ΜΟΝΑΔΕΣ ΑΡΙΣΤΕΙΑΣ
ΑΝΟΙΧΤΟΥ ΛΟΓΙΣΜΙΚΟΥ

ΜΟΝΑΔΕΣ ΑΡΙΣΤΕΙΑΣ ΑΝΟΙΧΤΟΥ ΛΟΓΙΣΜΙΚΟΥ

Συστήματα γεωγραφικών πληροφοριών

1ος Κύκλος Εκπαίδευσης

5ο σεμινάριο

27 Ιουνίου 2014

ΠΣΥ IMIS

Δρομολόγηση

Η δρομολόγηση (routing) είναι η διαδικασία εύρεσης των «καλύτερων» μονοπατιών σε δίκτυα μεταξύ σημείων αφετηρίας και σημείων προορισμού.

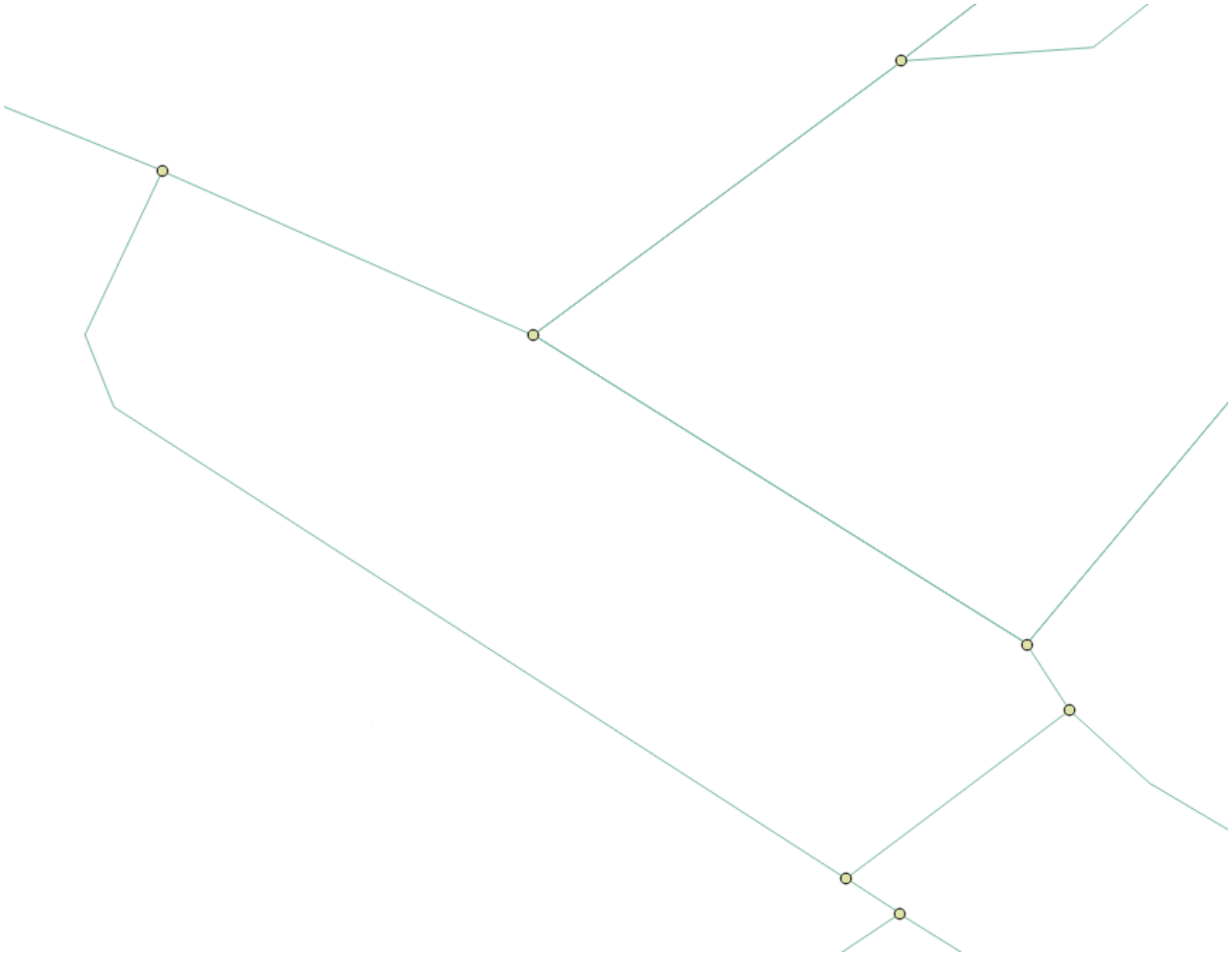
Χρησιμοποιείται τόσο σε τηλεπικοινωνιακά δίκτυα (τηλεφωνικά, δεδομένων) όσο και σε δίκτυα μεταφορών.

Δρομολόγηση

Η δρομολόγηση εκτελείται πάνω σε ένα δίκτυο (π.χ. οδικό) που αποτελείται από:

- **κόμβους** (π.χ. σημεία διασταύρωσης δρόμων) και
- **ακμές** (π.χ. μεμονωμένα τμήματα του οδικού δικτύου όπου τα άκρα τους είναι κόμβοι όπου δεν υπάρχει η επιλογή αλλαγής κατεύθυνσης μεταξύ αυτών)

Δρομολόγηση



Αλγόριθμοι Δρομολόγησης

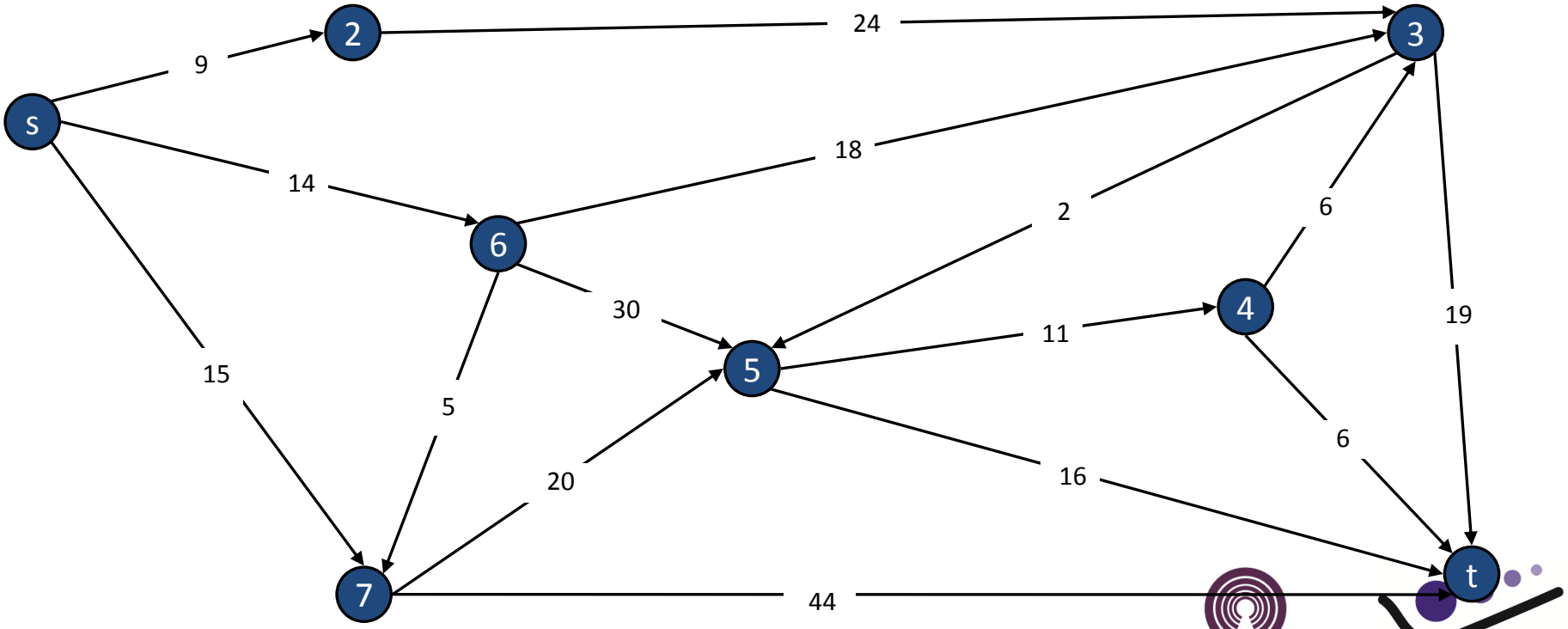
Για τη δρομολόγηση χρησιμοποιούνται αλγόριθμοι δρομολόγησης όπως οι:

- Dijkstra
- Bellman-Ford
- Floyd-Warshall
- A-star
- Shooting Star

Οι αλγόριθμοι αυτοί θεωρούν ένα δίκτυο από κόμβους (ή από ακμές) στο οποίο κάποιος μπορεί να μεταβεί από τον έναν στον αλλά αρκεί να έχουν μια κοινή ακμή (ή κοινό κόμβο)

Στις αποφάσεις επιλογής μονοπατιού σημασία έχουν «**βάρη**» που αποδίδονται σε ακμές ή κόμβους. Τα βάρη αντιπροσωπεύουν το κόστος διάσχισης μιας πλευράς ή κόμβου. Στα δίκτυα μεταφορών βάρος μπορεί να αποτελεί π.χ. το μήκος μιας ακμής, ο χρόνος διάσχισης που απαιτείται, το χρηματικό αντίτιμο για τη διάσχισή της κλπ.

Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

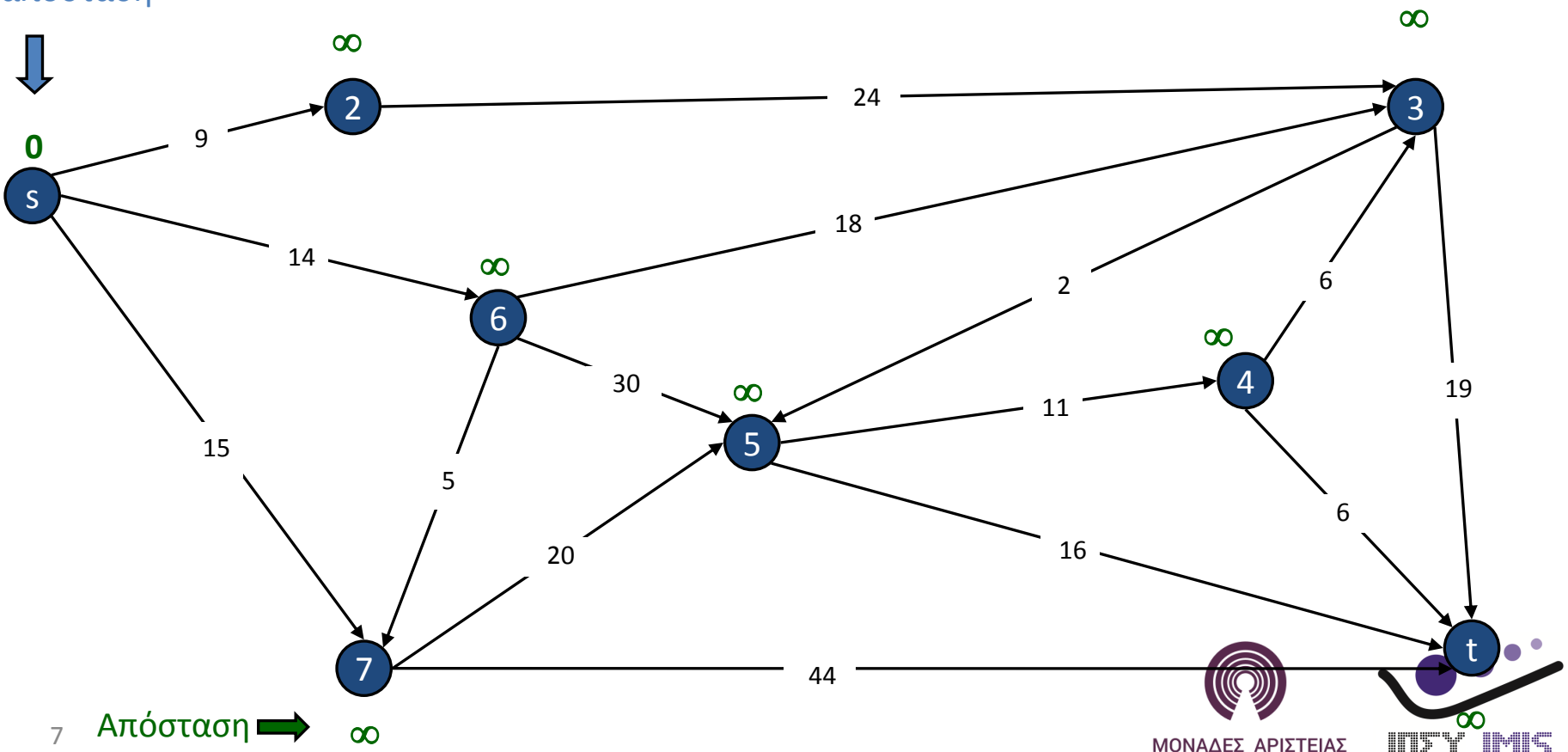


Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { }

PQ = { s, 2, 3, 4, 5, 6, 7, t }

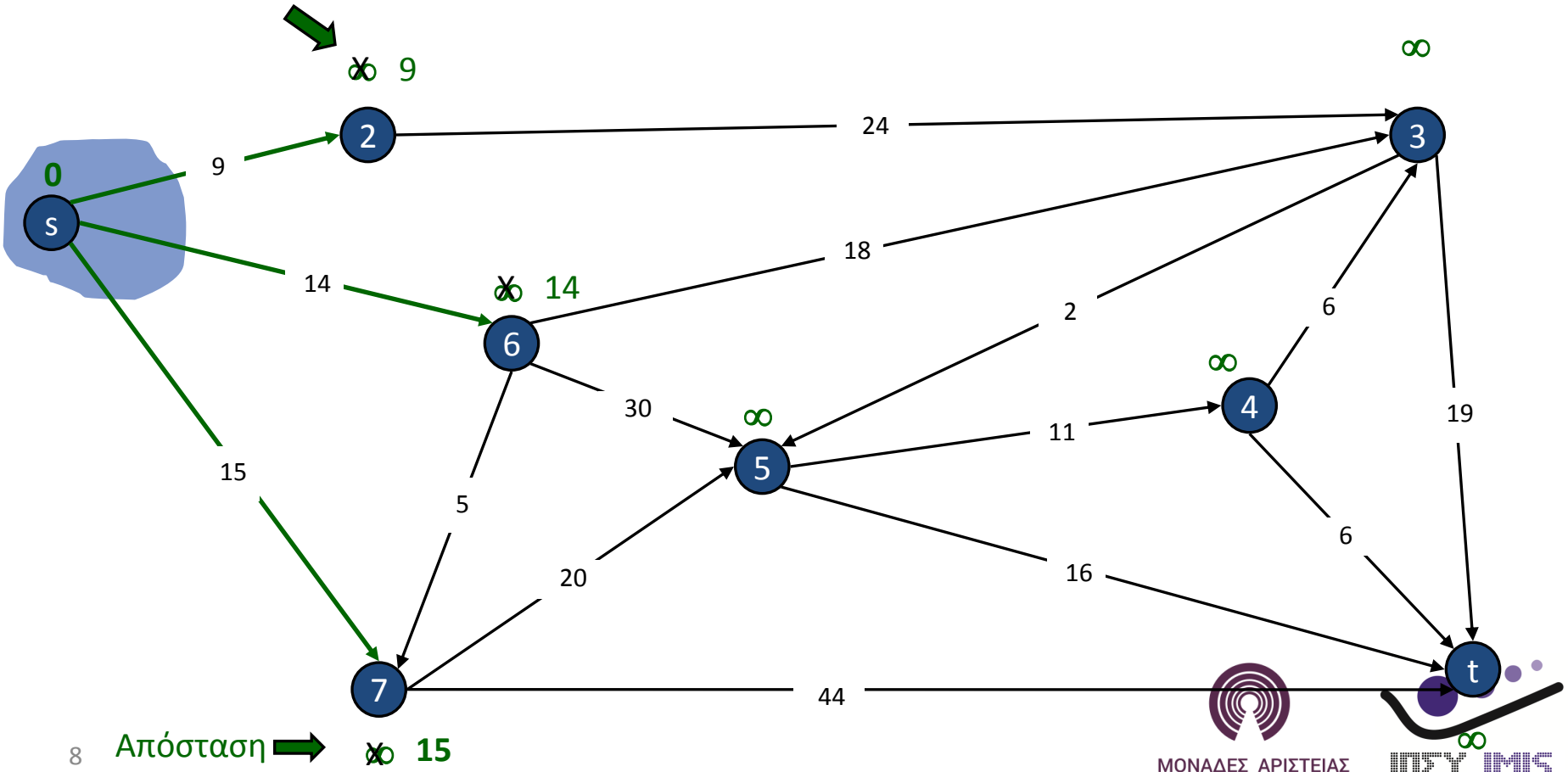
Ελάχιστη απόσταση



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = {s }

PQ = { 2, 3, 4, 5, 6, 7, t }

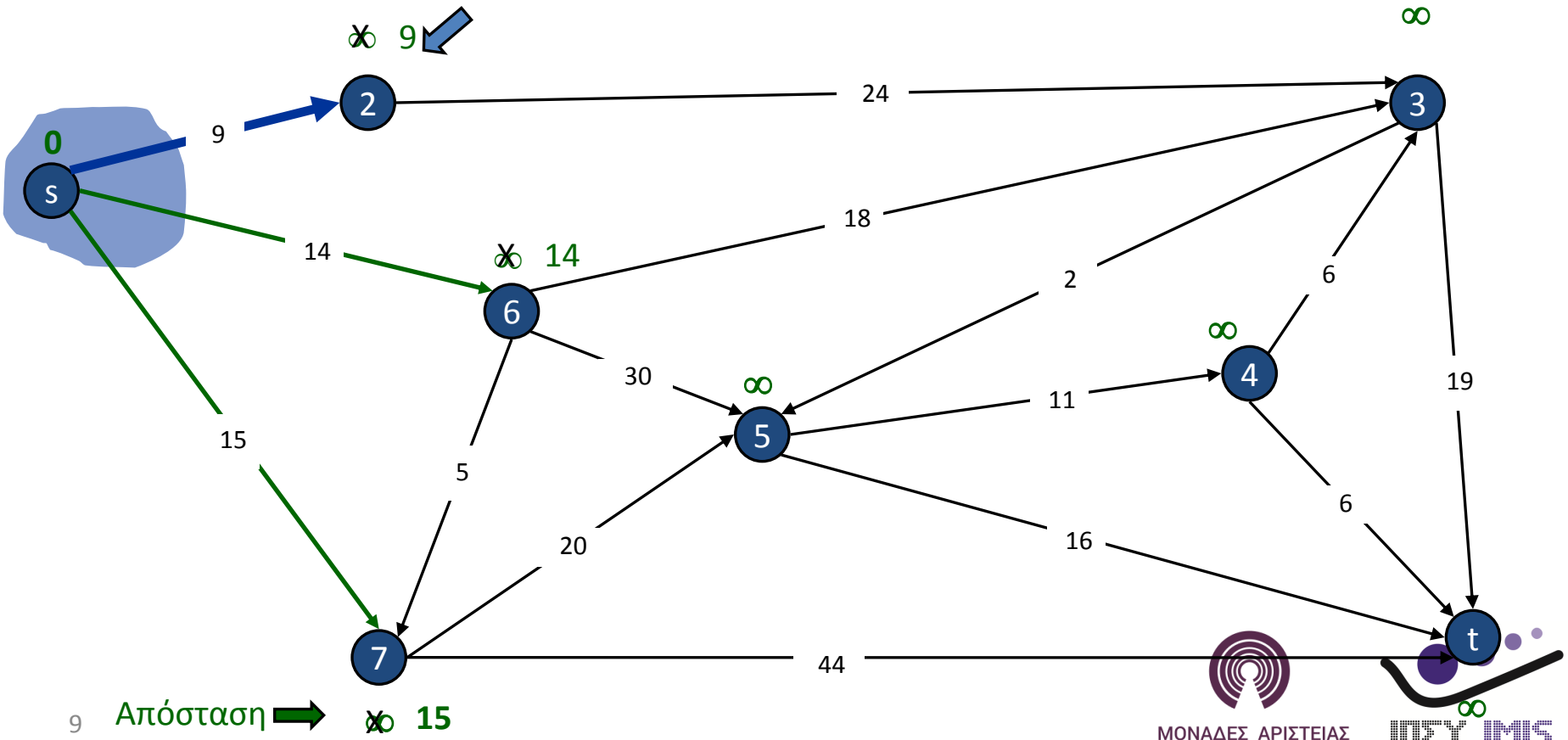


Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { }

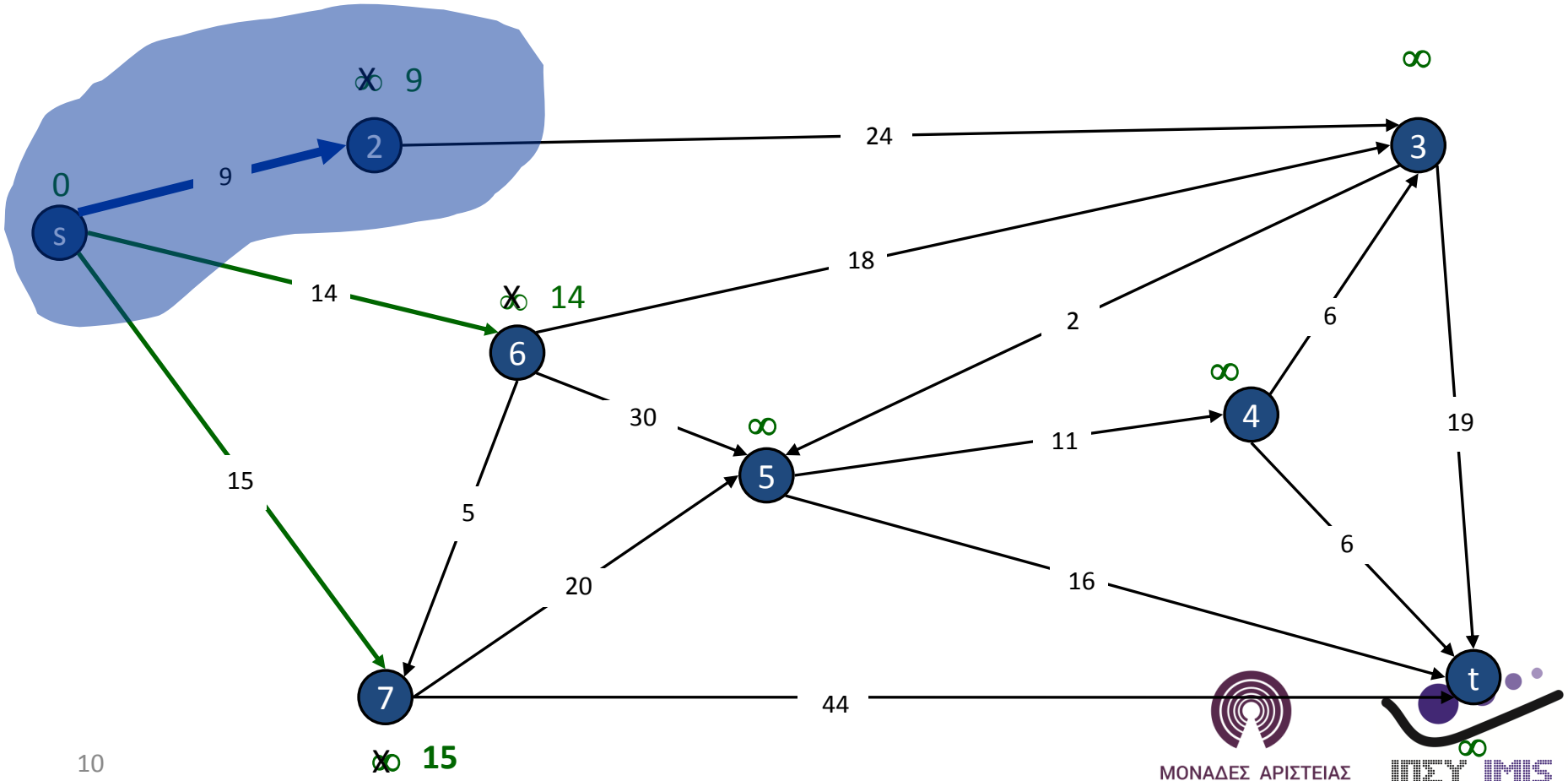
PQ = { s, 2, 3, 4, 5, 6, 7, t }

Ελάχιστη απόσταση



S = { s, 2 }

PQ = { 3, 4, 5, 6, 7, t }

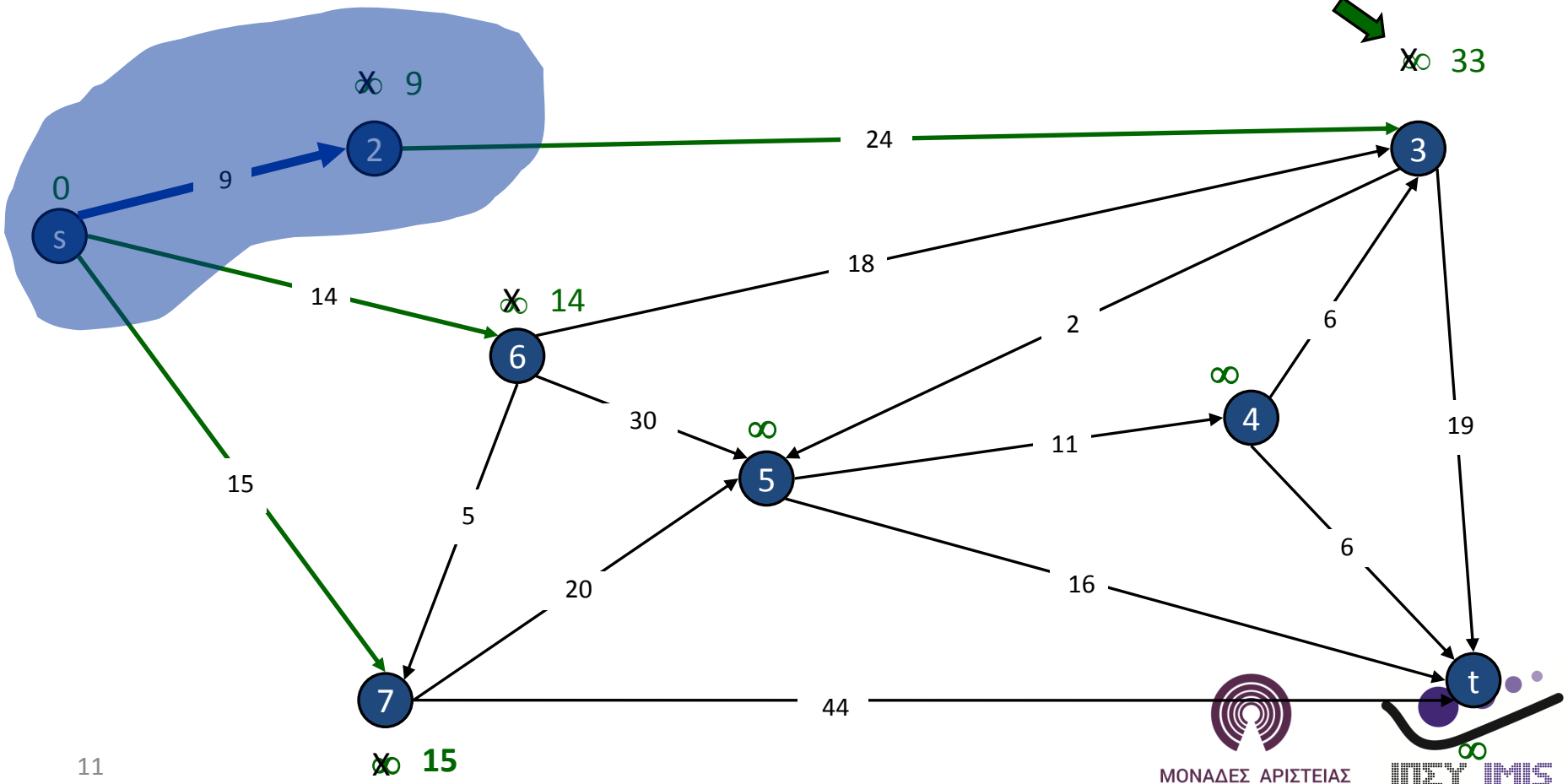


Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2 }

PQ = { 3, 4, 5, 6, 7, t }

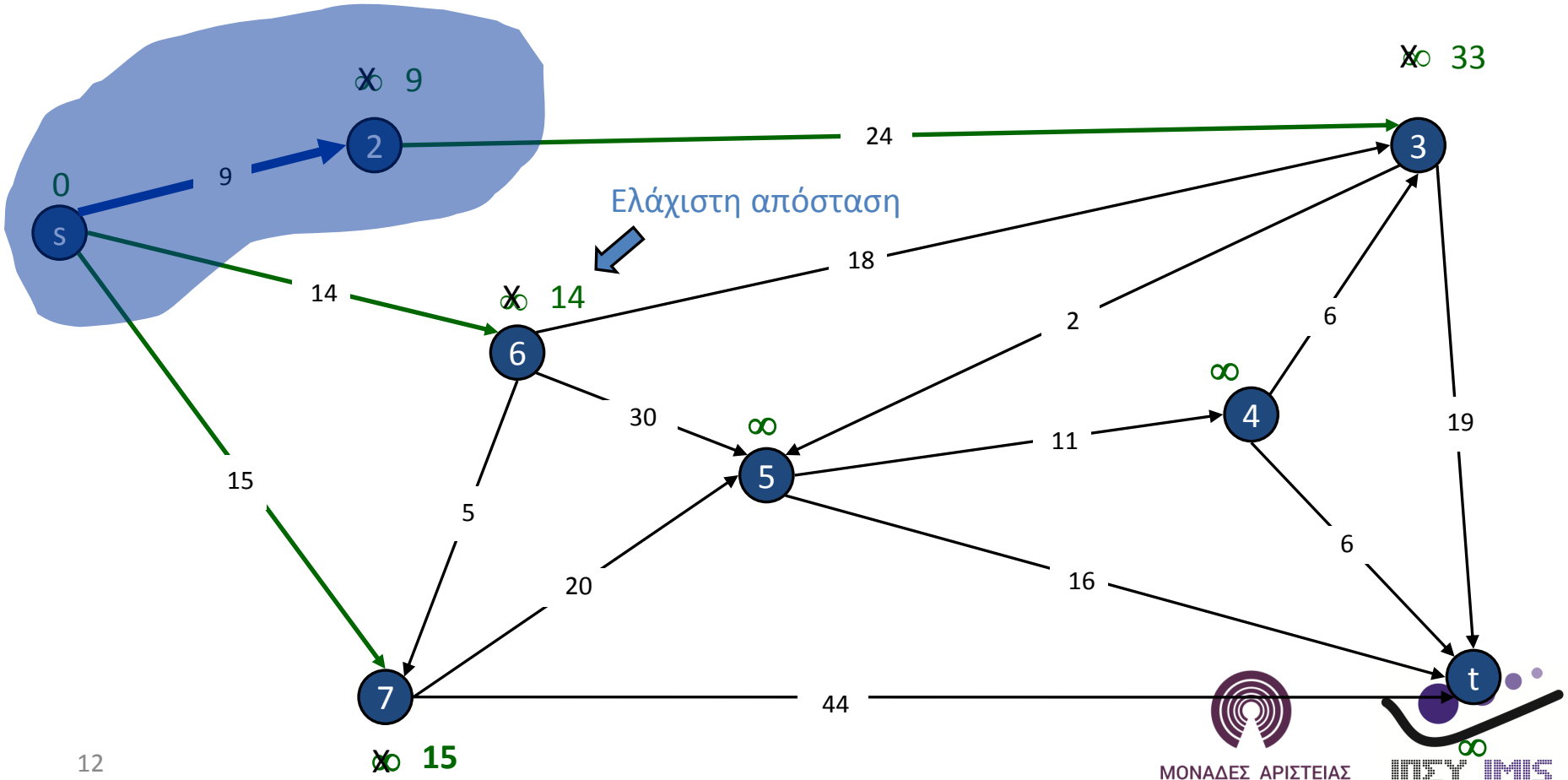
Ενημέρωση απόστασης



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2 }

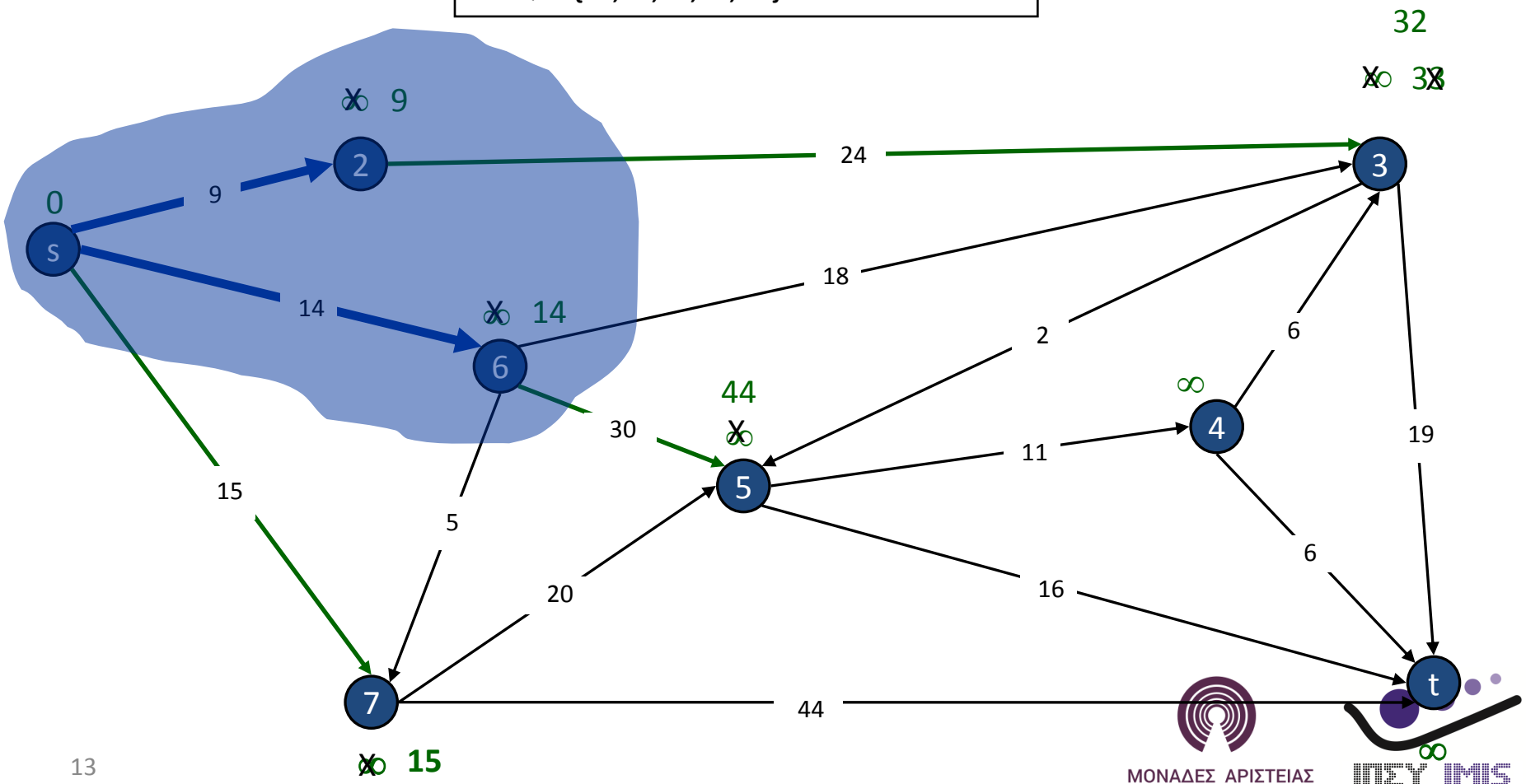
PQ = { 3, 4, 5, 6, 7, t }



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 6 }

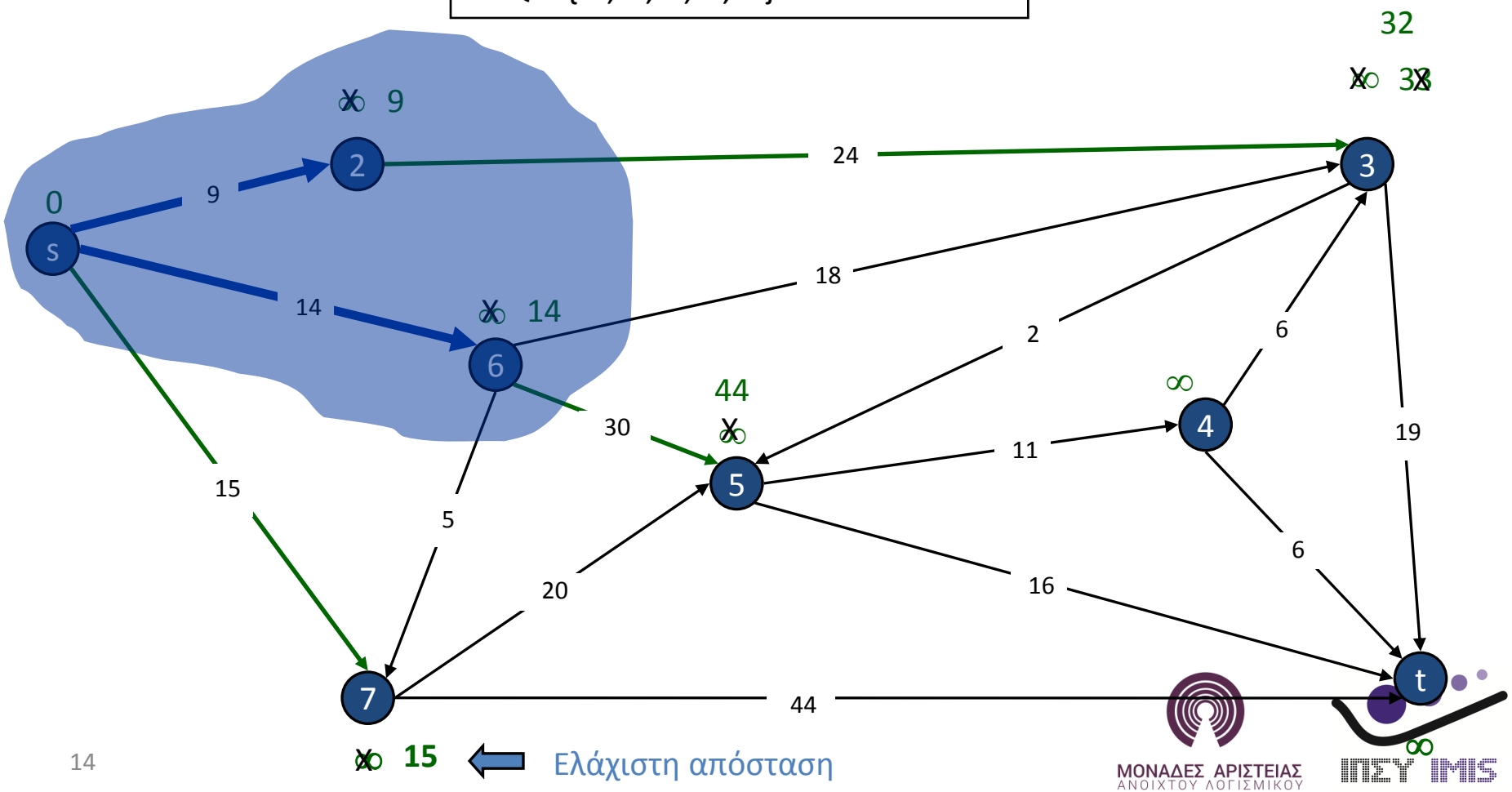
PQ = { 3, 4, 5, 7, t }



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 6 }

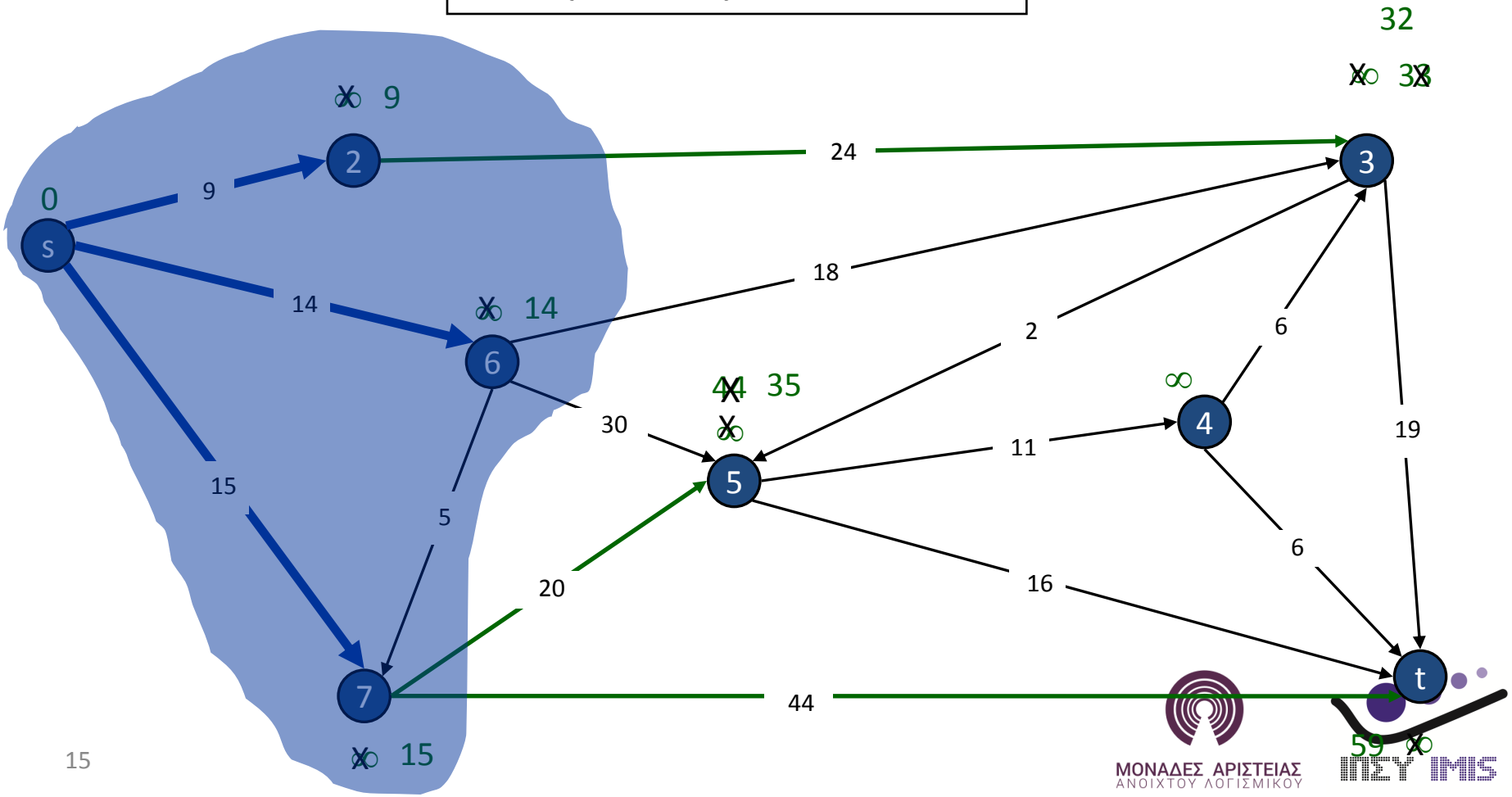
PQ = { 3, 4, 5, 7, t }



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 6, 7 }

PQ = { 3, 4, 5, t }

