Συντομότερα Μονοπάτια

Π. Λουρίδας

Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας, Οικονομικό Πανεπιστήμιο Αθηνών
louridas@aueb.gr

Επισκόπηση

1. Παραγραφοποίηση
2. Συντομότερα Μονοπάτια
3. Ο Αλγόριθμος του Dijkstra

Επισκόπηση

Διαμόρφωση Παραγράφων

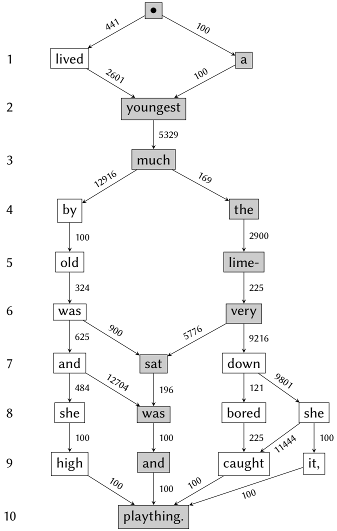
In olden times when wishing still helped one, there lived a king whose daughters were all beautiful, but the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forest and sat down by the side of the cool fountain, and when she was bored she took a golden ball, and threw it up on high and caught it, and this ball was her favorite plaything.

In olden times when wishing still helped one, there lived a king whose daughters were all beautiful, but the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forest and sat down by the side of the cool fountain, and when she was bored she took a golden ball, and threw it up on high and caught it, and this ball was her favorite plaything.

Παράδειγμα Παραγραφοποίησης

In olden times when wishing still helped one, there lived a king whose daughters were all beautiful, but the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forestand sat down by the side of the cool fountain, and when she was boredshe took a golden ball, and threw it up on high and caught it, and this ball was her favorite plaything.

Βέλτιστο Σπάσιμο Γραμμών



Παραδείγματα Γραμμών

was astonished whenever it shone in her face. Close by

was astonished whenever it shone in her face. Close by the

Εφαρμογή

- TEX (Knuth, Plass)
- LATEX
- Adobe Corporation

Επισκόπηση

Το Πρόβλημα των Συντομότερων Μονοπατιών

Ορισμός

Σε έναν γράφο $G = (V, E)$ και έναν κόμβο s του γράφου να βρεθούν τα συντομότερα μονοπάτια από τον κόμβο s σε κάθε άλλο κόμβο του γράφου.

Εφαρμογές

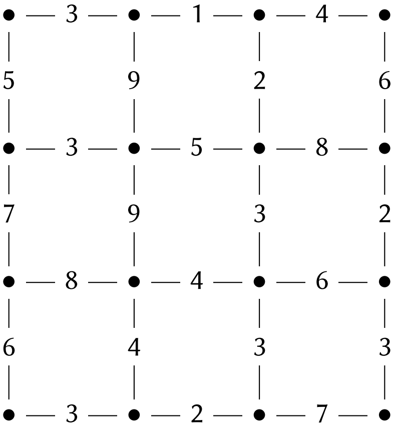
- Παραγραφοποίηση
- Πλοήγηση σε χάρτη
- Δρομολόγηση σε κίνηση
- Βελτιστοποίηση αεροπορικών ταξιδιών
- Δρομολόγηση σε τηλεπικοινωνιακά δίκτυα
- Δρομολόγηση robot
- ...

Επισκόπηση

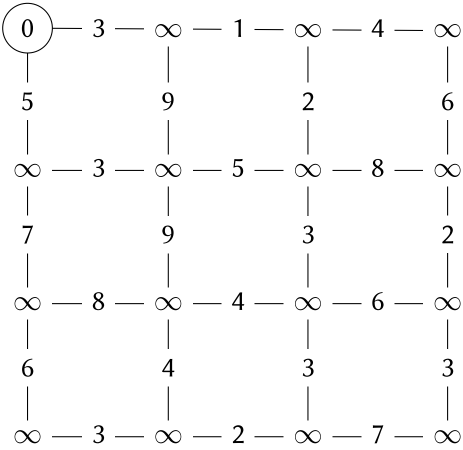
Ιδέα

- Επινοήθηκε από τον Ολλανδό Edsger Dijksra το 1956, δημοσιεύθηκε το 1959.
- Αρχικοποιούμε το γράφο με εκτιμήσεις για τα συντομότερα μονομάτια από τον κόμβο εκκίνησης. Ο κόμβος εκκίνησης έχει εκτίμηση 0, όλοι οι άλλοι ∞.
- Επιλέγουμε τον κόμβο με τη βέλτιστη εκτίμηση και ενημερώνουμε (χαλαρώνουμε) τις εκτιμήσεις για όλους τους κόμβους που συνδέονται με αυτόν.
- Επαλαμβάνουμε τη διαδικασία τόσες φορές όσοι είναι οι κόμβοι του γράφου.

