

%

## Zápočtový program

%

souvislost grafu

%

### popis algoritmu a postupu

% Program využívá algoritmu na násobení matic sousednosti A.  
% Příslušná mocnina n matice A určuje z kterých do kterých  
% vrcholů se lze dostat po cestě délky nejvýše n. Matice na  
% nultou je jednotková matice, jenom z vrcholu do toho samého  
% se lze dostat po cestě délky nula. První mocnina je samotná  
% matice A, nenulové prvky jsou právě mezi vrcholy, kde je  
% hrana. Výhodnější, co se týče početní náročnosti, nahrazovat  
% nenulové prvky jedničkami, je to tak jednodušší. A pokud je  
% graf na N vrcholech a příslušnou matici sousednosti umocním  
% na N-tou, potom v matici, co vznikne, budou jedničky právě  
% tam, kde existuje z vrcholu do vrcholu cesta nejvýše N.  
% Kdyby tam bylo místo [x,y], kde bude nula, pak se z vrcholu  
% x do vrcholu y nelze dostat cestou menší rovnou N. Protože  
% je to graf na N vrcholech, nemůže tam existovat nejkratší  
% cesta mezi libovolnými dvěma vrcholy větší než N. Proto mezi  
% vrcholy x a y není cesta žádná, leží v různých komponentách  
% souvislosti. Umocním matici A na N-tou a pak každý řádek  
% umocněné matice M značí jednu komponentu souvislosti, kde  
% jedničky jsou na místech, které indexují vrcholy patřící do  
% právě jedné komponenty souvislosti.  
% Kvůli ušetření počtu násobení dvou matic, nebudu násobit  
% maticí sousednosti A, ale začnu maticí A na druhou a to celé  
% na druhou a tak dále až narazím na mocninu dvojky, která je  
% větší než požadované N. To ale vůbec nevadí. Pokud  
% neexistuje cesta délky N a menší, tak nemůže existovat ani  
% cesta delší.  
% Matice M je řádu N x N a počet komponent souvislosti  
% nemusí být zdaleka N, každá z komponent je uvedena tolikrát  
% kolik vrcholů je v ní obsaženo. Proto po umocnění matice se  
% na ní začnu dívat místo seznam řádků, ale jako seznam  
% komponent souvislosti a vyházím všechny duplicity. A  
% spočítám kolik prvků má upravené pole a právě toto číslo je  
% počet komponent grafu G.  
% Graf se zadává ve formě seznamu hran, nebo také jako  
% matice souvislosti. Ze seznamu hran vytvořím matici  
% sousednosti a pak následuje již mocnění. Seznam hran se  
% předělává do matice takto: 1. vytvoří se matice N x N ze  
% samých nul. 2. přidá se diagonála, vznikne jednotková matice  
% 3. bere se 1 hrana {a,b} a přidá se jednička do matice na  
% místo [a,b] a [b,a].

% VSTUPNI DATA

**matice**([[1,1,0,0,0],[1,1,1,0,0],[0,1,1,0,0],[0,0,0,1,1],[0,0,0,1,1]],1).  
**matice**([[1,1,1],[1,1,0],[1,0,1]],2).

```

seznamhran([[1,2],[3,2],[4,5]],5).
seznamhran([[1,3],[4,3],[2,7],[6,5]],7).
seznamhran([[1,8],[8,2],[2,3],[3,1],[4,5]],8).

kompzeseznamu(S,N,K,P):-
    seznamhran(S,N),
    vytvornatici(S,N,V),
    nantou2(V,W),
    setrid(W,K),
    velikostpole(K,P).

kompzmatice(V,K,P):-
    matice(V,_),
    nantou2(V,W),
    setrid(W,K),
    velikostpole(K,P).

% prehozeni reprezentace matice ze seznamu radek na seznam sloupcu
% matice je implicitne seznam radku, pokud je myslena jako seznam sloupcu,
% je to uvedeno; je to transponovani (slo by take pouzit), ale je mysleno
% jako stejna matice s jinou reprezentaci
% prehod(+Co,-Kam).
prehod([[_|_],[]]).
prehod(M,[P|D]):-
    ubersloupec(M,P,M2),
    prehod(M2,D).

% z matice vznikne jako jeden vyst. parametr jeji prvni sloupec a druhy
% zbytek matice
% ubersloupec(+matice,-radka,-matice bez sloupce)
ubersloupec([],[],[]).
ubersloupec([[X|K]|M],[X|S],[K|M2]):-
    ubersloupec(M,S,M2).

% spocita kvadrat matice
% matkvadrat2(+matice,-kvadrat).
matkvadrat2(M,V):-
    prehod(M,Mm),
    matkvadrat(Mm,M,V).

% spocita druhou mocninu dane matice
% matkvadrat(+Puvodni M(seznam sloupcu),+M,-M na druhou).
matkvadrat(_,[],[]).
matkvadrat(M,[Pr|Dr],[Ra|Da]):-
    matradek(M,Pr,Ra),
    matkvadrat(M,Dr,Da),!.

% nasobeni matice s vektorem, vznikne vektor
% matradek(+matice(seznam sloupcu),+radek,-radek)
matradek([],_,[]).
matradek([P|D],Rvs,[R1|Rd]):-
    snasobradky2(P,Rvs,R1),
    matradek(D,Rvs,Rd).

% vypocita skalarni soucin dvou vektoru zadanych ve forme seznamu
% snasobradky(+seznam,+seznam,-vysledek).
snasobradky([],[],0).
snasobradky([X|S1],[Y|S2],V):-
    snasobradky(S1,S2,V1),
    S is X * Y,
    V is V1 + S.

% pro prehlednost pri nasobeni matic pro potrebu souvislostia nesouvislosti
% budu hodnoty nulove nechavat a hodnoty nenulove konvertovat na jednicky

