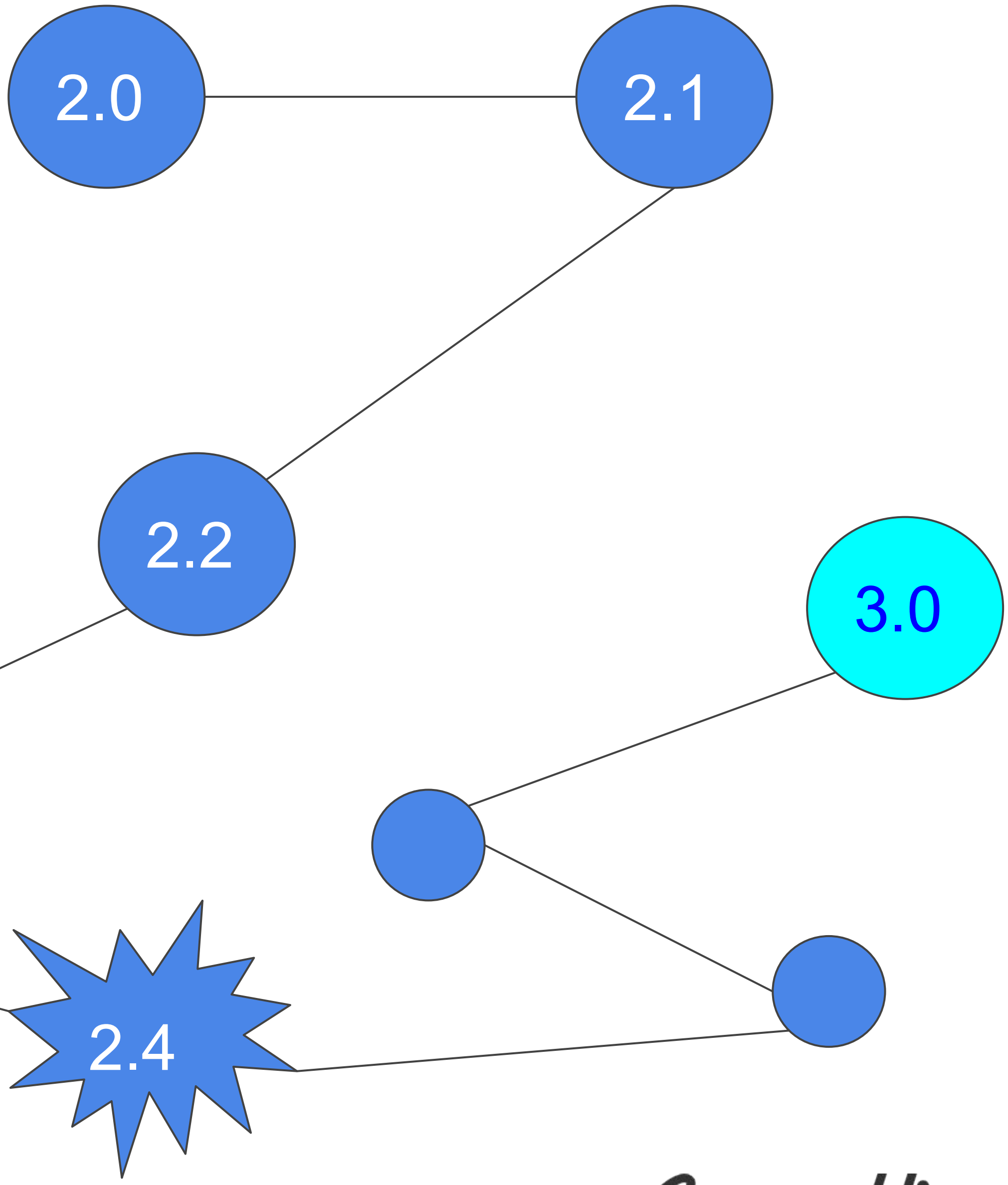
# ROADS

Georepublic

# THE REPRESENTED ROADS



# EVOLUTION


2.0
2.1
2.2
3.0
2.3
2.4

2.0

# 2013

- *pgr_dijkstra*
- *pgr_drivingDistance*
- *pgr_ksp*
- *pgr_apspJohnson*
- *pgr_apspWarshall*
- *pgr_kDijkstra*
- *pgr_astar*
- *pgr_bdAstar*
- *pgr_bdDijkstra*
- *pgr_tsp*
- *pgr_trsp*
- *pgr_alphaShape*
- *pgr_pointsAsPolygon*