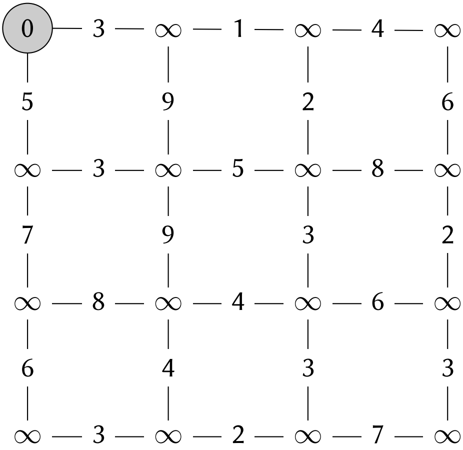
Ένα Πολεοδομικό Δίκτυο



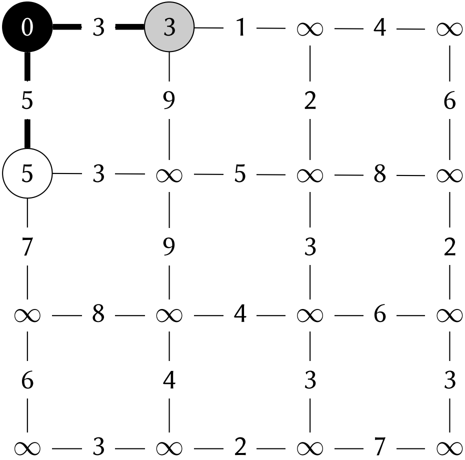
Dijkstra (1)



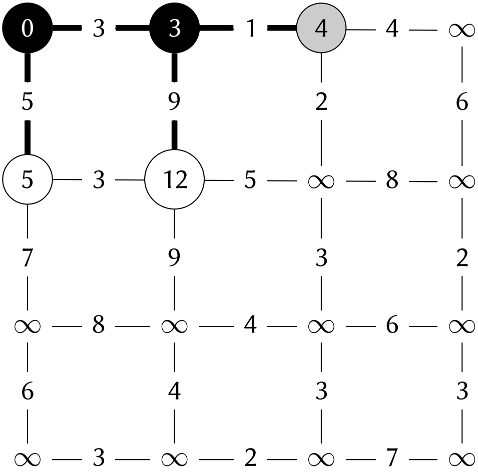
Dijkstra (2)



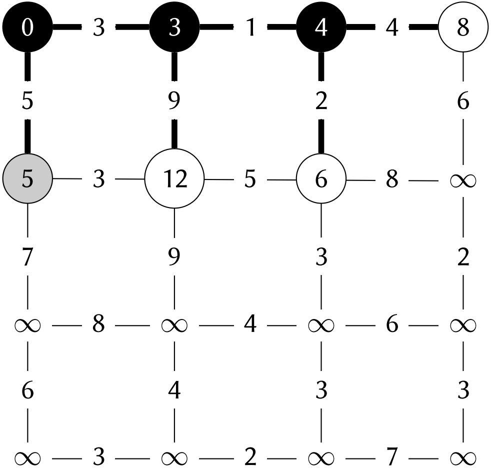
Dijkstra (3)



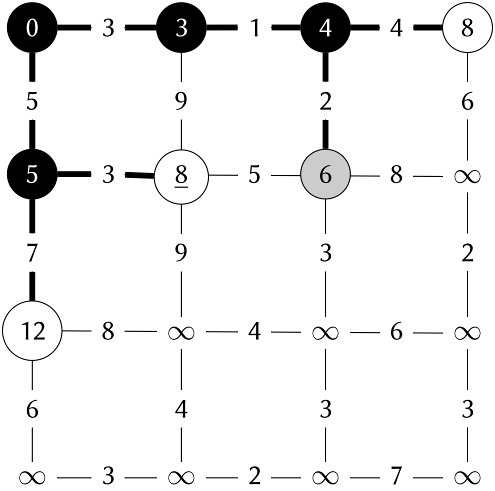
Dijkstra (4)



