**Demonstratie Kruskal & Prim**

Initial E(T) = 0 ∈  apcm.

Presupunem ca iınainte de etapa i avem E(T) ∈  apcm.

Fie e = uv muchia aleasa de algoritm la pasul i si S = componenta conexa a lui u in padurea curenta T - v.

Atunci toate muchiile din E(T) sunt in interiorul componentelor conexe ale lui T, deci nu exista muchie in E(T) de la S la V-S.

Rezulta E(T) ∪{e} ∈  apcm.

**Pseudocod Kruska:**

Intrare: G = (V;E;w) - graf conex ponderat

Iesire: un apcm T al lui G

Idee: La un pas este selectata o muchie de cost minim din G care nu formeaza cicluri cu muchiile deja selectate in T (care uneste doua componente conexe din graful deja construit)

Pseudocod:

T = (V,0): initial fiecare varf formeaza o componenta conexa in T

pentru i = 1; n-1 executa

alege o muchie ei = uv de cost minim din G astfel incat u s¸i v sunt in componente conexe

diferite in T

E(T) = E(T) ∪ {uv} reuneste compunenta lui u si componenta lui v

**Pseudocot Prim:**

Intrare: G = (V;E;w) - graf conex ponderat

Iesire: un apcm T al lui G

Idee: Se pornes¸te de la un varf s (care formeaza arborele initial). La un pas este selectata o muchie de

cost minim de la un varf deja adaugat la arbore la unul neadaugat; se adauga acest varf la arbore impreunacu muchia selectata.

Pseudocod:

Fie s- varful de start; T = ({s}, 0)

pentru i = 1; n - 1 executa

alege o muchie uv cu cost minim astfel incat u ∈ V(T) si v = ! ∈ V (T)

E(T) = E(T) ∪{uv}

V (T) = V (T) ∪ {v} adauga v la arbore

**Demonstratie Dijkstra**

Initial avem d[s] = δ (s, s) s¸i primul varf extras din Q este s, deci afirmatia se verifica dupa prima

iteratie.

Presupunem ca afirmatia este adevarata pana la iteratia curenta: d[x] = δ (s,x) pentru orice x ∈ V-Q.

Fie u ∈ Q varful extras la aceasta iteratie. Demonstram ca d[u] = δ (s,u) (adica eticheta lui u este

corect calculata, fiind egala cu distanta de la s la u)

Daca u nu este accesibil din s atunci, d[u] = ∞ = δ (s,u).

Altfel, fie P un drum minim de la s la u (avem w(P) = δ (s, u)).

Deoarece P are o extremitate s ∈ V -Q si cealalta extremitate u ∈ Q => P contine un arc cu o extremitate in V-Q si cealalta in Q. Fie xy primul astfel de arc din P (cu x ∈ V - Q si y ∈ Q)

Dupa relaxarea arcului xy avem:

d[y] <=d [x] + w(xy) = w([s-x]) + w(xy) = w([s-y])

Dar, deoarece la pasul curent si u si y sunt ın Q, din modul ın care algoritmul alege varful u avem

d[u] <=d[y] si deci d[y] <=w(P) = δ (s u) <= d[u]:

Rezulta:

d[u] = d[y] = w(P) = δ(s; u)

**Pseudocod Dijkstra**

Intrare: G = (V,E,w) - graf orientat ponderat cu ponderi pozitive (w : E -> R+) , s - varf de start

Iesire:

- vector d de distante, cu semnificatia d[x] = distanta de la s la varful x

- vectorul tata, reprezentand un arbore al distantelor fata de s (din care se poate determina cate un

drum minim de la s la fiecare varf x)

Idee: Fiecare varf u are asociata o eticheta de distanta d[u] = costul minim al unui drum de la s la u

descoperit pana la acel pas.

La un pas este ales ca varf curent varful u care estimat a fi cel mai apropiat de s si se descopera noi

drumuri catre vecinii lui u, actualizandu-se etichetele de distanta ale acestora.

