Δομές Δεδομένων και Αλγόριθμοι - Εργαστήριο 8
Γραφήματα

Τ.Ε.Ι. Ηπείρου, Τμήμα Μηχανικών Πληροφορικής Τ.Ε.
Χρήστος Γκόγκος - Αναπληρωτής Καθηγητής

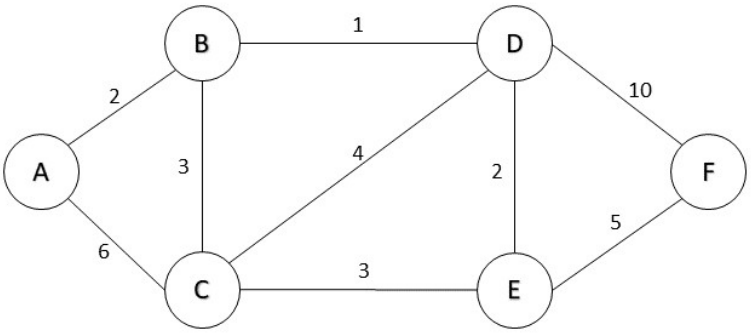
1 Εισαγωγή

Τα γραφήματα είναι δομές δεδομένων που συναντώνται συχνά κατά την επίλυση προβλημάτων. Η ευχέρεια προγραμματισμού αλγορίθμων που εφαρμόζονται πάνω σε γραφήματα είναι ουσιώδης. Καθώς μάλιστα συχνά ανακύπτουν προβλήματα για τα οποία έχουν διατυπωθεί αλγόριθμοι αποδοτικής επίλυσής τους η γνώση των αλγορίθμων αυτών αποδεικνύεται ισχυρός σύμμαχος στην επίλυση δύσκολων προβλημάτων.

2 Γραφήματα

Ένα γράφημα ή γράφος (graph) είναι ένα σύνολο από σημεία που ονομάζονται κορυφές (vertices) ή κόμβοι (nodes) για τα οποία ισχύει ότι κάποια από αυτά είναι συνδεδεμένα απευθείας μεταξύ τους με τμήματα γραμμών που ονομάζονται ακμές (edges ή arcs). Συνήθως ένα γράφημα συμβολίζεται ως $G = (V, E)$ όπου V είναι το σύνολο των κορυφών και E είναι το σύνολο των ακμών.

Αν οι ακμές δεν έχουν κατεύθυνση τότε το γράφημα ονομάζεται μη κατευθυνόμενο (undirected) ενώ σε άλλη περίπτωση ονομάζεται κατευθυνόμενο (directed). Ένα πλήρες γράφημα (που όλες οι κορυφές συνδέονται απευθείας με όλες τις άλλες κορυφές) έχει $\frac{|V||V-1|}{2}$ ακμές ($|V|$ είναι το πλήθος των κορυφών του γραφήματος). Αν σε κάθε ακμή αντιστοιχεί μια τιμή τότε το γράφημα λέγεται γράφημα με βάρη. Το γράφημα του σχήματος 1 είναι ένα μη κατευθυνόμενο γράφημα με βάρη.



Σχήμα 1: Ένα μη κατευθυνόμενο γράφημα 6 κορυφών με βάρη στις ακμές του

2.1 Αναπαράσταση γραφημάτων

Δύο διαδεδομένοι τρόποι αναπαράστασης γραφημάτων είναι οι πίνακες γειτνίασης (adjacency matrices) και οι λίστες γειτνίασης (adjacency lists).

Στους πίνακες γειτνίασης διατηρείται ένας δισδιάστατος πίνακας $n \times n$ όπου n είναι το πλήθος των κορυφών του γραφήματος. Για κάθε ακμή του γραφήματος που συνενώνει την κορυφή i με την κορυφή j εισάγεται στη

θέση i, j του πίνακα το βάρος της ακμής αν το γράφημα είναι με βάρη ενώ αν δεν υπάρχουν βάρη τότε εισάγεται η τιμή 1. Όλα τα υπόλοιπα στοιχεία του πίνακα λαμβάνουν την τιμή 0. Για παράδειγμα η πληροφορία του γραφήματος του σχήματος 1 διατηρείται όπως φαίνεται στον πίνακα 1.