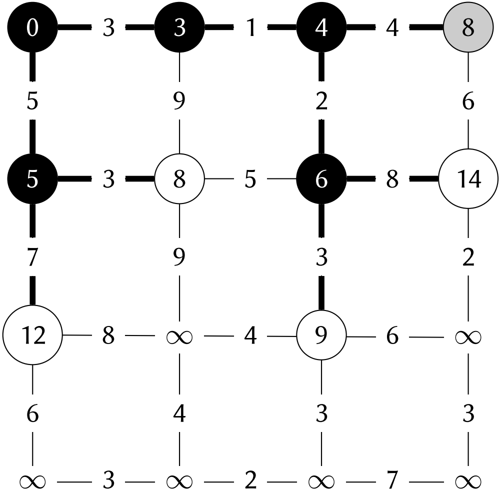
Dijkstra (5)



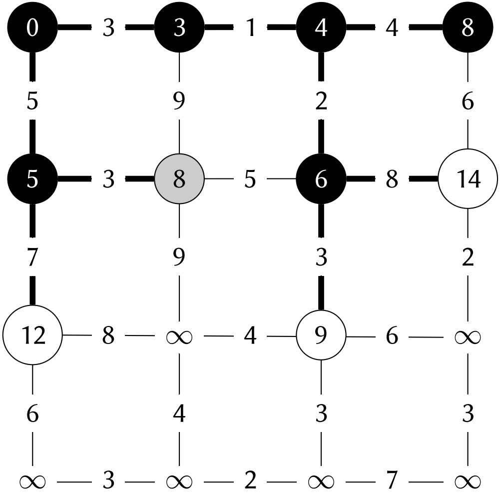
Dijkstra (6)



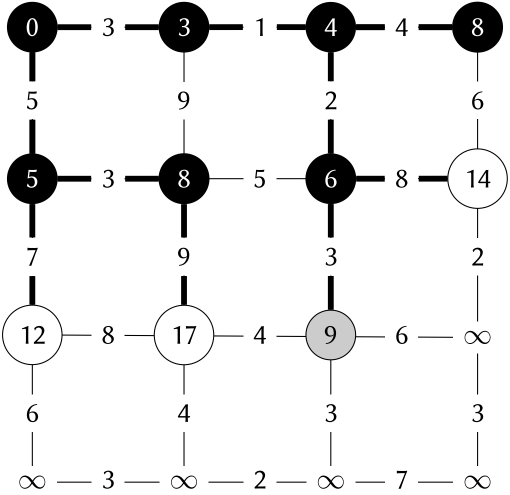
Dijkstra (7)



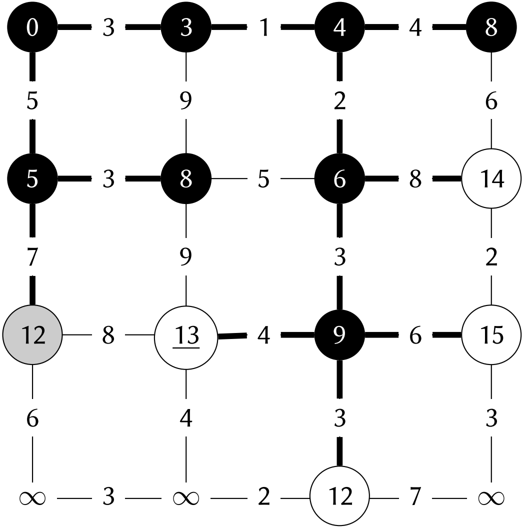
Dijkstra (8)



