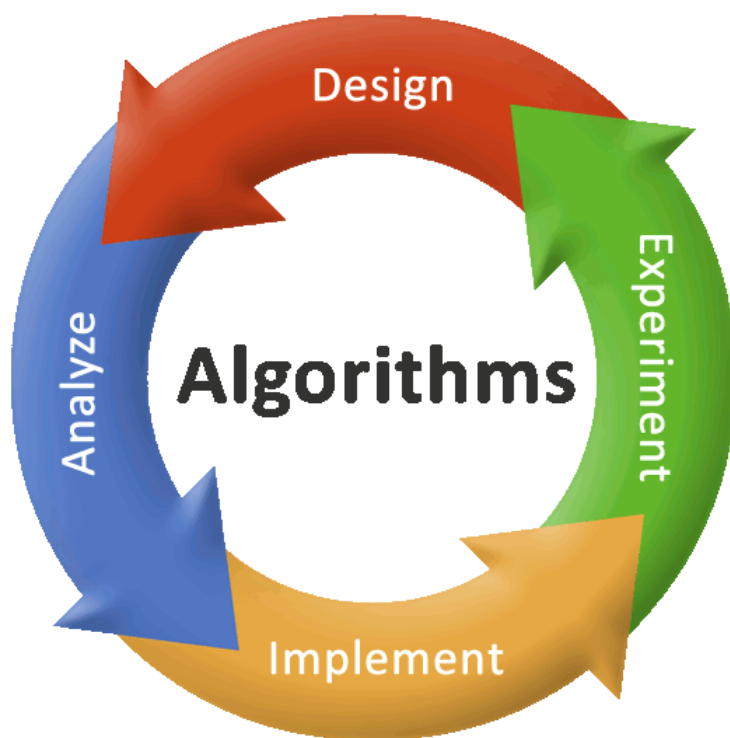


ACADEMIA TEHNICA MILITARA



## Algoritmul Edmonds-Karp

*Autori:*  
Marius SAVA  
Mihai FLOREA

*Indrumator:*  
Mihai PURA

June 17, 2013

# Algoritmul Edmonds-Karp

## 1. Structuri de date

### 1.1 Concept

Algoritmul Edmonds-Karp, publicat de Jack Edmonds si Richard Karp reprezinta o implementare eficienta a algoritmului Ford Fulkerson. Ideea care sta la baza algoritmului este de a identifica la fiecare pas un drum de crestere care contine un numar minim de arce. Adica, intr-un graf, in care muchiile au capacitate si costuri, vom incerca sa gasim cel mai scurt drum, cu costuri minime si cu flux maxim, intre doua noduri diferite.

### 1.2 Definitie

O retea de transport este un graf orientat  $G = (V, E)$  cu proprietatile:

1. exista doua noduri speciale in  $v$  :  $s$  este nodul sursa si  $t$  este nodul terminal.
2. este definita o functie totala de capacitate  $c: V \times V \rightarrow \mathbb{R}^+$  astfel incat:
  - $c(u,v)=0$  daca  $(u,v) \notin E$
  - $c(u,v) \geq 0$  daca  $(u,v) \in E$
3. pentru orice nod  $v \in V \setminus \{s, t\}$  exista cel putin o cale  $s \rightarrow v \rightarrow t$

Acestea sunt definitiile unora dintre notiunile cu care vom lucra. Vom defini pasii necesari pentru acest algoritm in cele ce urmeaza.

### 1.3 Implementare

Implementarea Edmonds-Karp alege intotdeauna cea mai scurta cale folosind o cautare in latime in graful rezidual unde fiecare arc are ponderea 1. Se poate demonstra ca lungimea cailor gasite astfel creste monoton cu fiecare noua ameliorare.

Ce este un drum de ameliorare? Un drum de ameliorare este un drum de la sursa la destinatie care are proprietatea ca fiecare muchie are  $cap[i] - flux[i] > 0$ , sau mai bine zis, "se mai poate pompa lichid prin conductele ce compun drumul".

Conform acestei metode se poate implementa un algoritm cu o complexitate ce depinde de capacitati (marim fluxul cu o unitate la fiecare pas).