	A	B	C	D	E	F
A	0	2	6	0	0	0
B	2	0	3	1	0	0
C	6	3	0	4	3	0
D	0	1	4	0	2	10
E	0	0	3	2	0	5
F	0	0	0	10	5	0

Πίνακας 1: Πίνακας γειτνίασης

Στις λίστες γειτνίασης διατηρούνται λίστες που περιέχουν για κάθε κορυφή όλη την πληροφορία των συνδέσεών της με τους γειτονικούς του κόμβους. Για παράδειγμα το γράφημα του σχήματος 1 μπορεί να αναπαρασταθεί με τις ακόλουθες 6 λίστες (μια ανά κορυφή). Κάθε στοιχείο της λίστας για τον κόμβο v είναι ένα ζεύγος τιμών (w, u) και αναπαριστά μια ακμή από τον κόμβο v στον κόμβο u με βάρος w, όπως φαίνεται στο πίνακα 2.

A	(2,B), (6,C)
B	(2,A), (3,C), (1,D)
C	(6,A), (3,B), (4,D), (3,E)
D	(1,B), (4,C), (2,E), (10,F)
E	(3,C), (2,D), (5,F)
F	(10,D), (5,E)

Πίνακας 2: Λίστα γειτνίασης

2.2 Ανάγνωση δεδομένων γραφήματος από αρχείο

Υπάρχουν πολλοί τρόποι με τους οποίους μπορεί να βρίσκονται διαθέσιμα τα δεδομένα ενός γραφήματος. Στη συνέχεια παρουσιάζεται μια απλή μορφή αποτύπωσης κατευθυνόμενων με βάρη γραφημάτων χρησιμοποιώντας αρχεία απλού κειμένου. Σύμφωνα με αυτή τη μορφή για κάθε κορυφή του γραφήματος καταγράφεται σε ξεχωριστή γραμμή του αρχείου κειμένου το όνομά της ακολουθούμενο από ζεύγη τιμών που αντιστοιχούν στις ακμές που ξεκινούν από τη συγκεκριμένη κορυφή. Στο κείμενο που ακολουθεί (graph1.txt) και το οποίο αφορά το γράφημα του σχήματος 1 η πρώτη γραμμή σημαίνει ότι ο κόμβος A συνδέεται με μια ακμή με βάρος 2 με τον κόμβο B καθώς και με μια ακμή με βάρος 6 με τον κόμβο C. Ανάλογα καταγράφεται η πληροφορία ακμών και για τις άλλες κορυφές.

```
A 2,B 6,C
B 2,A 3,C 1,D
C 6,A 3,B 4,D 3,E
D 1,B 4,C 2,E 10,F
E 3,C 2,D 5,F
F 10,D 5,E
```

Η ανάγνωση του αρχείου και η δημιουργία μιας δομής με διανύσματα που περιέχει την πληροφορία του γραφήματος μπορεί να γίνει με τη συνάρτηση read_data που δίνεται στη συνέχεια όπου fn είναι το όνομα του αρχείου. Η συνάρτηση print_data εμφανίζει τα δεδομένα του γραφήματος. Ο κώδικας έχει "σπάσει" σε 3 αρχεία (graph.hpp, graph.cpp και graph_ex1.cpp) έτσι ώστε να είναι ευκολότερη η επαναχρησιμοποίηση του.

```
#include <fstream>
```

```
#include <iostream>
#include <map>
#include <sstream>
#include <utility>
#include <vector>


using namespace std;

map<string, vector<pair<int, string>>> read_data(string fn);
void print_graph(map<string, vector<pair<int, string>>> &g);
```

