Δίκτυα Υπολογιστών Ι

Ακολουθεί ανάλυση του κώδικα της εργασίας 1 Κεφαλαίου 4 του μαθήματος Δίκτυα Υπολογιστών Ι.

Το πρόγραμμα:

1. Διαβάζει αρχείο με όνομα edges.txt στον ίδιο κατάλογο με το εκτελέσιμο, το αρχείο κειμένου περιέχει τις ακμές στο σχήμα και τα βάρη τους με τη μορφή:
   “Κόμβος1 Κόμβος2 Βάρος”, με κάθε ακμή σε ξεχωριστή γραμμή, οι κόμβοι αντιστοιχούν σε έναν αριθμό οπότε το πρόγραμμα διαβάζει ακέραιος1->Κόμβος1, ακέραιος2->Κόμβος2, ακέραιος3->Βάρος ακμής.

2. Αφού αρχικοποιηθεί ο πίνακας γειτνίασης με άπειρο σε όλες τις θέσεις, οι ακμές γράφονται στον πίνακα με τον εξής τρόπο:
   Έστω η ακμή “1 3 2”, που συνδέει τους κόμβος 1 και 3 με βάρος 2, αυτή μεταφράζεται στον πίνακα ως:

j \ i	0	1	2	3
0	INF	INF	INF	INF
1	INF	INF	INF	2
2	INF	INF	INF	INF
3	INF	2	INF	INF

   Η αρίθμηση των κόμβων αρχίζει από το 1, επομένως δεν χρησιμοποιείται ποτέ η γραμμή και στήλη 0, αυτό έγινε για ευκολότερη ανάγνωση του κώδικα, σημειώνεται επίσης ότι ο πίνακας είναι συμμετρικός ως προς την κύρια διαγώνιο του.

3. Το πρόγραμμα διαβάζει τον κόμβο αφετηρία και δεδομένου ότι είναι πάνω από 0 (αφού η αρίθμηση αρχίζει από το 1) και μικρότερο ή ίσο του αριθμού κόμβων (εφόσον το σχήμα δεν αλλάζει, θα είναι πάντα 6).

4. Ύστερα επιλέγονται ποιοί αλγόριθμοι να χρησιμοποιηθούν.

Dijkstra

Ο αλγόριθμος λειτουργεί ως εξής:

1. Αρχικοποιούνται οι πίνακες rout, from, P, οπού αποθηκεύουν αντίστοιχα το ελάχιστο κόστος, τον προηγούμενο κόμβο στη σειρά για το ελάχιστο μονοπάτι, και αν ο συγκεκριμένος κόμβος έχει διερευνηθεί.

2. Αρχικοποιούνται με άπειρο, 0, και false αντίστοιχα, εκτός από τον αρχικό κόμβο. Ο αρχικός κόμβος έχει κόστος 0 προς τον εαυτό του και είναι επίσης ο δικός του προηγούμενος.
3. Εκτυπώνεται επικεφαλίδα του πίνακα που θα προβάλει την πρόοδο του αλγορίθμου.

4. Αρχίζοντας από τον αρχικό κόμβο, ο αλγόριθμος κάνει τα εξής βήματα:
    a. Εκτυπώνει την πρόοδο του.
    b. Αυξάνει τον μετρητή φάσης κατά 1.
    c. Θέτει τον τωρινό κόμβο ως διερευνημένο.
    d. Για κάθε γείτονα του τωρινού κόμβου που δεν έχει διερευνηθεί:
        i. Ελέγχει αν το κόστος προς τον τωρινό συν το κόστος προς τον γείτονα είναι μικρότερο του κόστους του.
        ii. Αν είναι, το ενημερώνει, και θέτει τον προηγούμενο του γείτονα τον τωρινό.
    e. Βρίσκει τον κόμβο με το ελάχιστο μονοπάτι ο οποίος δεν έχει διερευνηθεί.