2.1

# SEP-2015

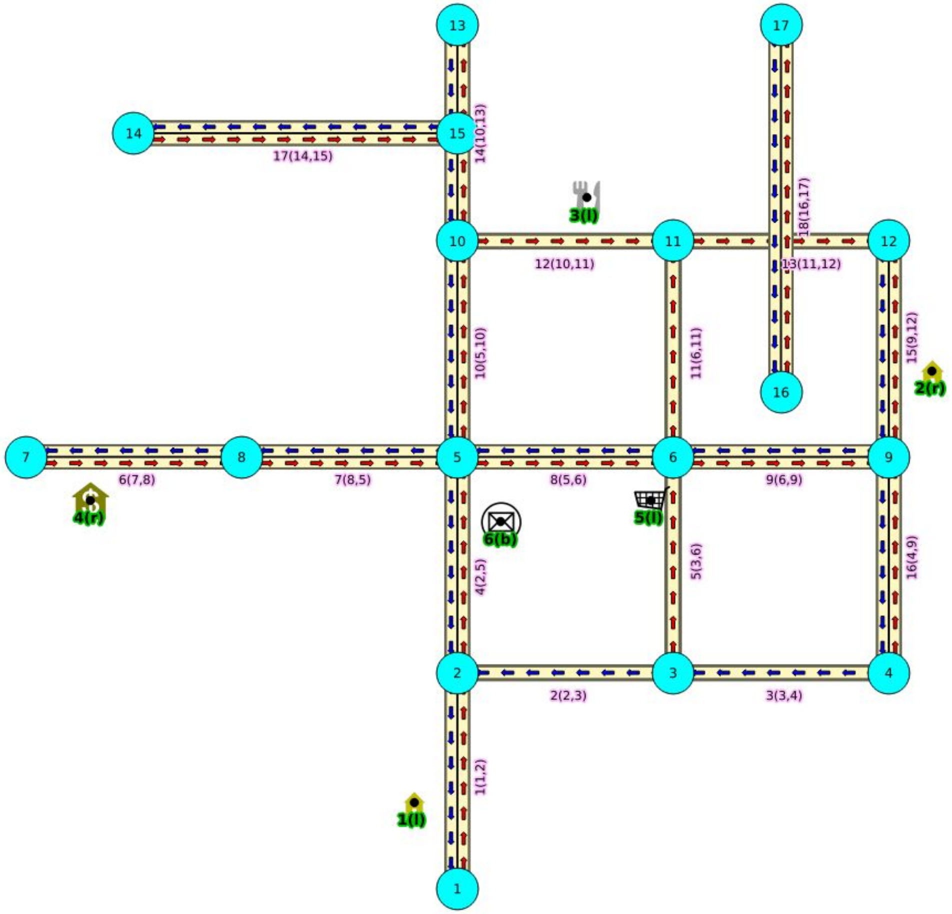
- *pgr_dijkstra*
- *pgr_drivingDistance*
- *pgr_ksp*
- *pgr_apspJohnson*
- *pgr_apspWarshall*
- *pgr_kDijkstra*
- *pgr_astar*
- *pgr_bdAstar*
- *pgr_bdDijkstra*
- *pgr_tsp*
- *pgr_trsp*
- *pgr_alphaShape*
- *pgr_pointsAsPolygon*

- *pgr_dijkstra*
  - *One to Many*
  - *Many to One*
  - *Many to Many*
- *pgr_trspViaVertices*
- *pgr_trspViaEdges*
- *pgr_labelGraph*
- *pgr_oneDepot*
- *pgr_gsoc_vrppdtw*

GSoC Students 2013 - 2014
Thank you Razequl & Manikata!

Georepublic

# Go from vertex 2 to vertex 3



Georepublic

# THE QUERY

```
SELECT * FROM pgr_dijkstra('
      SELECT id,
             source,
             target,
             cost,
             reverse_cost,
        FROM edge_table',
    2, 3);
```

# THE RESULT

```
 seq | path_seq | node | edge | cost | agg_cost
-----+----------+------+------+------+----------
   1 |        1 |    2 |    4 |    1 |        0
   2 |        2 |    5 |    8 |    1 |        1
   3 |        3 |    6 |    9 |    1 |        2
   4 |        4 |    9 |   16 |    1 |        3
   5 |        5 |    4 |    3 |    1 |        4
   6 |        6 |    3 |   -1 |    0 |        5
(6 rows)
```

# 2.2 MAR-2016

- *pgr_dijkstra(group)*
- *pgr_drivingDistance*
- *pgr_ksp*
- *pgr_johnson*
- *pgr_floydWarshall*
- *pgr_dijkstraCost (group)*
- *pgr_apspJohnson*
- *pgr_apspWarshall*
- *pgr_kDijkstra*
- pgr_astar
- pgr_bdAstar
- pgr_bdDijkstra
- pgr_tsp
- pgr_trsp(group)
- pgr_alphaShape
- pgr_pointsAsPolygon

- pgr_labelGraph
- pgr_oneDepot
- pgr_gsoc_vrppdtw
- *pgr_withPoints(group)*
- *pgr_withPointsCost(group)*
- *pgr_withPointsDD*
- *pgr_withPointsKSP*
- *pgr_dijkstraVia*

Georepublic

# 2.3 SEP-2016

- *pgr_dijkstra*
- *pgr_drivingDistance*
- *pgr_ksp*
- *pgr_Johnson*
- *pgr_floydWarshall*
- *pgr_dijkstraCost*
- *pgr_tsp*
- *pgr_TSP*
- *pgr_astar*
- *pgr_euclideanTSP*
- *pgr_astar*
- pgr_bdAstar
- pgr_bdDijkstra
- pgr_trsp
- pgr_alphaShape
- pgr_pointsAsPolygon