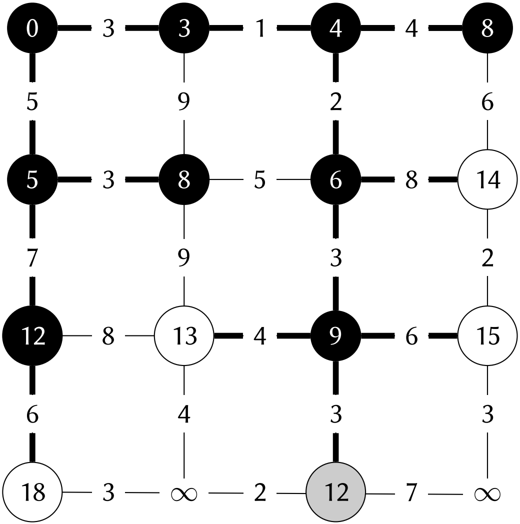
Dijkstra (9)



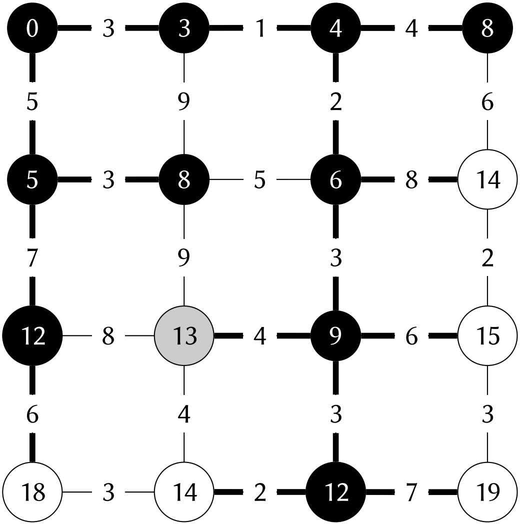
Dijkstra (10)



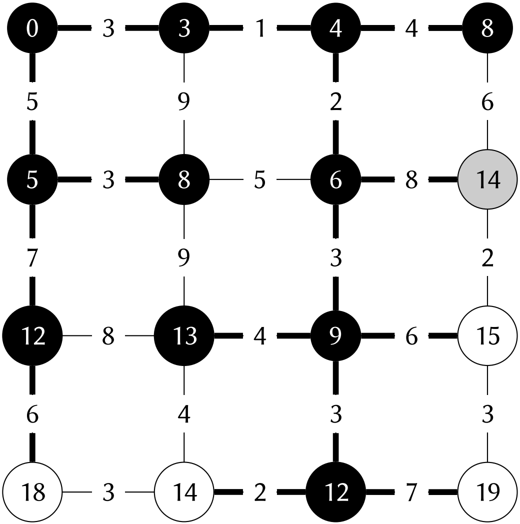
Dijkstra (11)



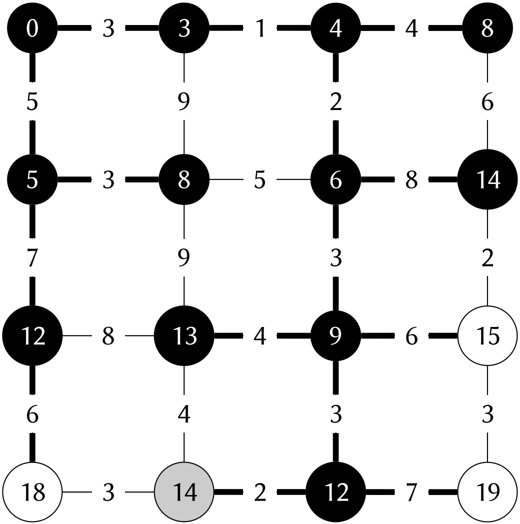
Dijkstra (12)



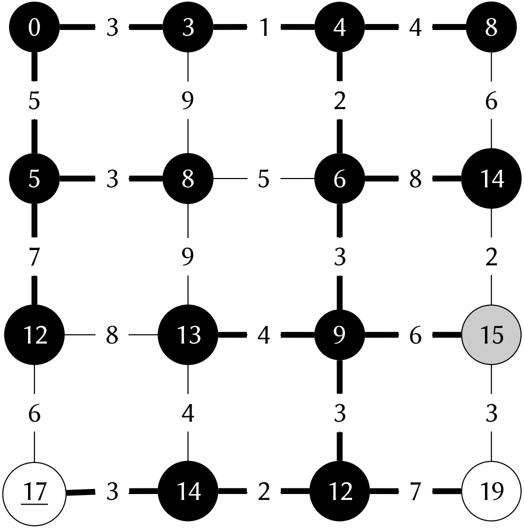
Dijkstra (13)