Edmonds si Karp au dat urmatorul algoritm ce are complexitate  $O(N * M^2)$ : de ameliorare il vom determina folosind un BFS (cautare in latime) si in loc sa marim fluxul cu o singura unitate, vom mari fluxul cu valoarea minima dintre capacitatile de pe drumul de ameliorare. Se observa intuitiv ca dupa saturarea unui drum de ameliorare se satureaza cel putin o muchie. Dupa  $O(M)$  saturari de drumuri se observa ca cel mai scurt drum de la sursa la destinatie trebuie sa creasca. Asadar

dupa  $O(N \cdot M)$  astfel de operatii destinatia nu va mai fi accesibila din sursa si prin urmare avem fluxul maxim. Cum fiecare operatie (din cele  $O(N \cdot M)$ ) au complexitate  $O(M+N)$  (BFS) rezulta complexitatea finala  $O(N \cdot M^2)$ .

## 1.4 Aplicatii

Algoritmul Edmonds-Karp este folosit in diverse domenii des intalnite, de la controlul avioanelor de pe un anumit aeroport pana la jocuri informatice, rolul principal fiind acela de gasire a drumului de cost minim intre un obiect principal si o tinta anume. Aici se afla o lista de probleme care se rezolva folosind flux maxim:

critice  
senat  
drumuri2  
Two shortest  
trafic  
paznici  
joc4

## 1.5 Avantaje si dezavantaje in raport cu structuri de date similare

Algoritmul Edmonds-Karp al drumului cel mai scurt are complexitatea  $O(nm^2)$ . Cateva dintre caracteristicile algoritmului sunt:

1. Implementare speciala a algoritmului de etichetare.
2. Mareste fluxul de-a lungul celor mai scurte drumuri de la nodul  $s$  la nodul  $t$  din retea reziduala.

# 2. Algoritmi

## 2.1 Algoritmul Ford-Fulkerson

Aceasta este o metoda iterativa de gasire a fluxului maxim intr-un graf care pleaca de la ideea urmatoare: cat timp mai exista un drum de ameliorare (o cale de la sursa la destinatie), pot pompa pe aceasta cale un flux suplimentar egal cu capacitatea reziduala a caii.

### **Ford Fulkerson**

**Input**  $G(V, E), s, t$

**Output**  $|f_{max}|$

$f(u, v) \leftarrow 0, \forall (u, v) \in V \times V$

**while**  $\exists$  a path  $p(s \rightarrow t)$  in  $G_f$  such that  $c_f(u, v) > 0, \forall (u, v) \in p$

**find**  $c_f(p) = \min\{c_f(u, v) | c_f(u, v) \in p\}$

```

for-each  $(u, v) \in p$ 
     $f(u, v) \leftarrow f(u, v) + c_f(p)$ 
     $f(v, u) \leftarrow -f(u, v)$ 
return  $|f_{max}|$ 

```

Acest algoritm reprezinta mai mult un sablon de rezolvare pentru ca nu detaliaza modul in care se alege drumul de ameliorare din reseaua reziduala.

Asa cum am vazut, algoritmul Ford-Fulkerson nu defineste o metoda de alegere a drumului de ameliorare pe baza caruia se modifica fluxul in paragraf. Implementarea Edmonds-Karp alege intotdeauna cea mai scurta cale folosind o cautare in latime in graful rezidual unde fiecare arc are ponderea 1.

## 2.1 Algoritmul Edmonds-Karp

### Descriere

**Edmonds-Karp**

**Input**  $G(V, E), s, t$

**Output**  $|f_{max}|$

$f(u, v) \leftarrow 0, \forall (u, v) \in V \times V$

**while true**

$p(s \rightarrow t) \leftarrow BFS(G_f, s, t)$

**if not**  $\exists p(s \rightarrow t)$

**break;**

**find**  $c_f(p) = \min\{c_f(u, v) | c_f(u, v) \in p\}$

**for-each**  $(u, v) \in p$

$f(u, v) \leftarrow f(u, v) + c_f(p)$

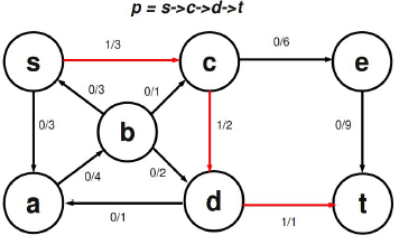
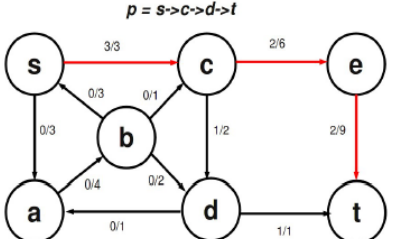
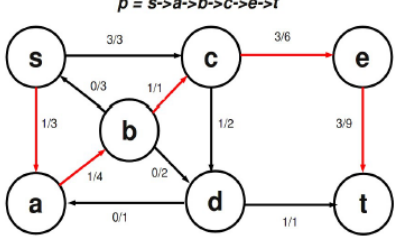
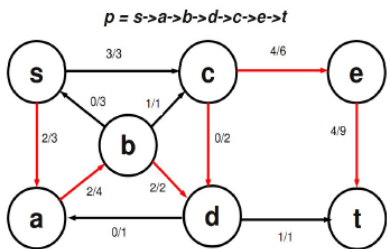
$f(v, u) \leftarrow -f(u, v)$