- pgr_labelGraph
- pgr_oneDepot
- pgr_gsoc_vrppdtw
- *pgr_withPoints(group)*
- *pgr_withPointsCost(group)*
- *pgr_withPointsDD*
- *pgr_withPointsKSP*
- *pgr_dijkstraVia*
- *pgr_dijkstraCostMatrix*
- *pgr_withPointsCostMatrix*

2.3

# GSoC Students

- *pgr_maxFlowPushRelabel(group)*
- *pgr_maxFlowEdmondsKarp(group)*
- *pgr_maxFlowBoykovKolmogorov(group)*
- *pgr_maximumCardinalityMatching*
- *pgr_edgeDisjointPaths*
- *pgr_contractGraph*

2015 GSoC Student
Thank you Rohith!

# CONTRACTION

# CONTRACTION

- Graph Contraction, when working on big graphs:
  - road graphs,
  - electric networks
- Speeds up some graph algorithms.
- The current implementation:
  - Flexible Framework
    - “Easy” to add a new operation.
  - Dead end contraction
  - Linear contraction

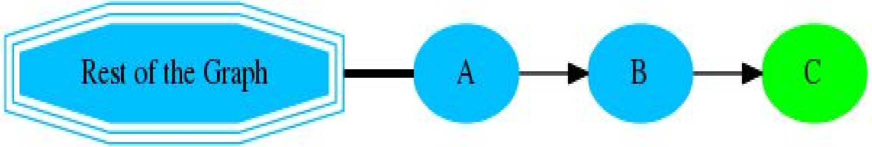
# The contraction skeleton

- An initial set up that may involve analyzing the graph given as input and setting the non contractible nodes or edges.
- A cycle that will go and perform a contraction operation until/while possible, and then move to the next contraction operation.
- Adding a new operation then becomes an “easy” task:
  - Add new contraction operation class.
  - Add some interaction between contractions.
- Currently, there are two implemented operation for contracting a graph
  - Dead End contraction
  - Linear contraction

# DEAD END

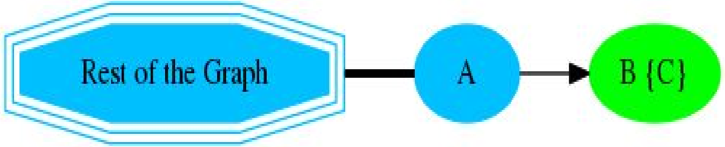
Undirected Graph

- The number of adjacent vertices is one.

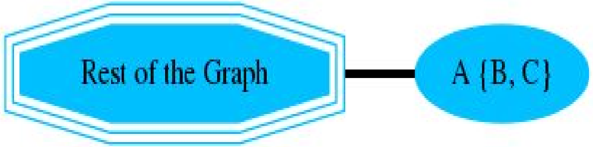


Directed Graph

- Case 1
  - No outgoing edges
  - At least one incoming edge.



- Case 2
  - One incoming edge
  - One outgoing edge
  - Same identifier on the edges.