```

```
snasobradky2([], [], 0).
snasobradky2([X|S1], [Y|S2], V):-
    snasobradky2(S1, S2, V1),
    S is X * Y,
    Vv is V1 + S,
    Vv > 0,
    V is 1.
snasobradky2([X|S1], [Y|S2], V):-
    snasobradky2(S1, S2, V1),
    S is X * Y,
    V is V1 + S.

% algoritmus nam v matici vyhodi jednotlivé komponenty souvislosti,
% potrebujú, aby každá byla zobrazena pouze jednou.
% setrid(+matice, -seznam komponent)
setrid([], []).
setrid([P|D], [P|Da]):-
    vypustvsechny(P, D, D1),
    setrid(D1, Da).

% vypusti vsechny vyskyty ze seznamu
% vypustvsechny(+co, +zceho, -covznikne).
vypustvsechny(X, [X|D], Da):-vypustvsechny(X, D, Da), !.
vypustvsechny(X, [Y|D], [Y|Da]):-vypustvsechny(X, D, Da).
vypustvsechny(_, [], []).

% vynasobi matici na n-tou, kde n je nejmensi vetsi mocnina dvojky
% nantou(+matice, -maticenantou, +pocet aktualni, +pocet konecny).
nantou(M, N, A, P):-
    matkvadrat2(M, Nta),
    A =< P,
    Aa is 2 * A,
    nantou(Nta, N, Aa, P).
nantou(N, N, A, P):-
    A > P.

% volani pro potrebný pocet parametru
nantou2(M, N):-
    velikost(M, P),
    nantou(M, N, 1, P).

% spocita velikost matice, pro matici susednosti je to i pocet vrcholu
% velikost(+matice, -pocet prvku prvnio prvku)
velikost([[_|D]_|_], P):-
    velikost([D|_|_], P1),
    P is P1+1.
velikost([[]|_|_], 0).

% spocita velikost pole, pro matici susednosti umocnenou na cislo vetsi
% nez n, je toto cislo i pocet komponent souvislosti grafu
% velikostpole(+pole, -pocet prvku).
velikostpole([], 0).
velikostpole([_|S], V):-
    velikostpole(S, V1),
    V is V1 + 1.

% vytvori matici n krat n slozenou ze samych nul
% nulova(+rozmer, +kolik jeste radek, -matice).
nulova(_, 0, []).
nulova(N, K, [H|T]):-
    nuluj(N, H),
    Kk is K - 1,
    nulova(N, Kk, T), !.
```

```
% volani pro potrebný počet argumentu
% nulova2(+rozmer,-matice).
nulova2(N,M) :-nulova(N,N,M).

% vytvori vektor ze samych nul
% nuluj(+rozmer,-matice).
nuluj(0,[]).
nuluj(N,[0|D]) :-
    N1 is N-1,
    nuluj(N1,D),!.

% prida do matice jednický namisto hran grafu, aby vznikla matice
% susednosti
% pridej(+seznam,+matice,-matice).
pridej([],M,M).
pridej([[P,D]|T],M,N) :-
    pridejbod([P,D],M,M1),
    pridejbod([D,P],M1,M2),
    pridej(T,M2,N).

% prida do matice prave jednu hranu
% pridejbod(+bod,+matice,-matice).
pridejbod([1,B],[R|D],[Rv|D]) :-vlozdoradky(R,B,Rv).
pridejbod([A,B],[J|M],[J|N]) :-
    Aa is A - 1,
    pridejbod([Aa,B],M,N).

% vlozi do vektoru na zadanou pozici jednicku
% vlozdoradky(+radka,+pozice,-vystupni radka).
vlozdoradky([_|D],1,[1|D]).
vlozdoradky([A|D],B,[A|Da]) :-
    Bb is B - 1,
    vlozdoradky(D,Bb,Da).

% prida jednický na diagonalu
% pridejdia(+matice,+rozmer,-matice).
pridejdia(N,0,N).
pridejdia(M,P,N) :-
    pridejbod([P,P],M,Mm),
    Pp is P - 1,
    pridejdia(Mm,Pp,N).

% vytvori ze seznamu hran matici souvislosti
% vytvormatici(+seznam,+rozmer,-matice).
vytvormatici(S,N,M) :-
    nulova2(N,M1),
    pridejdia(M1,N,M2),
    pridej(S,M2,M).
```

### Přehled závislosti procedur na sobě