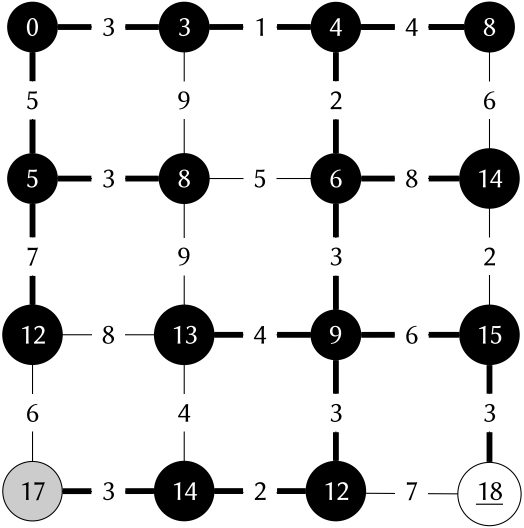
Dijkstra (14)



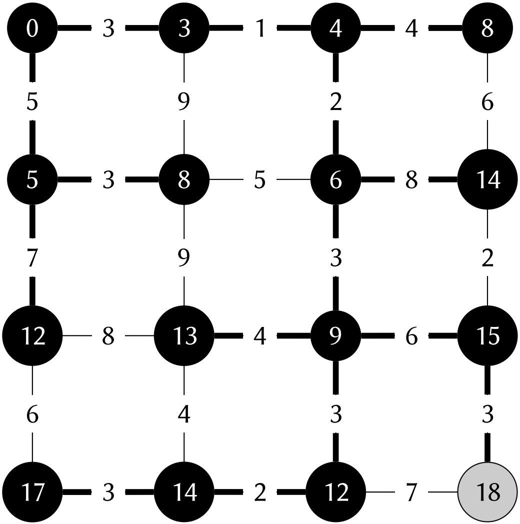
Dijkstra (15)



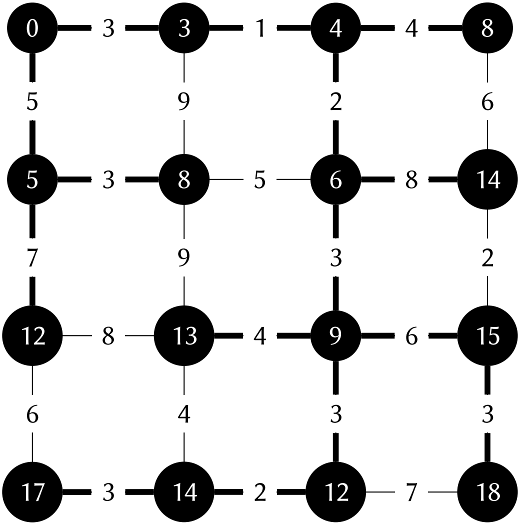
Dijkstra (16)



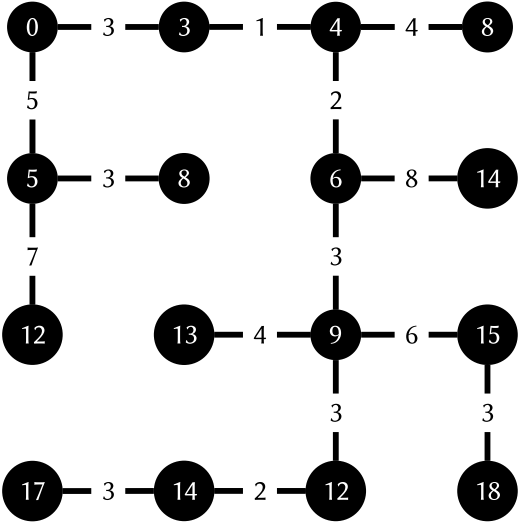
Dijkstra (17)



Dijkstra (18)



Συνεκτικό Δέντρο (Spanning Tree)



Χρήση Ουράς Προτεραιότητας

Για να υλοποιηθεί ο αλγόριθμος, χρειαζόμαστε μια ουρά προτεραιότητας με τις εξής λειτουργίες:

- `Insert`(pq, i, d) εισάγει το i στην ουρά pq με τιμή d
- `ExtractMin`(pq) επιστρέφει το στοιχείο με τη χαμηλότερη τιμή και το αφαιρεί από την ουρά
- `Size`(pq) επιστρέφει τον αριθμό των στοιχείων στην ουρά
- `Update`(pq, v, d) ενημερώνει την ουρά αλλάζοντας την τιμή του στοιχείου v σε d

Ο Αλγόριθμος του Dijkstra

Algorithm 1: Dijkstra's algorithm.