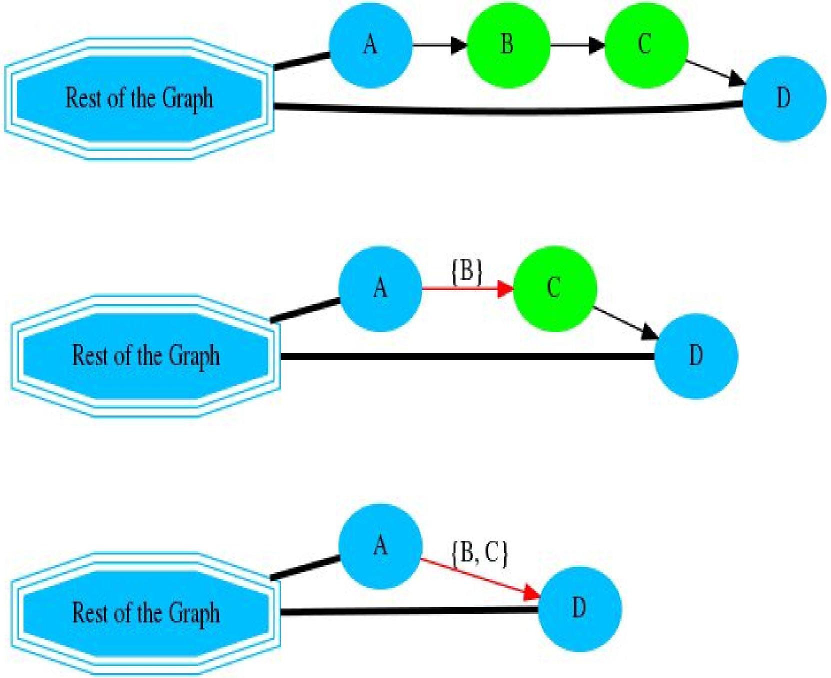


# LINEAR

### Linear Node

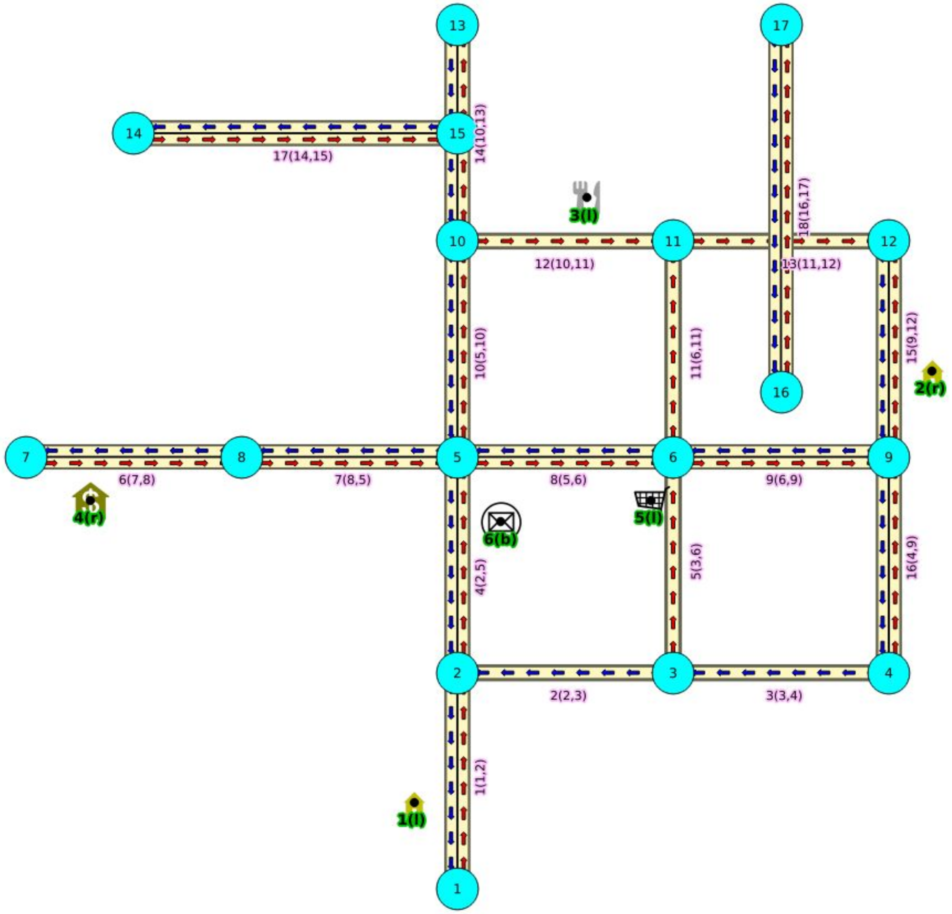
- Two adjacent vertices.
- At least one incoming edge and one outgoing edge.