Κώδικας 1: header file με τις συναρτήσεις για ανάγνωση και εμφάνιση γραφημάτων (graph.hpp)

```
#include "graph.hpp"

using namespace std;

map<string, vector<pair<int, string>>> read_data(string fn) {
  map<string, vector<pair<int, string>>> graph;
  fstream filestr;
  string buffer;
  filestr.open(fn.c_str());
  if (filestr.is_open())
    while (getline(filestr, buffer)) {
      string buffer2;
      stringstream ss;
      ss.str(buffer);
      vector<string> tokens;
      while (ss >> buffer2)
        tokens.push_back(buffer2);
      string vertex1 = tokens[0].c_str();
      for (size_t i = 1; i < tokens.size(); i++) {
        int pos = tokens[i].find(",");
        int weight = atoi(tokens[i].substr(0, pos).c_str());
        string vertex2 =
            tokens[i].substr(pos + 1, tokens[i].length() - 1).c_str();
        graph[vertex1].push_back(make_pair(weight, vertex2));
      }
    }
  else {
    cout << "Error opening file: " << fn << endl;
    exit(-1);
  }
  return graph;
}

void print_graph(map<string, vector<pair<int, string>>> &g) {
  for (const auto &p1 : g) {
    for (const auto &p2 : p1.second)
      cout << p1.first << "<--" << p2.first << "-->" << p2.second << " ";
    cout << endl;
  }
}
```

Κώδικας 2: source file με τις συναρτήσεις για ανάγνωση και εμφάνιση γραφημάτων (graph.cpp)

```
#include "graph.hpp"

using namespace std;

```

```
int main() {
  map<string, vector<pair<int, string>>> graph = read_data("graph1.txt");
  print_graph(graph);
  return 0;
}
```

Κώδικας 3: Ανάγνωση και εκτύπωση των δεδομένων του γραφήματος του σχήματος 1 (graph_ex1.cpp)

Η μεταγλώττιση και η εκτέλεση του κώδικα γίνεται με τις ακόλουθες εντολές:

```
$ g++ -Wall -std=c++11 graph.cpp graph_ex1.cpp -o graph_ex1
$ ./graph_ex1
```

Η δε έξοδος που παράγεται είναι η ακόλουθη:

```
A<--2-->B A<--6-->C
B<--2-->A B<--3-->C B<--1-->D
C<--6-->A C<--3-->B C<--4-->D C<--3-->E
D<--1-->B D<--4-->C D<--2-->E D<--10-->F
E<--3-->C E<--2-->D E<--5-->F
F<--10-->D F<--5-->E
```

2.3 Κατευθυνόμενα ακυκλικά γραφήματα

Τα κατευθυνόμενα ακυκλικά γραφήματα (Directed Acyclic Graph=DAG) είναι κατευθυνόμενα γραφήματα για τα οποία δεν μπορεί να εντοπιστεί διαδρομή από μια κορυφή προς την ίδια. Στο σχήμα X παρουσιάζεται ένα γράφημα το οποίο δεν θα ήταν DAG αν υπήρχε μια ακόμα ακμή από το A στο B.

2.4 Σημαντικοί αλγόριθμοι γραφημάτων

Υπάρχουν πολλοί αλγόριθμοι που εφαρμόζονται σε γραφήματα προκειμένου να επιλύσουν ενδιαφέροντα προβλήματα που ανακύπτουν σε πρακτικές εφαρμογές. Οι ακόλουθοι αλγόριθμοι είναι μερικοί από αυτούς:

- Αναζήτηση συντομότερων διαδρομών από μια κορυφή προς όλες τις άλλες κορυφές (Dijkstra)
- Αναζήτηση συντομότερης διαδρομής από μια κορυφή προς μια άλλη κορυφή (Floyd Warshall)
- Αναζήτηση κατά βάθος (Depth First Search). Είναι αλγόριθμος διάσχισης γραφήματος ο οποίος ξεκινά από έναν κόμβο αφετηρία και επισκέπτεται όλους τους άλλους κόμβους που είναι προσβάσιμοι χρησιμοποιώντας της ακμές του γραφήματος. Λειτουργεί επεκτείνοντας μια διαδρομή όσο βρίσκει νέους κόμβους τους οποίους μπορεί να επισκεφθεί. Αν δεν βρίσκει νέους κόμβους οπισθοδρομεί και διερευνά άλλα τμήματα του γραφήματος.
- Αναζήτηση κατά πλάτος (Breadth First Search). Αλγόριθμος διάσχισης γραφήματος που ξεκινώντας από έναν κόμβο αφετηρία επισκέπτεται τους υπόλοιπους κόμβους σε αύξουσα σειρά απόστασης από την αφετηρία.
- Εντοπισμός ελάχιστου συνεκτικού δένδρου (Prim, Kruskal)
- Τοπολογική ταξινόμηση (Topological Sort)
- Εντοπισμός κυκλωμάτων Euler (Eulerian circuit)
- Εντοπισμός ισχυρά συνδεδεμένων συνιστωσών (Stongly Connected Components)