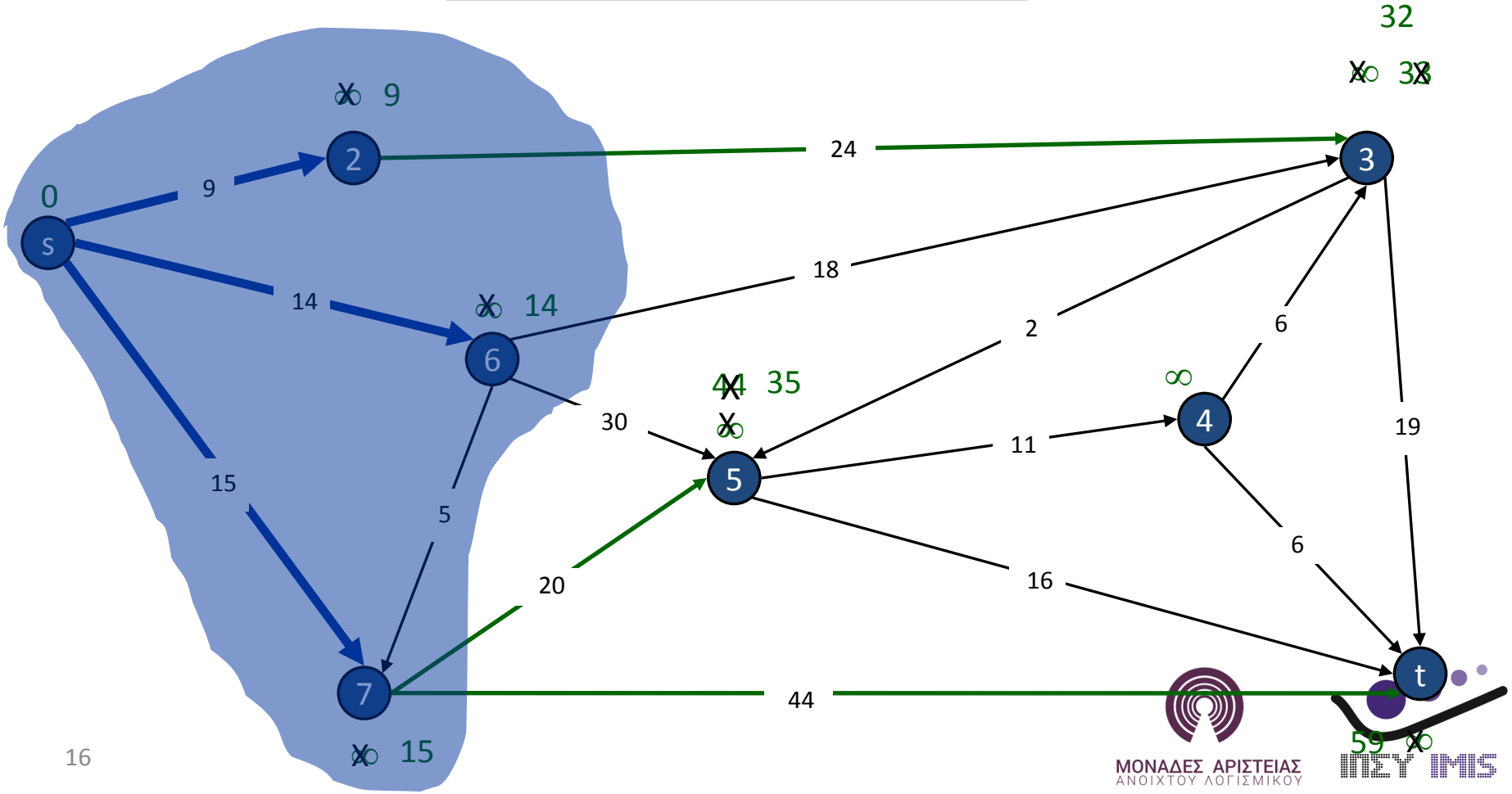


Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 6, 7 }

PQ = { 3, 4, 5, t }

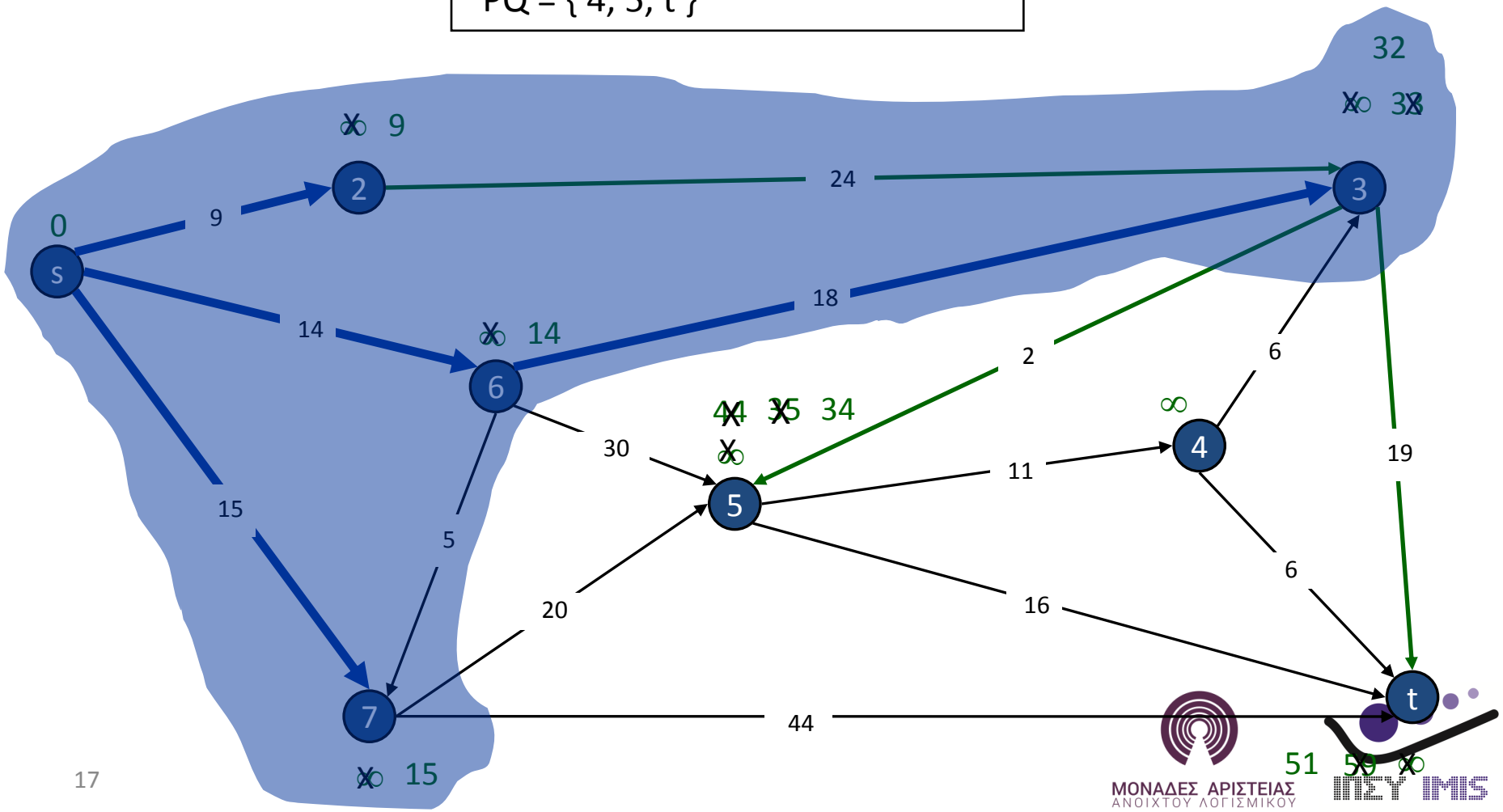
Ελάχιστη απόσταση



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 3, 6, 7 }

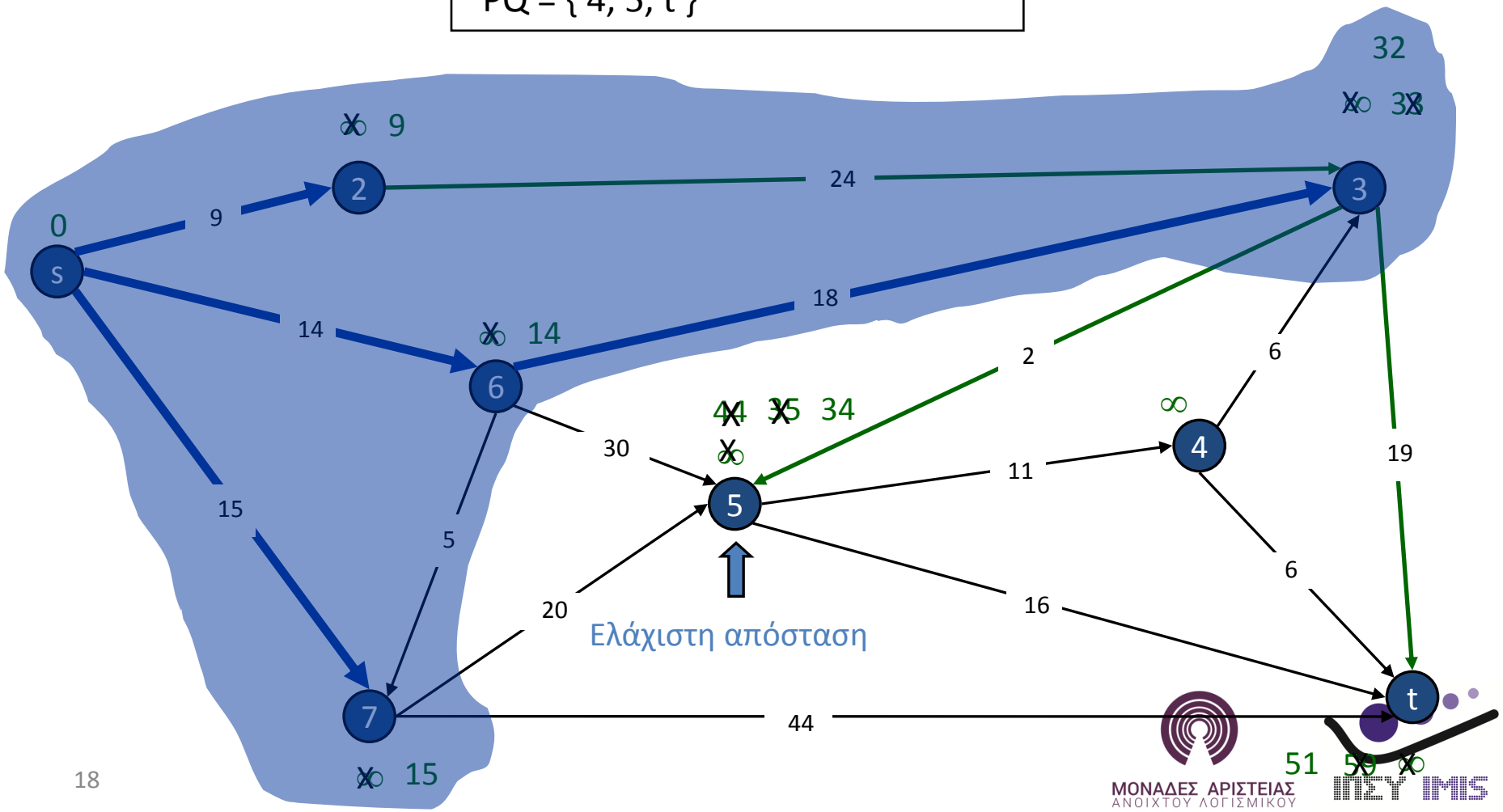
PQ = { 4, 5, t }



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 3, 6, 7 }

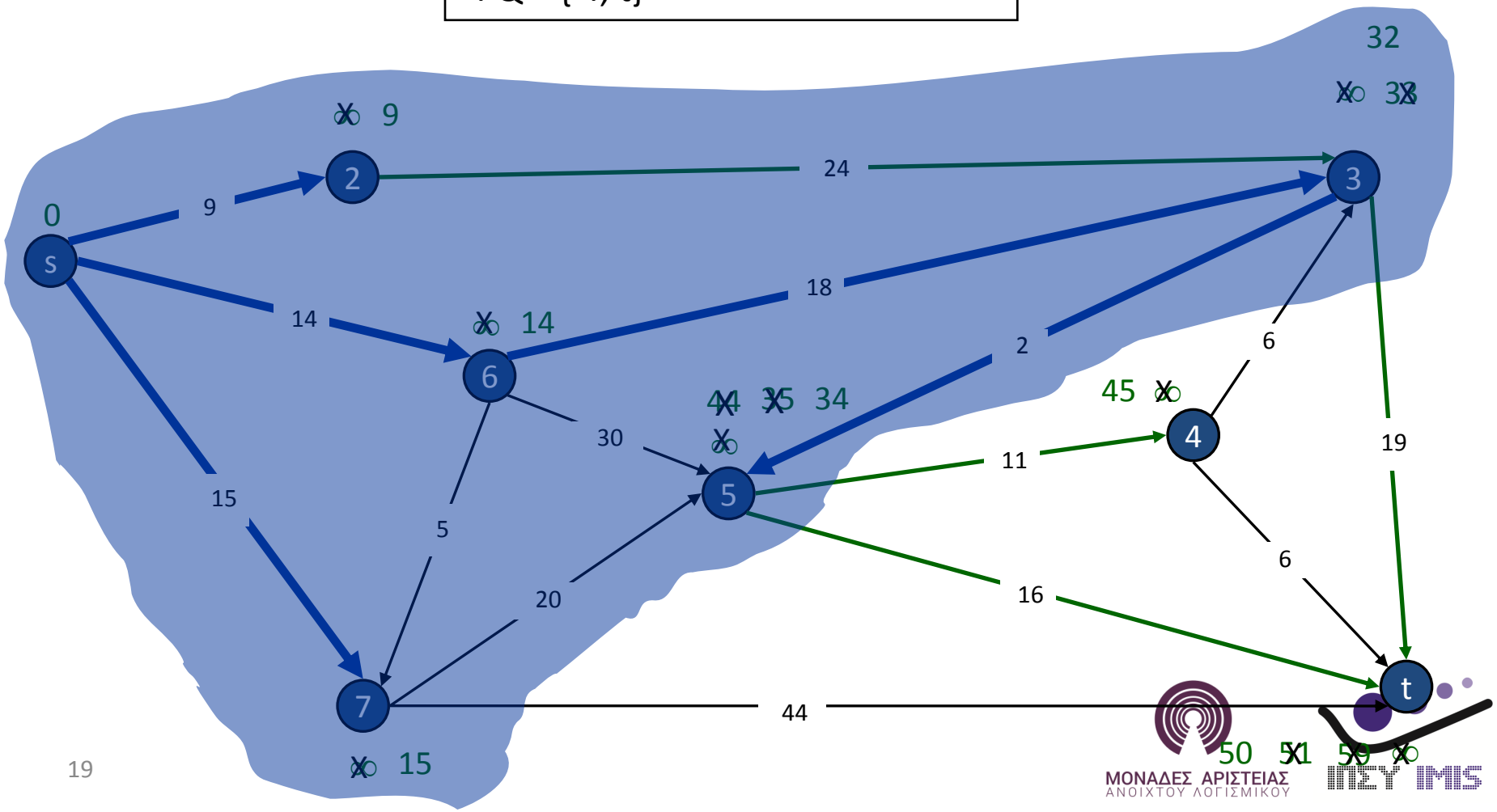
PQ = { 4, 5, t }



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 3, 5, 6, 7 }

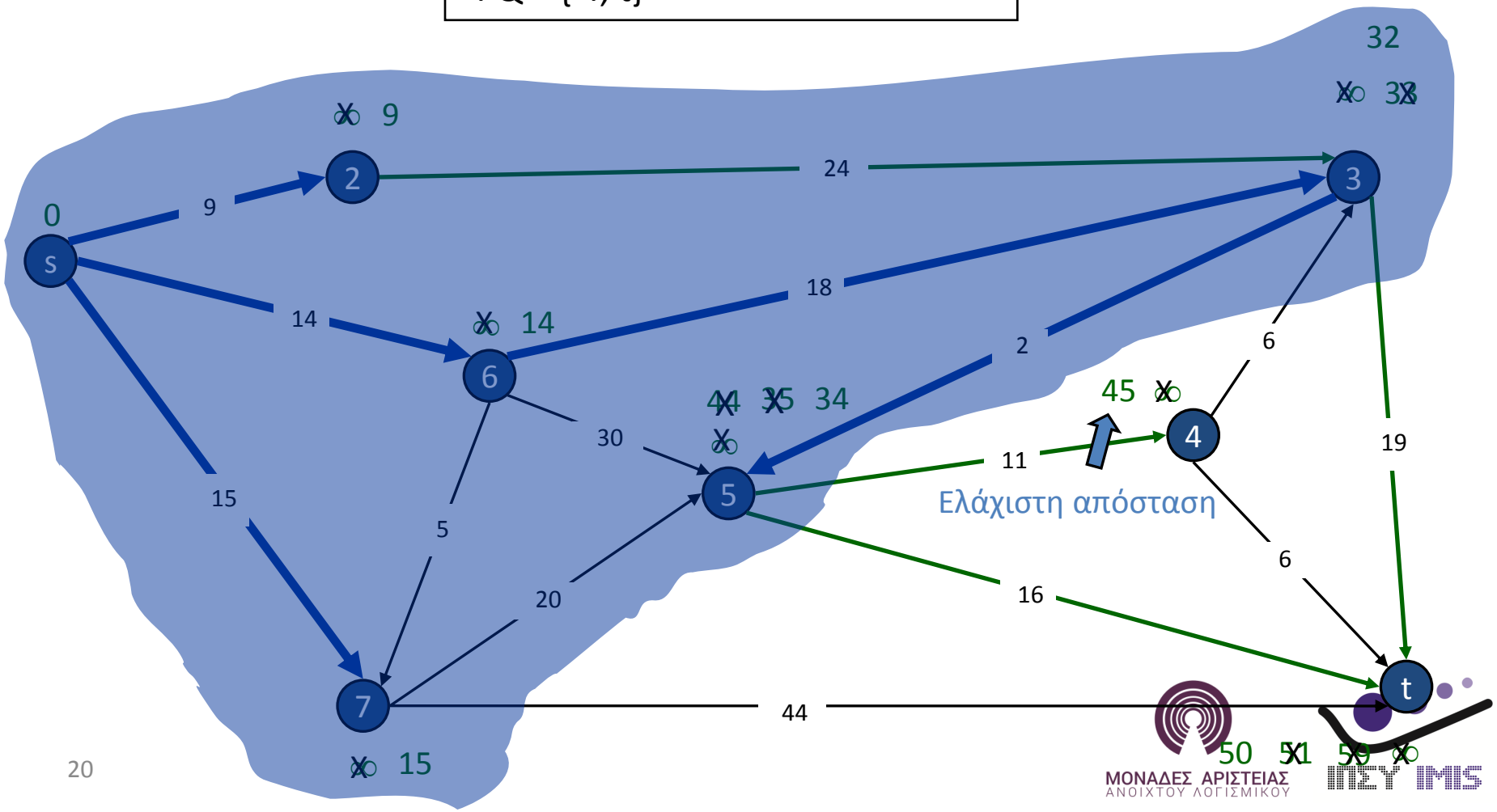
PQ = { 4, t}



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 3, 5, 6, 7 }

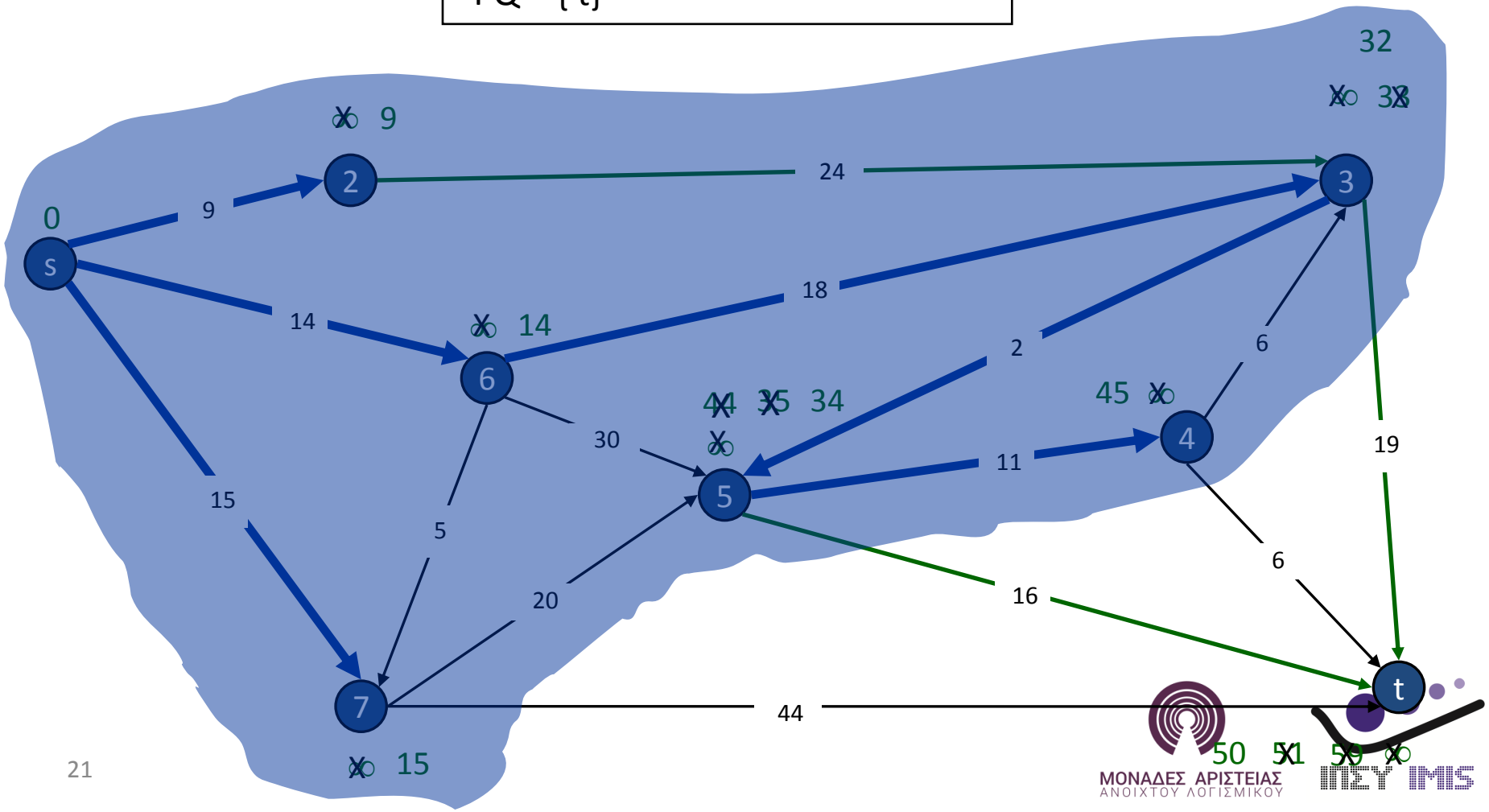
PQ = { 4, t}



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 3, 5, 6, 7, 4}

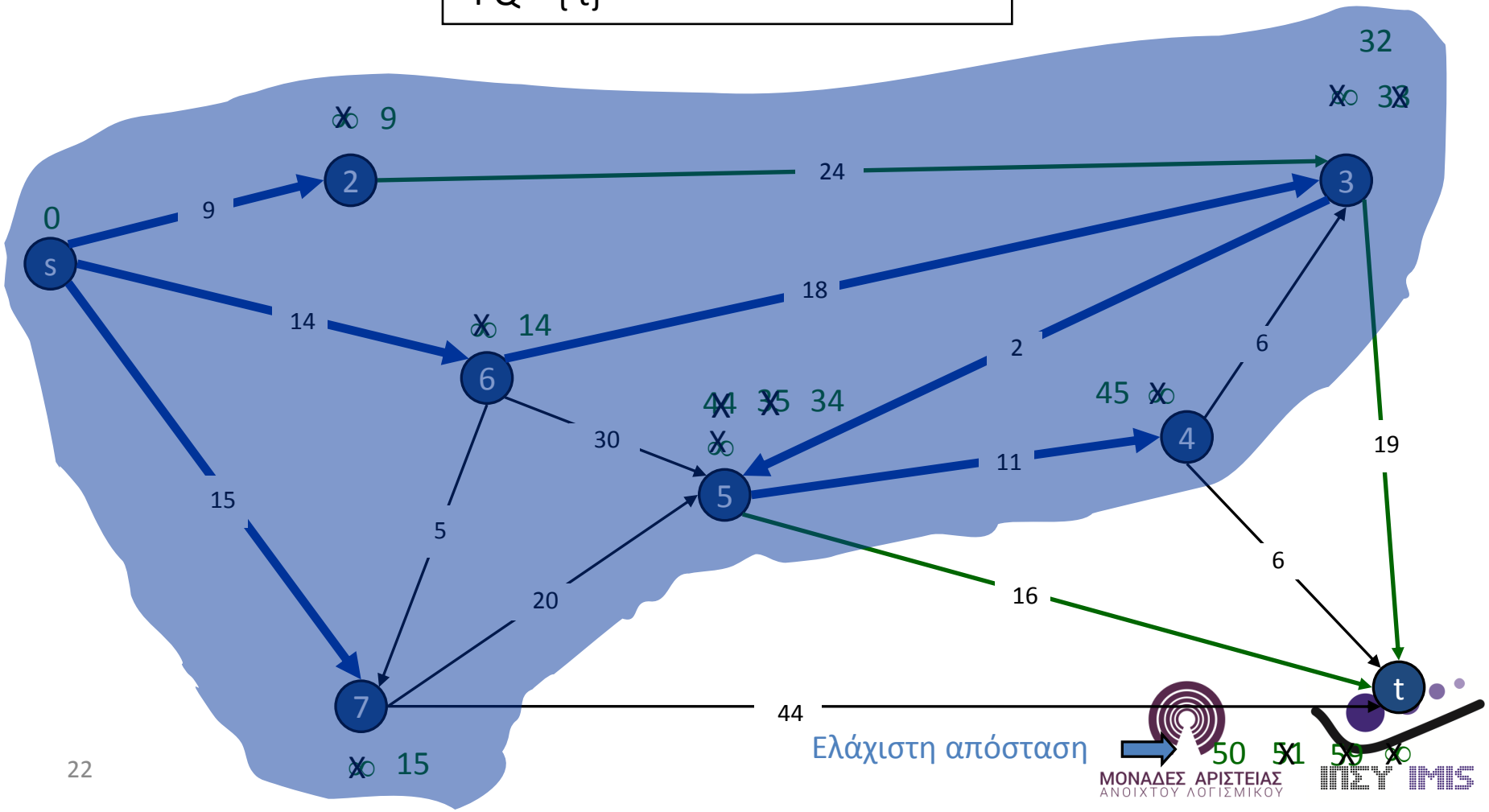
PQ = { t}



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 3, 5, 6, 7, 4}

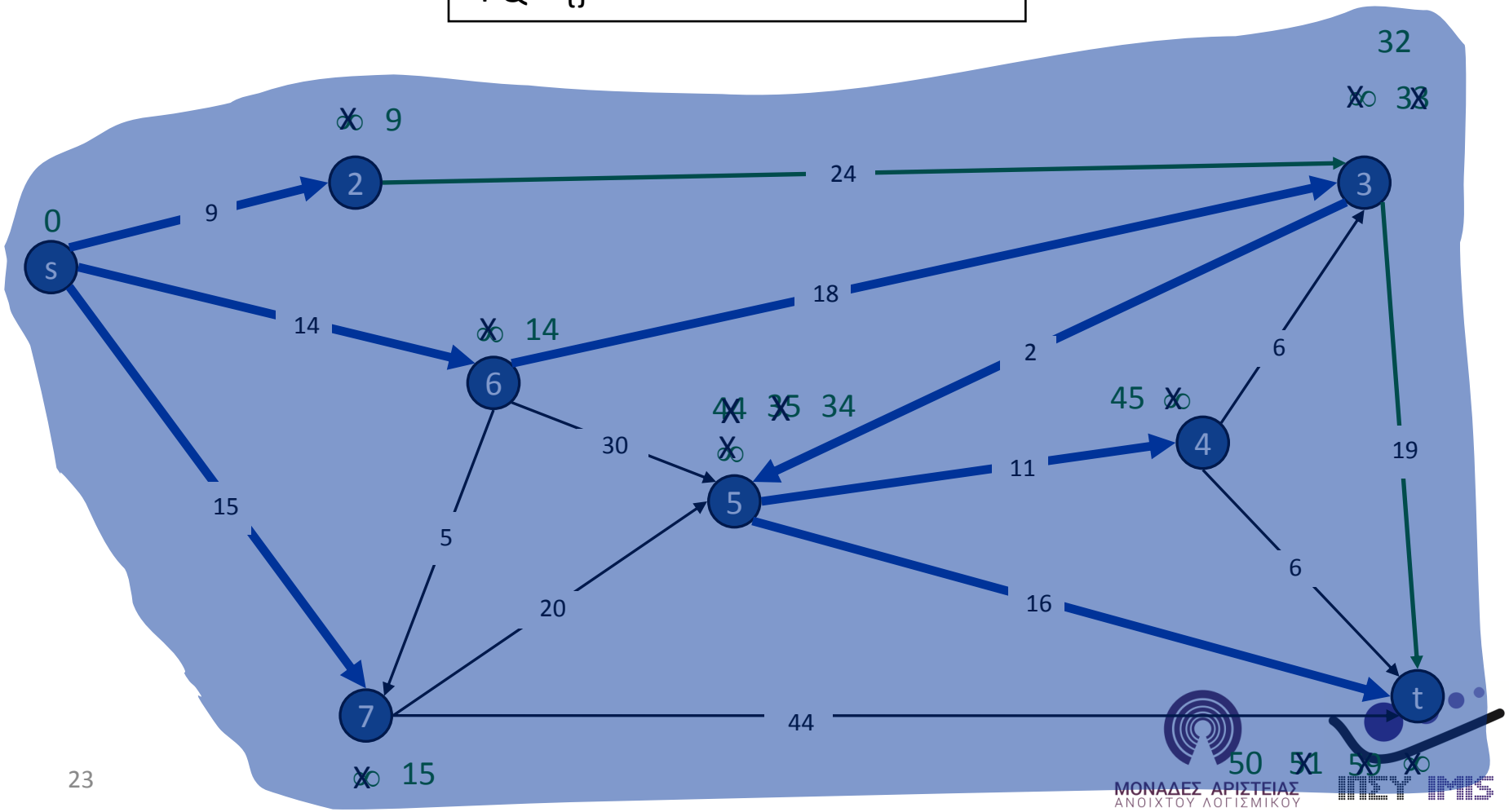
PQ = { t}



Εύρεση του συντομότερου μονοπατιού με τον αλγόριθμο Dijkstra

S = { s, 2, 3, 5, 6, 7, 4, t}

PQ = {}



pgRouting

Το **pgRouting** είναι λογισμικό ΕΛΛΑΚ το επεκτείνει την γεωχωρική βάση δεδομένων PostGIS/PostgreSQL δίνοντας τη δυνατότητα εκτέλεσης ερωτημάτων δρομολόγησης.