### Linear Contraction

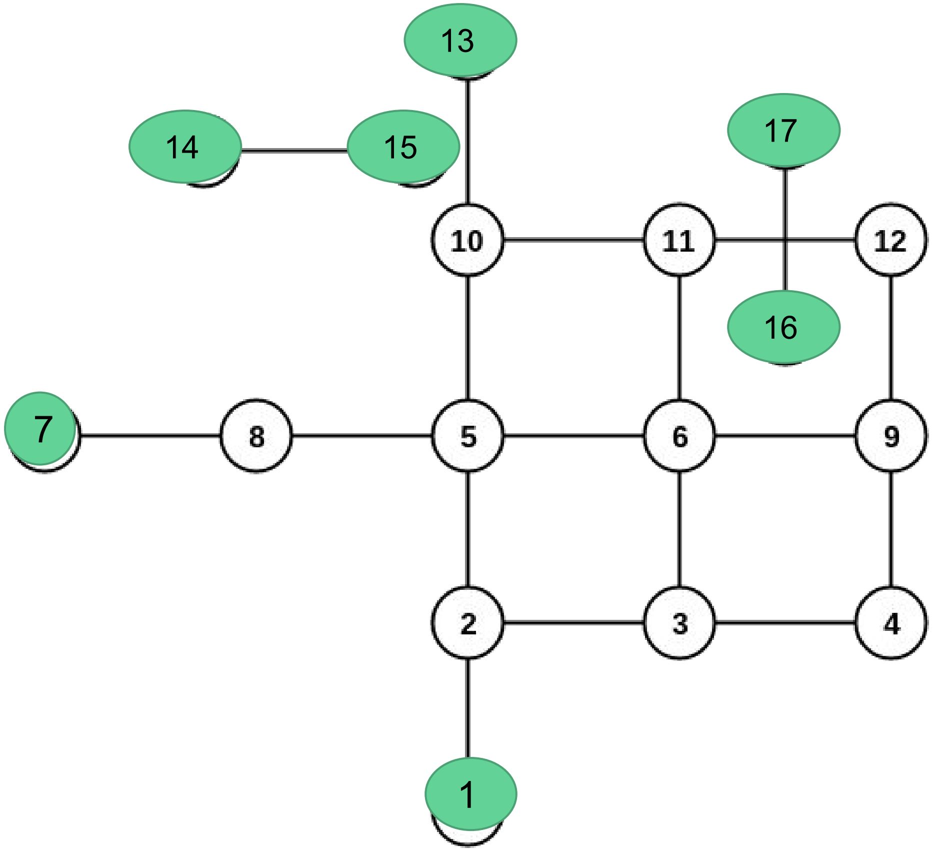


# EXAMPLE

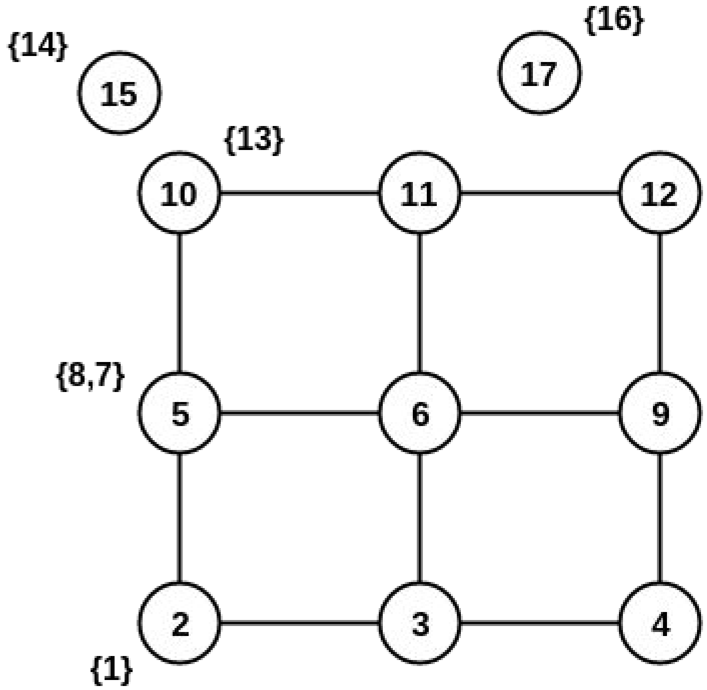
# THE REPRESENTED ROADS



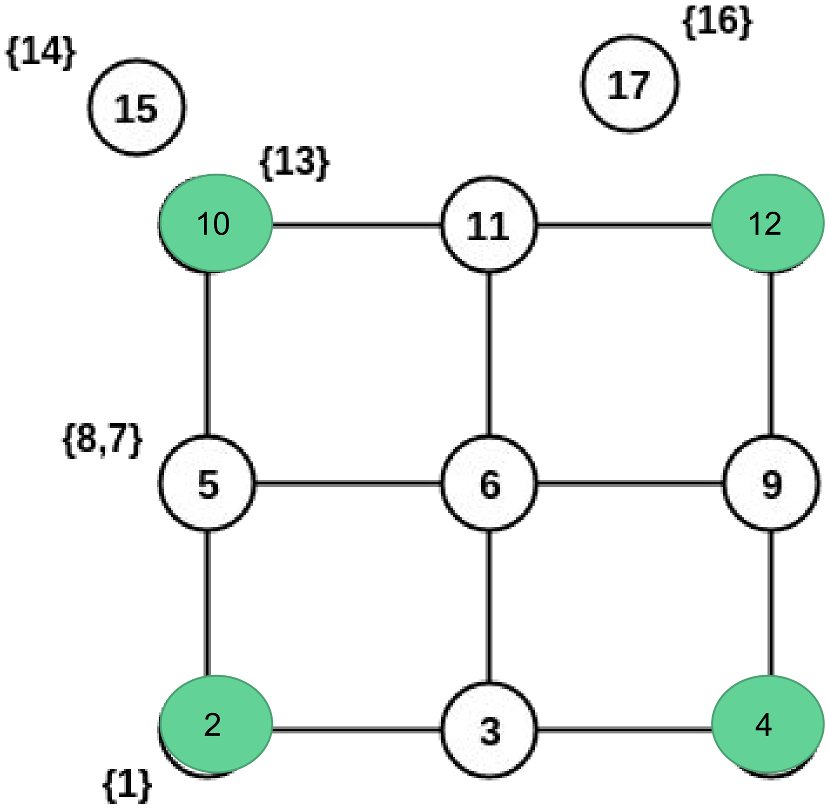
# DEAD END



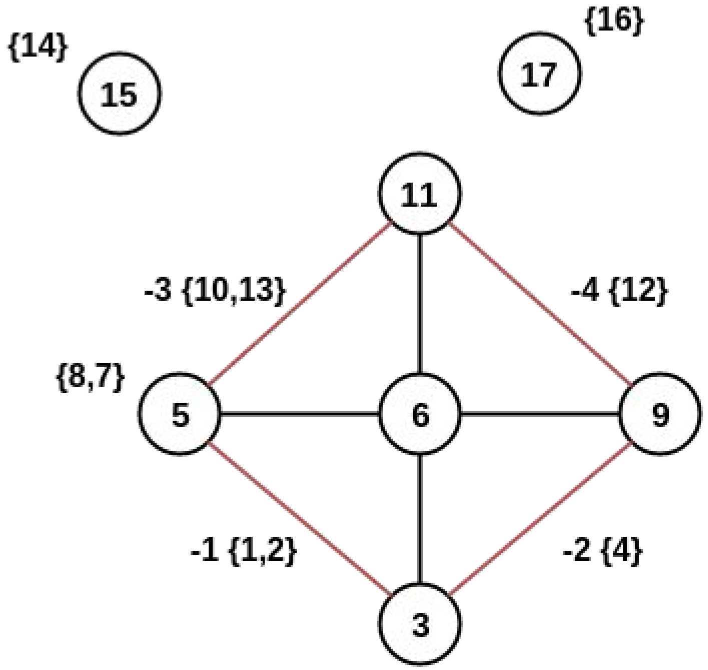
# AFTER CONTRACTION



# LINEAR



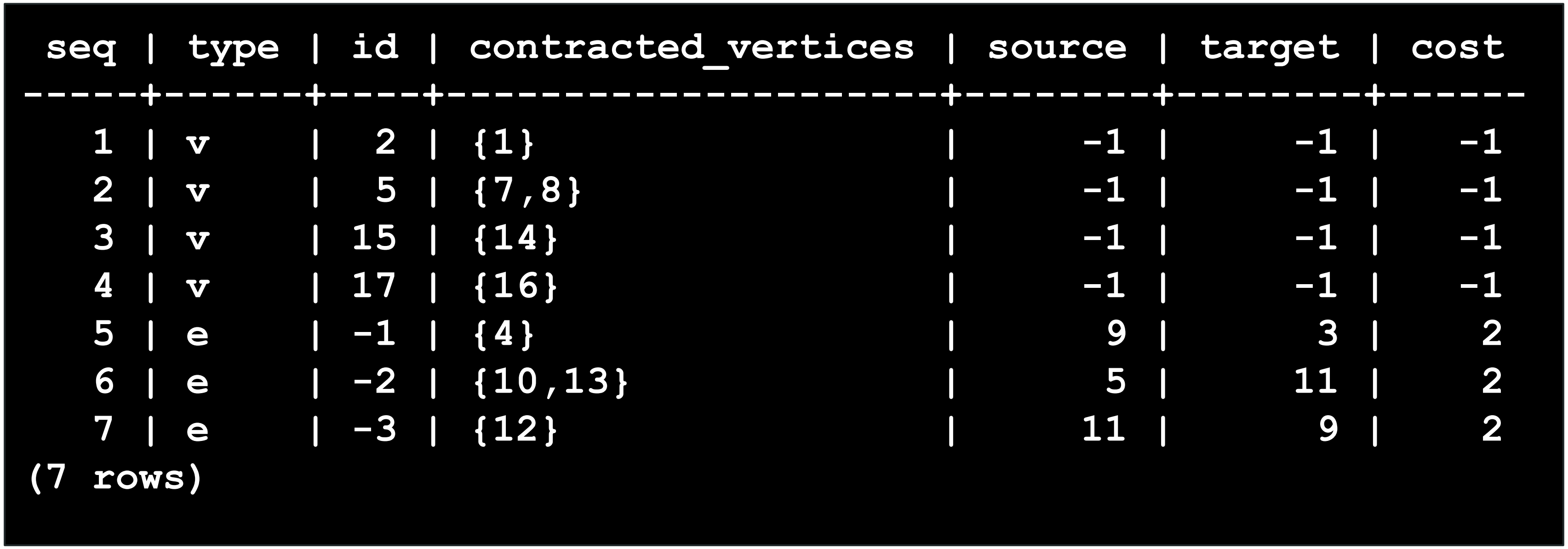
# AFTER CONTRACTION



# THE QUERY

```
SELECT *
FROM pgr_contractGraph(
    'SELECT id,
        source,
        target,
        cost,
        reverse_cost
     FROM edge_table',
     ARRAY[1, 2]);
```

# THE RESULTS



```
 seq | type | id | contracted_vertices | source | target | cost
-----+------+----+---------------------+--------+--------+------
   1 | v    |  2 | {1}                 |     -1 |     -1 |   -1
   2 | v    |  5 | {7,8}               |     -1 |     -1 |   -1
   3 | v    | 15 | {14}                |     -1 |     -1 |   -1
   4 | v    | 17 | {16}                |     -1 |     -1 |   -1
   5 | e    | -1 | {4}                 |      9 |      3 |    2
   6 | e    | -2 | {10,13}             |      5 |     11 |    2
   7 | e    | -3 | {12}                |     11 |      9 |    2
(7 rows)
```