Pseudocod:

initializeaza multimea varfurilor nevizitate Q cu V

pentru fiecare u ∈ V executa

d[u] = ∞; tata[u] = 0

d[s] = 0

cat timp Q != 0; executa

u = extrage un varf cu eticheta d minima din Q

pentru fiecare uv ∈ E executa //relaxarea arcului uv

daca d[u] + w(u, v) < d[v] atunci

d[v] = d[u] + w(u; v);

tata[v] = u;

scrie d, tata

**Retele de transport:**

O retea de transport este un cvintuplu N = (G,S, T, I, c) unde

\_-G = (V,E) este un graf orientat cu V = S [ I [ T, unde S, I, T sunt multimi disjuncte, nevide,

cu semnificatia:

– S - multimea surselor (intrarilor)

– T - multimea destinatiilor (iesirilor)

– I - multimea varfurilor intermediare

\_-c : E -> N este functia capacitate, cu semnificatia: c(e) = cantitatea maxima care poate fi

transportata prin arcul e.

**Fluxul:**

Fluxul este o functie f : E ->N cu proprietatile:

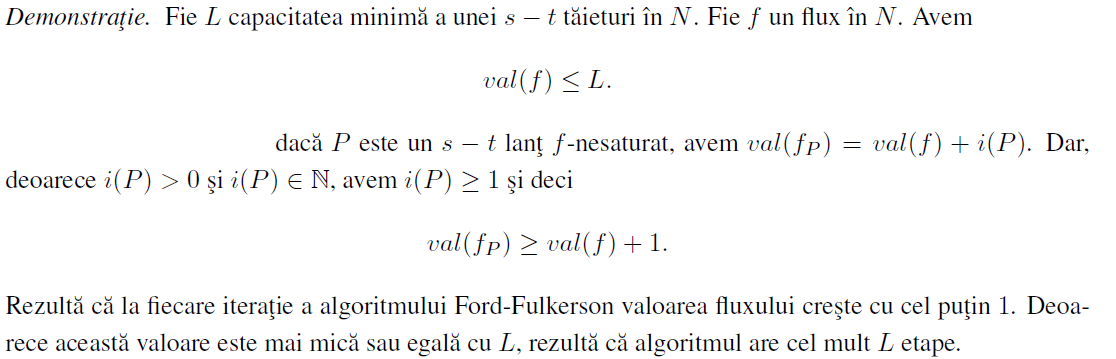
1. 0 <=f(e) <= c(e); ∀e ∈ E(G) - condita de marginire

2. pentru orice varf intermediar v ∈ I, fluxul (total) care intra in varful v este egal cu fluxul (total)

care iese din varful v

**Capacitatea unei taieturi** K = (X,Y ), notata cu c(K), se defineste ca fiind suma capacitatilor arcelor directe ale taieturii (=suma capacitatilor arcelor care ies din X)

**Demonstratie Ford-Fulkerson**



**Pseudocod Ford-Fukerson**

Intrare: N = (G,s, t, I, c) - retea de transport cu capacitati numere naturale

Iesire: un flux maxim f ın G (+ o s - t taietura minima in G)

Idee: La un pas este revizuit fluxul curent pe un s-t lant¸ nesaturat . Se repeta revizuriea pana cand nu mai exista un astfel de lant¸.

Pseudocod:

1. Fie f un flux in N

2. Cat timp exista un s - t lant¸ f-nesaturat P in G

determina un astfel de lant¸ P in G (()

revizuieste fluxul f de-a lungul lantului P

3. Fie X= multimea varfurilor accesibile din s prin lanturi f-nesaturate

returneaza f, K = (X; V - X)

**Grafuri Planare**

G=(V,E) se numeste graf planar daca exista o reprezentare in plan a lui G astfel incat muchiile nu se intersecteaza.

O reprezentare planara a lui G se nueste **harta** M=(V,E,F), unde F este multimea fetelor

O **fata** este o suprafata delimitate de un ciclu elementar fara a fi daiata de un drum

d(f)= gradul unei fete;

d(F)=|E|;

**Teorema Poliedrala a lui Euler**: |V|-|E|+|F|=2