Το μεταφορικό δίκτυο αποθηκεύεται σε βάση PostgreSQL. Αυτό δίνει τα παρακάτω πλεονεκτήματα:

- Τα δεδομένα μπορούν να τροποποιηθούν από πολλά λογισμικά GIS όπως το QGIS, και το uDig. Οι τροποποιήσει μπορεί να γίνονται είτε από PCs είτε από φορητές συσκευές.
- Οι αλλαγές στα δεδομένα αποτυπώνονται απευθείας χωρίς να απαιτείται προεπεξεργασία.
- Τα βάρη μπορούν να υπολογίζονται δυναμικά με SQL ερωτήματα συνυπολογίζοντας τιμές από διαφορετικές στήλες ή/και πίνακες.

Το pgRouting είναι λογισμικό ΕΛ/ΛΑΚ και διατίθεται με την άδεια GPLv2

Εισαγωγή δεδομένων και επεξεργασία

Σημαντικές πηγές δεδομένων: OpenStreetMap

Για τη χρήση OSM δεδομένων χρησιμοποιείται το λογισμικό **OSMOSIS** το οποίο μπορεί να προεπεξεργαστεί τα OSM δεδομένα (π.χ. περικοπή δεδομένων)

Τα OSM δεδομένα εισάγονται με το **osm2pgrouting**. Τα είδη του οδικού δικτύου που θα επιλεγούν καθορίζονται από το αρχείο **config.xml**.

Υποστηριζόμενοι Αλγόριθμοι

Dijkstra

Υπολογίζει το συντομότερο μονοπάτι από έναν κόμβο αφετηρία προς όλους τους άλλους (και προς τον κόμβο στόχο).

```
CREATE OR REPLACE FUNCTION shortest_path(
                                        sql text,
                                        source_id integer,