Στη συνέχεια θα παρουσιαστεί ο αλγόριθμος αναζήτησης των συντομότερων διαδρομών από μια κορυφή προς όλες τις άλλες κορυφές. Ο αλγόριθμος αυτός είναι γνωστός και ως αλγόριθμος του Dijkstra.

3 Αλγόριθμος του Dijkstra για εύρεση συντομότερων διαδρομών

Ο αλγόριθμος δέχεται ως είσοδο ένα γράφημα $G = (V, E)$ και μια κορυφή του γραφήματος s η οποία αποτελεί την αφετηρία. Υπολογίζει για όλες τις κορυφές $v \in V$ το μήκος του συντομότερου μονοπατιού από την κορυφή s στην κορυφή v. Για να λειτουργήσει σωστά θα πρέπει κάθε ακμή να έχει μη αρνητικό βάρος. Αν το γράφημα περιέχει ακμές με αρνητικό βάρος τότε μπορεί να χρησιμοποιηθεί ο αλγόριθμος των Bellman-Ford.

3.1 Περιγραφή του αλγορίθμου

Ο αλγόριθμος εντοπίζει τις συντομότερες διαδρομές προς τις κορυφές του γραφήματος σε σειρά απόστασης από την κορυφή αφετηρία. Σε κάθε βήμα του αλγορίθμου η αφετηρία και οι ακμές προς τις κορυφές για τις οποίες έχει ήδη βρεθεί συντομότερο μονοπάτι σχηματίζουν το υποδένδρο S του γραφήματος. Οι κορυφές που είναι προσπελάσιμες με 1 ακμή από το υποδένδρο S είναι υποψήφιες να αποτελέσουν την επόμενη κορυφή που θα εισέλθει στο υποδένδρο. Επιλέγεται μεταξύ τους η κορυφή που βρίσκεται στη μικρότερη απόσταση από την αφετηρία. Για κάθε υποψήφια κορυφή u υπολογίζεται το άθροισμα της απόστασής της από την πλησιέστερη κορυφή v του δένδρου συν το μήκος της συντομότερης διαδρομής από την αφετηρία s προς την κορυφή v. Στη συνέχεια επιλέγεται η κορυφή με το μικρότερο άθροισμα. Όταν επιλεγεί η κορυφή που πρόκειται να προστεθεί στο δένδρο τότε προστίθεται στο σύνολο των κορυφών που απαρτίζουν το υποδένδρο S και για κάθε μία από τις υποψήφιες κορυφές που συνδέονται με μια ακμή με την κορυφή που επιλέχθηκε ενημερώνεται η απόστασή της από το υποδένδρο εφόσον προκύψει μικρότερη τιμή.

Ψευδοκώδικας Το σύνολο S περιέχει τις κορυφές για τις οποίες έχει προσδιοριστεί η συντομότερη διαδρομή από την κορυφή s ενώ το διάνυσμα d περιέχει τις αποστάσεις από την κορυφή s
1. Αρχικά $S = s$, $d_s = 0$ και για όλες τις κορυφές $i \neq s, d_i = \infty$
2. Μέχρι να γίνει $S = V$
3. Εντοπισμός του στοιχείου $v \notin S$ με τη μικρότερη τιμή d_v και προσθήκη του στο S
4. Για κάθε ακμή από την κορυφή v στην κορυφή u με βάρος w ενημερώνεται η τιμή d_u έτσι ώστε:
$$d_u = min(d_u, d_v + w)$$
5. Επιστροφή στο βήμα 2.