5. Στο τέλος, εκτυπώνει το routing table του start, το οποίο περιέχει το συνολικό κόστος μονοπατιού από start προς i, και τον επόμενο κόμβο στην διαδρομή προς i.

Bellman-Ford

Ο αλγόριθμος λειτουργεί πανομοιότηπα με του Dijkstra μέχρι το βήμα 4, με την εξαίρεση ότι δεν χρησιμοποιείται πίνακας P.

4. Το βήμα επαναλαμβάνεται μέχρι να μην γίνονται άλλες αλλαγές στα κόστη. Η επανάληψη κάνει τις εξής ενέργειες:
    a. Αντιγράφεται ο πίνακας rout σε ένα προσωρινό πίνακα, αυτό γίνεται επειδή όταν γίνονται αλλαγές στους πρώτους κόμβους, ενδέχεται να επηρεάσουν τους υπολογισμούς των αργότερων, ενώ η ενημέρωση τους θα έπρεπε να γινόταν στο επόμενο βήμα. Αυτή η μέθοδος είναι τεχνικά αργότερη, όμως εμφανίζει σωστά τα βήματα του αλγορίθμου. Πχ:
    Υλοποίηση με προσωρινό πίνακα:

```
Bellman-Ford
h       1       2       3       4       5       6
0       0(1)    ∞(0)    ∞(0)    ∞(0)    ∞(0)    ∞(0)
1       0(1)    2(1)    5(1)    1(1)    ∞(0)    ∞(0)
2       0(1)    2(1)    4(4)    1(1)    2(4)    10(3)
3       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)
4       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)
```

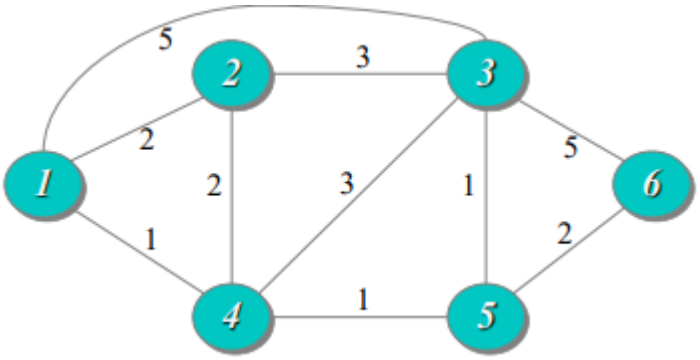
    Υλοποίηση χωρίς προσωρινό πίνακα:

```
Bellman-Ford
h       1       2       3       4       5       6
0       0(1)    ∞(0)    ∞(0)    ∞(0)    ∞(0)    ∞(0)
1       0(1)    2(1)    5(1)    1(1)    2(4)    4(5)
2       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)
3       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)
```

    b. Θέτει την σημαία αλλαγής ως false.
    c. Εμφανίζει την πρόοδο του αλγορίθμου.
    d. Αυξάνει την φάση h κατά 1.
    e. Ύστερα για κάθε κόμβο i:
        i. Για κάθε γείτονα του κόμβου i:
            1. Ελέγχει αν το κόστος από start στο i συν το κόστος από τον i προς τον γείτονα είναι μικρότερο του κόστους από το i προς τον γείτονα. Για τους υπολογισμούς χρησιμοποιείται ο πίνακας rout, και οι νέες τιμές αποθηκεύονται στον πίνακα temp_rout. Αν είναι:
            2. Ορίζει την μεταβλητή αλλαγής σε true.
            3. Ενημερώνει το κόστος του γείτονα.
            4. Θέτει τον προηγούμενο το κόμβο το i.
    f. Αντιγράφει τα νέα κόστη που είναι στον πίνακα temp_rout πίσω στον rout.
5. Στο τέλος εμφανίζει μία τελευταία φορά τον πίνακα
6. Με τον ίδιο τρόπο όπως στον αλγόριθμο του Dijkstra, εμφανίζει το routing table του start.