                                        target_id integer,
                                        directed boolean,
                                        has_reverse_cost boolean)
        RETURNS SETOF path_result
```

Το ερώτημα SQL πρέπει να επιστρέψει τα παρακάτω:

```
id, source, target, cost
```

Υποστηριζόμενοι Αλγόριθμοι

Dijkstra

- **id**: an int4 identifier of the edge
- **source**: an int4 identifier of the source vertex
- **target**: an int4 identifier of the target vertex
- **cost**: an float8 value, of the edge traversal cost. (a negative cost will prevent the edge from being inserted in the graph).
- **reverse_cost** (optional): the cost for the reverse traversal of the edge. This is only used when the directed and has_reverse_cost parameters are true (see the above remark about negative costs).

Παράδειγμα

```
SELECT vertex_id, edge_id, cost, the_geom FROM
shortest_path('
    SELECT gid AS id,
           source,
           target,
           to_cost AS cost
    FROM ways',
75351,
31356,
false,
false) AS foo, ways WHERE edge_id=gid;
```

Υποστηριζόμενοι Αλγόριθμοι

A Star

Υπολογίζει το συντομότερο μονοπάτι από έναν κόμβο αφετηρία προς όλους τον κόμβο προορισμό. Σε κάθε αναζήτηση επιδιώκει να μειώσει τόσο το συνολικό μήκος του μονοπατιού που έχει ακολουθηθεί μέχρι στιγμής όσο και την απόσταση του ενδιάμεσου κόμβου από τον προορισμό.