Εκτέλεση του αλγορίθμου Στη συνέχεια ακολουθεί παράδειγμα εκτέλεσης του αλγορίθμου για το γράφημα του σχήματος 1.

Συνεπώς ισχύει ότι:

- Για την κορυφή A η διαδρομή αποτελείται μόνο από τον κόμβο A και έχει μήκος 0.
- Για την κορυφή B η διαδρομή είναι η A-B με μήκος 2.
- Για την κορυφή C η διαδρομή είναι η A-B-C με μήκος 5.
- Για την κορυφή D η διαδρομή είναι η A-B-D με μήκος 3.
- Για την κορυφή E η διαδρομή είναι η A-B-D-E με μήκος 5.
- Για την κορυφή F η διαδρομή είναι η A-B-D-E-F με μήκος 10.

Απόδοση του αλγορίθμου Η ταχύτητα εκτέλεσης του αλγορίθμου εξαρτάται από τις δομές δεδομένων που χρησιμοποιούνται για να αναπαρασταθεί το γράφημα. Γενικά, πρόκειται για έναν εξαιρετικά γρήγορο αλγόριθμο με πολυπλοκότητα χειρότερης περίπτωσης $O(|E|log|V|)$, όπου $|E|$ είναι ο αριθμός των ακμών και $|V|$ ο αριθμός των κορυφών του γραφήματος.

3.2 Κωδικοποίηση του αλγορίθμου

```
1  #include <climits>
2  #include <map>
3  #include <set>
4  #include <vector>
5
6  using namespace std;
7
8  void compute_shortest_paths_to_all_vertices(
9      map<string, vector<pair<int, string>>> &graph, string source,
10     map<string, int> &shortest_path_distances);
```

$S = \{A\}, d_A = 0, d_B = 2, d_C = 6, d_D = \infty, d_E = \infty, d_F = \infty$	Από το S μπορούμε να φτάσουμε στις κορυφές 2 και 3 με μήκος διαδρομής 2 και 6 αντίστοιχα. Επιλέγεται η κορυφή 2.
$S = \{A, B\}, d_A = 0, d_B = 2, d_C = 5, d_D = 3, d_E = \infty, d_F = \infty$	Από το S μπορούμε να φτάσουμε στις κορυφές C και D με μήκος διαδρομής 5 και 3 αντίστοιχα. Επιλέγεται η κορυφή D.
$S = \{A, B, D\}, d_A = 0, d_B = 2, d_C = 5, d_D = 3, d_E = 5, d_F = 13$	Από το S μπορούμε να φτάσουμε στις κορυφές C, E και F με μήκος διαδρομής 5, 5 και 13 αντίστοιχα. Επιλέγεται (με τυχαίο τρόπο) ανάμεσα στις κορυφές C και E η κορυφή E.
$S = \{A, B, D, C\}, d_A = 0, d_B = 2, d_C = 5, d_D = 3, d_E = 5, d_F = 13$	Από το S μπορούμε να φτάσουμε στις κορυφές E και F με μήκος διαδρομής 5 και 13 αντίστοιχα. Επιλέγεται η κορυφή E.
$S = \{A, B, D, C, E\}, d_A = 0, d_B = 2, d_C = 5, d_D = 3, d_E = 5, d_F = 10$	Η μοναδική κορυφή στην οποία μένει να φτάσουμε από το S είναι η κορυφή F και το μήκος της συντομότερης διαδρομής από την A στην F είναι 10.
$S = \{A, B, D, C, E, F\}, d_A = 0, d_B = 2, d_C = 5, d_D = 3, d_E = 5, d_F = 10$	