Input: $G = (V, E, s)$, a graph and a starting node s

Output: $(pred, dist)$: *pred* is an array of size $|V|$ such that $pred[i]$ is the predecessor of node i in the shortest path from s; *dist* is an array of size $|V|$ such that $dist[i]$ is the length of the shortest path calculated from node s to i

Data: *pq*, a priority queue of nodes that prioritises by the path length to a node

```
foreach v in V do
    pred[v] ← ∅
    if v ≠ s then
        dist[v] ← ∞
    else
        dist[v] ← 0
    Insert(pq, v, dist[v])
while Size(pq) ≠ 0 do
    u ← ExtractMin(pq)
    foreach v in AdjacencyList(G, u) do
        if dist[v] > dist[u] + Weight(u, v) then
            dist[v] ← dist[u] + Weight(u, v)
            pred[v] ← u
            Update(pq, v, dist[v])
return (pred, dist)
```

Υλοποίηση

- Η ουρά προτεραιότητας μπορεί να υλοποιηθεί μέσω ενός πίνακα.
- Τότε η λειτουργία `Insert(`pq, v, d`)` είναι $pq[v] \leftarrow d$, το οποίο απαιτεί σταθερό χρόνο, δηλαδή $O(1)$.
- Η λειτουργία `Update(`pq, v, d`)` είναι επίσης $pq[v] \leftarrow d$ και απαιτεί χρόνο $O(1)$.
- Η λειουργία `ExtractMin(`pq`)` απαιτεί αναζήτηση κατά μήκος του συνόλου του πίνακα, αφού αυτός δεν είναι ταξινομημένος, και άρα για n στοιχεία στη λίστα απαιτεί χρόνο $O(n)$.

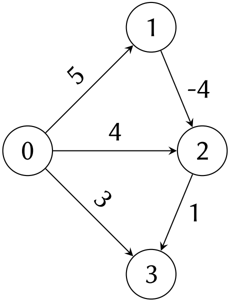
Απόδοση

- Η αρχικοποίηση, γραμμές 1–7, εκτελείται $|V|$ φορές
- Όλες οι λειτουργίες της αρχικοποίησης απαιτούν σταθερό χρόνο, άρα συνολικά απαιτεί χρόνο $O(|V|)$.
- Ο αλγόριθμος εξάγει κάθε κόμβο του γράφου μια φορά, στη γραμμή 9, άρα έχουμε $|V|$ κλήσεις της `ExtractMin(`pq`)` , όπου η κάθε μία απαιτεί χρόνο $O(|V|)$, άρα συνολικά απαιτείται χρόνος $O(|V|^2)$.
- Η διαδικασία χαλάρωσης στις γραμμές 11–14 εκτελείται έως $|E|$ φορές, μία για κάθε σύνδεσμο του γράφου, άρα έχουμε $|E|$ κλήσεις της `Update(`pq, v, d`)` που απαιτούν χρόνο $O(|E|)$.
- Συνολικά, ο αλγόριθμος του Dijkstra απαιτεί χρόνο $O(|V| + |V|^2 + |E|) = O(|V|^2)$.

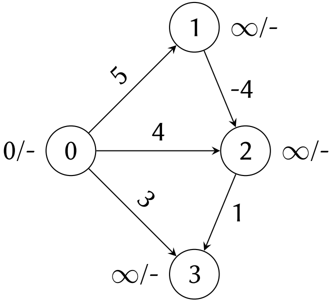
Απόδοση—Βελτιώσεις

- Με χρήση πιο αποδοτικής ουράς προτεραιότητας ο χρόνος μπορεί να μειωθεί σε $O((|V| + |E|) \lg |V|)$.
- Αν όλοι οι κόμβοι του γράφου συνδέονται με τον κόμβο εκκίνησης, τότε ο αριθμός των κόμβων δεν μπορεί να είναι μεγαλύτερος από τον αριθμό των συνδέσμων, και επομένως ο αλγόριθμος τρέχει σε χρόνο $O(|E| \lg |V|)$.