Ενδεικτικά αποτελέσματα

Χρησιμοποιώντας τα παραδείγματα που δόθηκαν στις διαφάνειες και αλλάζοντας το αρχείο edges.txt και τις σταθερές VERTICES και EDGES, φαίνεται ότι δίνονται τα ίδια αποτελέσματα και με τους δύο αλγόριθμους. Ο αρχικός κόμβος είναι ο 1.



ch4 > edges.txt

```
1   1 3 5
2   1 2 2
3   1 4 1
4   2 3 3
5   2 3 2
6   4 3 3
7   4 5 1
8   3 5 1
9   3 6 5
10  5 6 2
```

Φάση h	1	2	3	4	5	6	P
0	0	∝	∝	∝	∝	∝	
1	-	2 (1)	5 (1)	1 (1)	∝	∝	1
2		2 (1)	4 (4)	-	2 (4)	∝	1,4
3		-	4 (4)		2 (4)	∝	1,4,2
4			3 (5)		-	4 (5)	1,4,2,5
5			-			4 (5)	1,4,2,5, 3
6						-	1,4,2,5, 3,6

```
Dijkstra
h       1       2       3       4       5       6       P
0       0(1)    ∞(0)    ∞(0)    ∞(0)    ∞(0)    ∞(0)
1       0(1)    2(1)    5(1)    1(1)    ∞(0)    ∞(0)    1
2       0(1)    2(1)    4(4)    1(1)    2(4)    ∞(0)    1,4
3       0(1)    2(1)    4(4)    1(1)    2(4)    ∞(0)    1,2,4
4       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)    1,2,4,5
5       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)    1,2,3,4,5
6       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)    1,2,3,4,5,6
```

Φάση h	1	2	3	4	5	6
0	0	∝	∝	∝	∝	∝
1	0	2 (1)	5 (1)	1 (1)	∝	∝
2	0	2 (1)	4 (4)	1 (1)	2 (4)	10 (3)
3	0	2 (1)	3 (5)	1 (1)	2 (4)	4 (5)
4	0	2 (1)	3 (5)	1 (1)	2 (4)	4 (5)

```
Bellman-Ford
h       1       2       3       4       5       6
0       0(1)    ∞(0)    ∞(0)    ∞(0)    ∞(0)    ∞(0)
1       0(1)    2(1)    5(1)    1(1)    ∞(0)    ∞(0)
2       0(1)    2(1)    4(4)    1(1)    2(4)    10(3)
3       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)
4       0(1)    2(1)    3(5)    1(1)    2(4)    4(5)
```

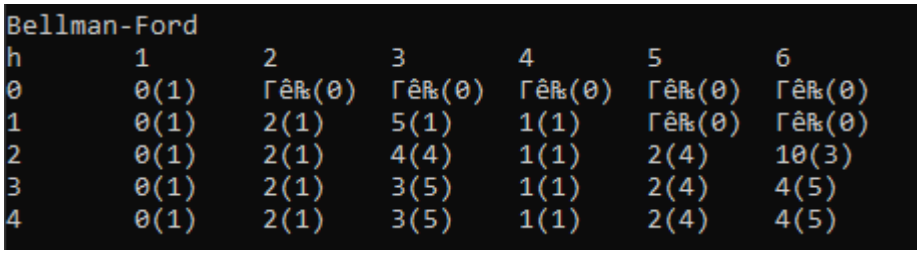
Τα routing table που δημιούργησαν οι δύο αλγόριθμοι είναι ίδια μεταξύ τους, και επίσης με το παράδειγμα.

```
Routing Table (1)
Dest    Next    Cost
1       1       0
2       2       2
3       4       3
4       4       1
5       4       2
6       4       4
```

Routing Table (1)

Destination Node	Next-Hop (Node)	Cost
1	1	0
2	2	2
3	4	3
4	4	1
5	4	2
6	4	4

Σημείωση: Λόγω ζητημάτων υποστήριξης ενδέχεται να μην εμφανίζεται σωστά το σύμβολο του απείρου στο terminal. Πχ:



Κουλουράς Ιωάννης
Ε20075