Πίνακας 3: Αναλυτική εκτέλεση του αλγορίθμου

Κώδικας 4: header file (dijkstra.hpp)

```
#include "dijkstra.hpp"

using namespace std;

void compute_shortest_paths_to_all_vertices(
    map<string, vector<pair<int, string>>> &graph, string source,
    map<string, int> &shortest_path_distances) {
  vector<string> S{source};
  set<string> NS;
  for (auto &kv : graph)
    if (kv.first == source)
      shortest_path_distances[kv.first] = 0;
    else {
      NS.insert(kv.first);
      shortest_path_distances[kv.first] = INT_MAX;
    }

  while (!NS.empty()) {
    string v1 = S.back();
    for (pair<int, string> w_v : graph[v1]) {
      int weight = w_v.first;
      string v2 = w_v.second;
      if (NS.find(v2) != NS.end())
```

Σύνολο S	A	B	C	D	E	F
$\{\}$	0	∞	∞	∞	∞	∞
$\{A\}$	0	2_A	6_A	∞	∞	∞
$\{A, B\}$	0	2_A	5_B	3_B	∞	∞
$\{A, B, D\}$	0	2_A	5_B	3_B	5_D	13_D
$\{A, B, D, C\}$	0	2_A	5_B	3_B	5_D	13_D
$\{A, B, D, C, E\}$	0	2_A	5_B	3_B	5_D	10_E
$\{A, B, D, C, E, F\}$	0	2_A	5_B	3_B	5_D	10_E

Πίνακας 4: Συνοπτική εκτέλεση του αλγορίθμου

```
24            if (shortest_path_distances[v1] + weight < shortest_path_distances[v2])
25              shortest_path_distances[v2] = shortest_path_distances[v1] + weight;
26        }
27        int min = INT_MAX;
28        string pmin = "None";
29        for (string v2 : NS) {
30          if (shortest_path_distances[v2] < min) {
31            min = shortest_path_distances[v2];
32            pmin = v2;
33          }
34        }
35        // in case the graph is not connected
36        if (pmin == "None")
37          break;
38        S.push_back(pmin);
39        NS.erase(pmin);
40      }
41    }
```

Κώδικας 5: source file (dijkstra.cpp)

```
1  #include "dijkstra.hpp"
2  #include "graph.hpp"
3
4  using namespace std;
5
6  int main() {
7    map<string, vector<pair<int, string>>> graph = read_data("graph1.txt");
8    map<string, int> shortest_path_distances;
9    string source = "A";
10   compute_shortest_paths_to_all_vertices(graph, source,
11                                          shortest_path_distances);
12   for (auto pair : shortest_path_distances)
13     cout << "Shortest path from vertex " << source << " to vertex "
14          << pair.first << " has length " << pair.second << endl;
15 }
```

Κώδικας 6: source file (dijkstra_ex1.cpp)

Η μεταγλώττιση και η εκτέλεση του κώδικα γίνεται με τις ακόλουθες εντολές:

```
1 $ g++ -std=c++11 graph.cpp dijkstra.cpp dijkstra_ex1.cpp -o dijkstra_ex1
2 $ ./dijkstra_ex1
```

Η δε έξοδος που παράγεται είναι η ακόλουθη:

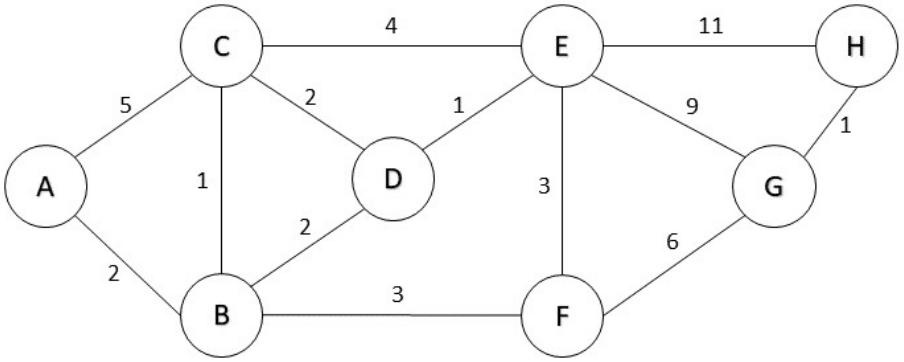
```
1 Shortest path from vertex A to vertex A has length 0
2 Shortest path from vertex A to vertex B has length 2
3 Shortest path from vertex A to vertex C has length 5
```

```
4 Shortest path from vertex A to vertex D has length 3
5 Shortest path from vertex A to vertex E has length 5
6 Shortest path from vertex A to vertex F has length 10
```