```
CREATE OR REPLACE FUNCTION shortest_path_astar(
                                                    sql text,
                                                    source_id integer,
                                                    target_id integer,
                                                    directed boolean,
                                                    has_reverse_cost boolean)
        RETURNS SETOF path_result
```

Το ερώτημα SQL πρέπει να επιστρέψει τα παρακάτω:

```
id, source, target, cost, x1, y1, x2, y2
```

ΜΟΝΑΔΕΣ ΑΡΙΣΤΕΙΑΣ
ΑΝΟΙΧΤΟΥ ΛΟΓΙΣΜΙΚΟΥ

ΙΜΣΥ IMIS

Υποστηριζόμενοι Αλγόριθμοι

A Star

- **id**: an int4 identifier of the edge
- **source**: an int4 identifier of the source vertex
- **target**: an int4 identifier of the target vertex
- **cost**: an float8 value, of the edge traversal cost. (a negative cost will prevent the edge from being inserted in the graph).
- **x1**: x coordinate of the start point of the edge
- **y1**: y coordinate of the start point of the edge
- **x2**: y coordinate of the end point of the edge
- **y2**: y coordinate of the end point of the edge
- **reverse_cost** (optional): the cost for the reverse traversal of the edge. This is only used when the directed and has_reverse_cost parameters are true (see the above remark about negative costs).

Υποστηριζόμενοι Αλγόριθμοι

Shooting Star

Υπολογίζει το συντομότερο μονοπάτι από έναν κόμβο αφετηρία προς όλους τον κόμβο προορισμό. Δρομολογεί από ακμή σε ακμή (αντί από κόμβο σε κόμβο) επιτρέποντας την υλοποίηση πολύπλοκων πολιτικών περιορισμών (π.χ. «απαγορεύεται η στροφή δεξιά»)

```
CREATE OR REPLACE FUNCTION shortest_path_shooting_star(
                                                sql text,
                                                source_id integer,
                                                target_id integer,
                                                directed boolean,
                                                has_reverse_cost boolean)
RETURNS SETOF path_result
```

Το ερώτημα SQL πρέπει να επιστρέψει τα παρακάτω:

```
id, source, target, cost, x1, y1, x2, y2, rule, to_cost
```

Υποστηριζόμενοι Αλγόριθμοι

Shooting Star

- **id**: an int4 identifier of the edge
- **source**: an int4 identifier of the source vertex
- **target**: an int4 identifier of the target vertex
- **cost**: double precision value of the edge traversal cost. (a negative cost will prevent the edge from being inserted in the graph).
- **reverse_cost** (optional): the cost for the reverse traversal of the edge. This is only used when the directed and has_reverse_cost parameters are true (see the above remark about negative costs).
- **directed**: true if the graph is directed
- **has_reverse_cost**: if true, the reverse_cost column of the SQL generated set of rows will be used for the cost of the traversal of the edge in the opposite direction.
- **x1**: double precision value of x coordinate for edge's start vertex
- **y1**: double precision value of y coordinate for edge's start vertex
- **x2**: double precision value of x coordinate for edge's end vertex
- **y2**: double precision value of y coordinate for edge's end vertex
- **rule**: a string with a comma separated list of edge ids which describes a rule for turning restriction (if you came along these edges, you can pass through the current one only with the cost stated in to_cost column)
- to_cost: a cost of restricted passage (can be very high in a case of turn restriction or comparable with an edge cost in a case of traffic light)