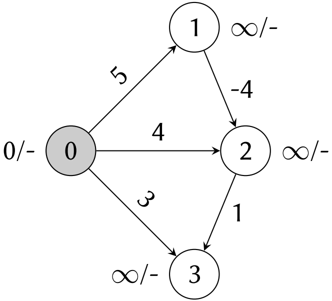
Πότε δεν Δουλεύει ο Αλγόριθμος;



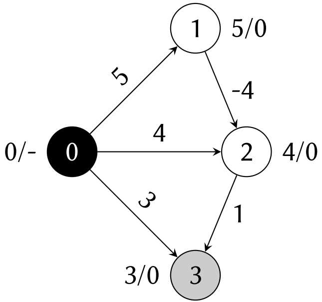
Γράφος με Αρνητικά Βάρη (1)



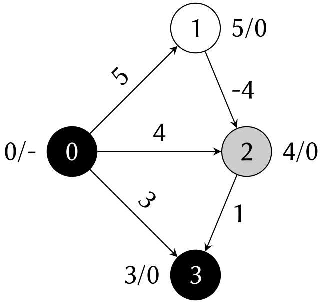
Γράφος με Αρνητικά Βάρη (2)



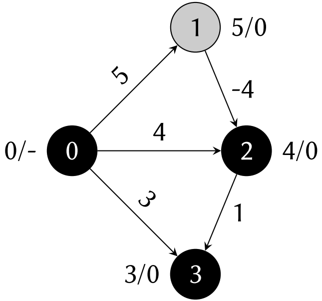
Γράφος με Αρνητικά Βάρη (3)



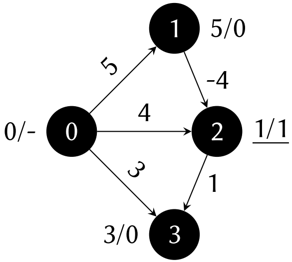
Γράφος με Αρνητικά Βάρη (4)



Γράφος με Αρνητικά Βάρη (5)



Γράφος με Αρνητικά Βάρη (6)



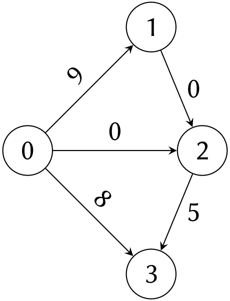
Το συντομότερο μονοπάτι από τον κόμβο 0 στον κόμβο 3 είναι:

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3$$

αλλά ο αλγόριθμος θα αναφέρει ότι είναι το μονοπάτι:

$$0 \rightarrow 3.$$

Γράφος Αναζυγιασμένος



Δυστυχώς η ιδέα δεν δουλεύει.

Συντομότερα Μονομάτια μεταξύ Συνόλου Ζευγών

Algorithm 2: All pairs shortest paths.

Input: $G = (V, E)$, a graph and a starting node s

Output: $(pred, dist)$: *pred* is an array of size $|V| \times |V|$ such that $pred[i][j]$ is the predecessor of node j in the shortest path from i to j; *dist* is an array of size $|V| \times |V|$ such that $dist[i][j]$ is the length of the shortest path calculated from node i to j

foreach *u in V* **do**
  foreach *v in V* **do**
    $pred[u][v] \leftarrow \varnothing$
    $dist[u][v] \leftarrow 0$
foreach *u in V* **do**
  $(pred[u], dist[u]) \leftarrow$ `Dijkstra(`G, u`)`
return *(pred, dist)*

Διάμετρος Γράφου

Ορισμός

Η διάμετρος ενός γράφου είναι το μήκος του μακρύτερου συντομότερου μονοπατιού.

Για να υπολογίσουμε τη διάμετρο ενός γράφου, αρκεί να υπολογίσουμε τα συντομότερα μονοπάτια μεταξύ όλων των ζευγών κόμβων και στη συνέχεια να αναζητήσουμε το μακρύτερο μονοπάτι μεταξύ αυτών που βρήκαμε.