4 Παραδείγματα

4.1 Παράδειγμα 1

Για το σχήμα 2 και με αφετηρία την κορυφή A συμπληρώστε τον πίνακα εκτέλεσης του αλγορίθμου για την εύρεση των συντομότερων διαδρομών του Dijkstra και καταγράψτε τις διαδρομές που εντοπίζονται από την αφετηρία προς όλες τις άλλες κορυφές.



Σχήμα 2: Ένα μη κατευθυνόμενο γράφημα 8 κορυφών με βάρη στις ακμές του

Ο ακόλουθος πίνακας δείχνει την εκτέλεση του αλγορίθμου

Σύνολο S	A	B	C	D	E	F	G	H
$\{\}$	0	∞	∞	∞	∞	∞	∞	∞
$\{A\}$	0	2_A	5_A	∞	∞	∞	∞	∞
$\{A,\}$	0	2_A	3_B	∞	∞	5_B	∞	∞
$\{A,B,C\}$	0	2_A	3_B	4_B	7_C	5_B	∞	∞
$\{A,B,C,D\}$	0	2_A	3_B	4_B	5_D	5_B	∞	∞
$\{A,B,C,D,E\}$	0	2_A	3_B	4_B	5_D	5_B	14_E	16_E
$\{A,B,C,D,E,F\}$	0	2_A	3_B	4_B	5_D	5_B	11_F	16_E
$\{A,B,C,D,E,F,G\}$	0	2_A	3_B	4_B	5_D	5_B	11_F	12_E
$\{A,B,C,D,E,F,G,H\}$	0	2_A	3_B	4_B	5_D	5_B	11_F	12_E

Πίνακας 5: Συνοπτική εκτέλεση του αλγορίθμου

Οι συντομότερες διαδρομές είναι:

- Για την κορυφή A η διαδρομή είναι η A με μήκος 0
- Για την κορυφή B η διαδρομή είναι η A-B με μήκος 2
- Για την κορυφή C η διαδρομή είναι η A-B-C με μήκος 3
- Για την κορυφή D η διαδρομή είναι η A-B-D με μήκος 4
- Για την κορυφή E η διαδρομή είναι η A-B-D-E με μήκος 5
- Για την κορυφή F η διαδρομή είναι η A-B-F με μήκος 5
- Για την κορυφή G η διαδρομή είναι η A-B-F-G με μήκος 11
- Για την κορυφή H η διαδρομή είναι η A-B-F-G-H με μήκος 12

4.2 Παράδειγμα 2

Γράψτε πρόγραμμα που να διαβάζει ένα γράφημα όπως έχει περιγραφεί στην παράγραφο Χ και να εμφανίζει για κάθε κορυφή το βαθμό της, δηλαδή το πλήθος των κορυφών με τις οποίες συνδέεται απευθείας καθώς και το άθροισμα από τα βάρη αυτών των ακμών. Επιπλέον για κάθε κορυφή να εμφανίζει τις κορυφές οι οποίες μπορούν να προσεγγιστούν με διαδρομές μήκους 1,2,3 κοκ.

5 Ασκήσεις

1. Υλοποιήστε τον αλγόριθμο των Bellman-Ford για την εύρεση της συντομότερης διαδρομής από μια κορυφή προς όλες τις άλλες κορυφές.
2. Υλοποιήστε έναν αλγόριθμο τοπολογικής ταξινόμησης για DAGs.

Αναφορές

[1] Geeks for Geeks, Graphs and its representations https://www.geeksforgeeks.org/graph-and-its-representations/

[2] Algorithm visualization, Dijkstra's shortest path https://www.cs.usfca.edu/ galles/visualization/Dijkstra.html