**return**  $|f_{max}|$

Plecand de la ideea ca drumurile de ameliorare gasite au lungimi din ce in ce mai mari se poate arata ca in aceasta implementare fluxul se mareste de cel mult  $O(V * E)$  deci complexitatea algoritmului va fi  $O(V * E^2)$ .

### Exemplificare

Sa luam un exemplu de rulare al acestui algoritm. Vom considera starea retelei dupa ce a fost gasita prima cale de pompare flux (initial toate arcele sunt etichetate cu 0/0 conform notatiei stabilite).

Capacitatea reziduala a caii de ameliorare	Graful si calea de ameliorare gasita
$c_f(p) = \min(c_f(s,c), c_f(c,d), c_f(d,t)) =$ $= \min(3-0, 2-0, 1-0) = \min(3, 2, 1) = 1$	
$c_f(p) = \min(c_f(s,c), c_f(c,e), c_f(e,t)) =$ $= \min(3-1, 6-0, 9-0) = \min(2, 6, 9) = 2$	
$c_f(p) = \min(c_f(s,a), c_f(a,b), c_f(b,c),$ $c_f(c,e), c_f(e,t)) =$ $= \min(3-0, 4-0, 1-0, 6-2, 9-2) =$ $= \min(3, 4, 1, 4, 7) = 1$	
$c_f(p) = \min(c_f(s,a), c_f(a,b), c_f(b,d),$ $c_f(d,c), c_f(c,e), c_f(e,t)) =$ $= \min(3-1, 4-1, 2-0, 0-(-1), 6-3, 9-3) =$ $= \min(2, 3, 2, 1, 3, 6) = 1$	

## 2.2 Implementare C

```

include <stdio>
define N 128
define oo 0x3f3f3f3f //infinite
int cap[N][N], flux[N][N];
int t[N];
int n, m;
int bfs (int source, int sink)
{
    int Q[N + 1], p = 0, q = 0;
    bool use[N];
    memset (use, 0, sizeof (use));
    memset (t, 0, sizeof (t));
    Q[0] = source;
    use[source] = 1;
    while (p != q)
    {
        int u = Q[p++]; //scoatem primul element din coada
        for (int i = source; i != sink; ++i) // pt fiecare nod ( adjacent )
            if (!use[i]) // nu am folosit nodul
                if (cap[u][i] - flux[u][i] > 0) // mai putem pompa?
                {
                    Q[++q] = i; // inseram nodul i in coada
                    t[i] = u;
                    use[i] = 1;
                }
    }
    if (t[sink])
        return 1;
    return 0;
}

int edmond-karp (int source, int sink)
{
    int flow = 0; //fluxul
    int i, min;
    while (bfs (source, sink)) // cat timp mai exista un drum de ameliorare
    {
        min = oo;
        for (i = sink; i != t[i])
            if (cap[ t[i] ][i] - flux[ t[i] ][i] < min)
                min = cap[ t[i] ][i] - flux[ t[i] ][i]; //calculam minimul dintre ca-
        pacitatile ramase de pe drum
        for (i = sink ; i != t[i])

```

```

    {
        flux[ t[i] ][i] += min; //adaugam min la fluxul de pe arcele de pe drum
        flux[i][ t[i] ] -= min; //scadem minimul de pe arcele inverse
    }
    flow += min; // adaugam minimul la flux
}
return flow;
}

```

### 3. Referinte

- Introducere in algoritmi, Thomas H. Cormen, Charles E. Leiserson, Ronald R. Rivest Capitolul VI Algoritmi pe grafuri: Flux maxim
- Introducere in analiza algoritmilor, Cristian A. Giumale Cap. V Algoritmi pe grafuri: Fluxuri maxime intr-un graf
- Resurse wiki- Edmonds-Karp, Max Flow - Min Cut Theorem
- Aplicatii flux maxim