2016 GSoC Student
Thank you Andrea!

# FLOW

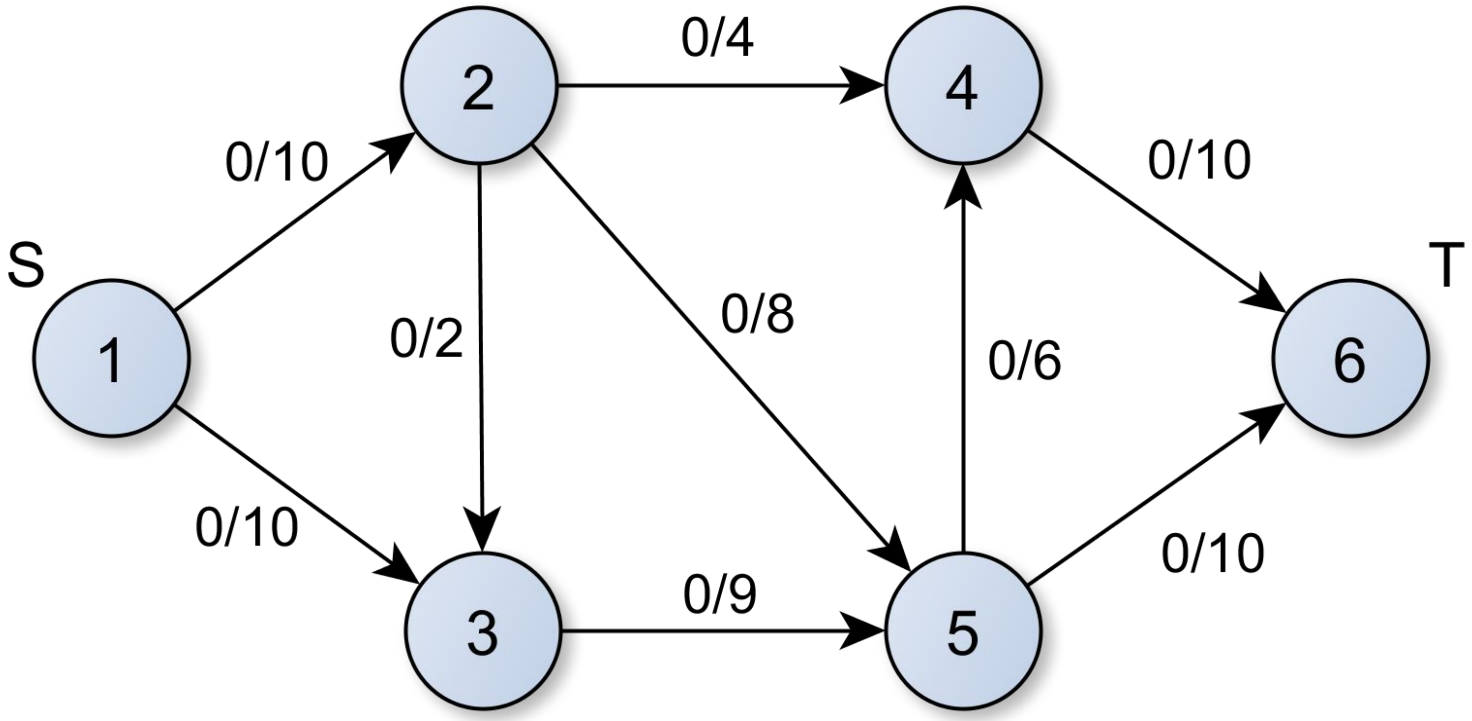
# Maximum Flow

Maximum flow algorithms route the maximum amount of one commodity from one source S to one sink T.

This functionality was added to the library, in addition to some example applications:

- Edge disjoint paths
- Multiple source/sink flow
- Maximum cardinality matching

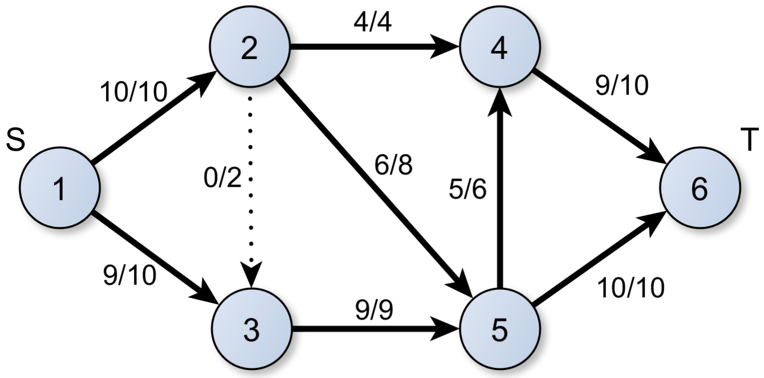
# Maximum Flow



# Maximum Flow

```
CREATE TABLE flow_example (
    id SERIAL,
    source INTEGER,
    target INTEGER,
    capacity INTEGER
);


INSERT INTO flow_example (source, target, capacity) VALUES
 (1, 2,10),
 (1, 3, 10);
 (2, 3, 2),
 (2, 4, 4);
 (2, 5, 8);
 (3, 5, 9);
 (4, 6, 10);
 (5, 4, 6);
 (5, 6, 10);
```

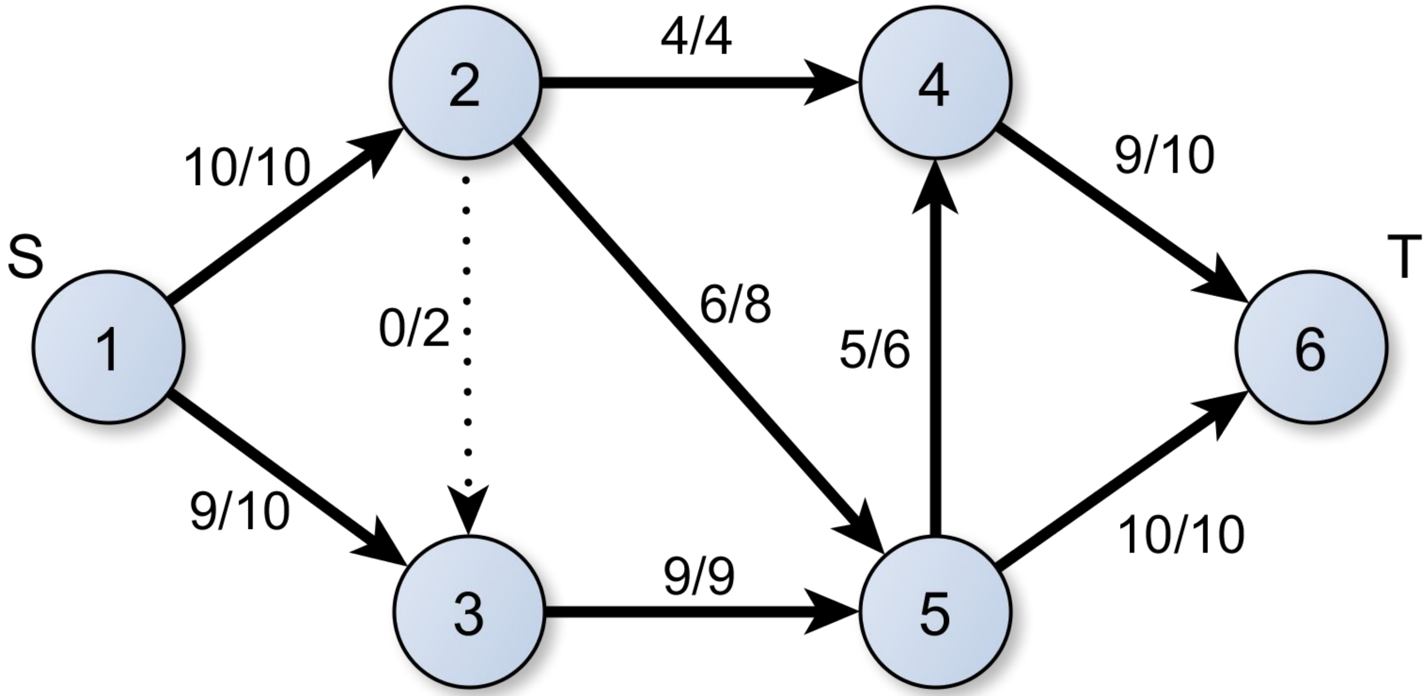


# THE QUERY

```
SELECT * FROM pgr_maxFlowEdmondsKarp(
  'SELECT id,
     source,
     target,
     capacity
    FROM flow_example',
    1, 6
);
```

# The results

```
 seq | edge_id | source | target | flow | residual_capacity
-----+---------+--------+--------+------+-------------------
   1 |       1 |      1 |      2 |   10 |                 0
   2 |       2 |      1 |      3 |    9 |                 1
   3 |       4 |      2 |      4 |    4 |                 0
   4 |       5 |      2 |      5 |    6 |                 2
   5 |       6 |      3 |      5 |    9 |                 0
   6 |       7 |      4 |      6 |    9 |                 1
   7 |       8 |      5 |      4 |    5 |                 1
   8 |       9 |      5 |      6 |   10 |                 0
(8 rows)
```


S
1
2
3
4
5
6
T
10/10
9/10
4/4
0/2
6/8
9/9
5/6
9/10
10/10

2015 GSoC Student
Thank you Sarthak!

# *osm2pgrouting*

Georepublic

2.4

# MAR-2017

## PLAN

- *pgr_astar*
    - *One to Many*
    - *Many to One*
    - *Many to Many*

Fork me on GitHub

github
SOCIAL CODING

https://github.com/pgRouting

Georepublic

Thank you!

Georepublic

# More Information

| | |
|---|---|
| Website: | pgrouting.org |
| Documentation: | docs.pgrouting.org |
| Workshop: | workshop.pgrouting.org |
| Support: | pgrouting.org/support.html |

**… or talk to me during FOSS4G 2017 ASIA:**

➔ Vicky vicky@georepublic.de