



Fakulteta za
informacijske študije
Faculty of information studies

Univerza v Ljubljani
Fakulteta *za strojništvo*



Kreativno jedro:
Simulacije
Creative core: Simulations

Kampus šola HPC 2014

MaxClique

Algoritem za iskanje največje klike

Matjaž Depolli
Odsek za komunikacijske sisteme
Inštitut „Jožef Stefan“



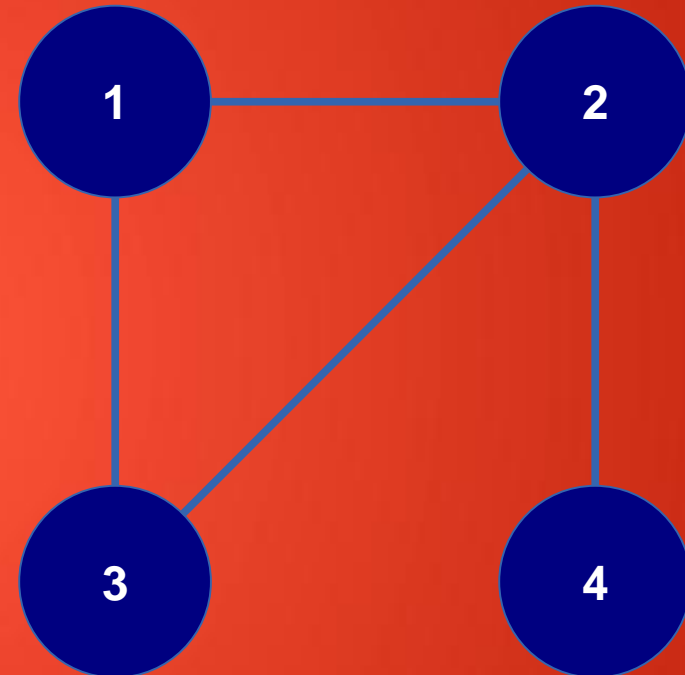
Naložba v vašo prihodnost
OPERACIJO DELNO FINANCIRA EVROPSKA UNIJA
Evropski sklad za regionalni razvoj



REPUBLIKA SLOVENIJA
**MINISTRSTVO ZA IZOBRAŽEVANJE,
ZNANOST IN ŠPORT**

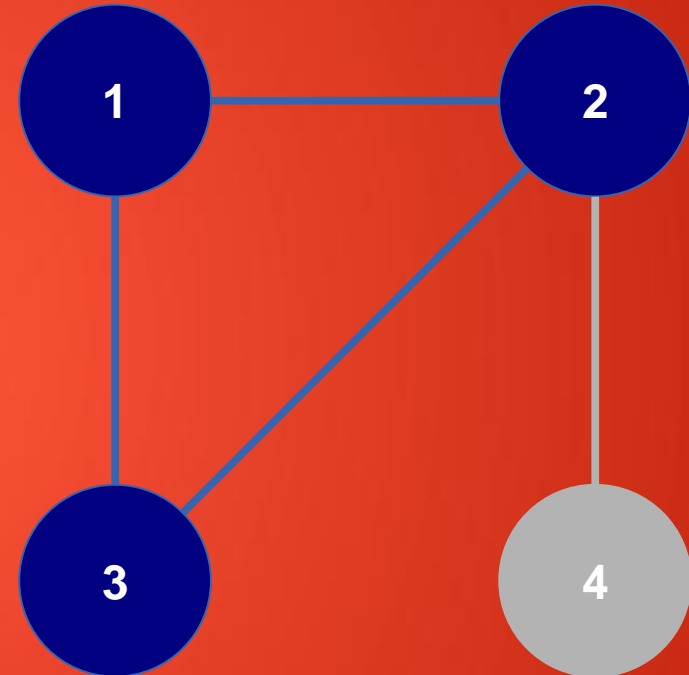
Klika

- Graf = $\langle V, E \rangle$
V so vozlišča (Vertices)
E so povezave (Edges)
- $V = \{1, 2, 3, 4\}$
- $E = \{(1,2), (1,3), (2,3), (2,4)\}$
- Klika = polno povezan graf
- Induciran podgraf =
vzamemo del vozlišč in vse
povezave med temi vozlišči



Največja klika

- Induciran podgraf nad vozlišči $\{1, 2, 3\}$
- Algoritem za iskanje največje klike poišče največji podgraf, ki je polno povezan
- Vedno najde le enega



Osnovni algoritem največje klike

- $\langle V_{\max}, E_{\max} \rangle$ je trenutno najdena največja klika
 - Na začetku sta V_{\max} in E_{\max} prazni množici
- Za vse možne V_1 , kjer:
 - $V_1 =$ podmnožica vozlišč V
 - $E_1 =$ iz V_1 inducirana množica robov E
- Preveri če je $\langle V_1, E_1 \rangle$ poln graf in $|V_1| > |V_{\max}|$
 - če ni, pojdi dalje
 - če je, potem $\langle V_{\max}, E_{\max} \rangle = \langle V_1, E_1 \rangle$

Kombinatorična eksplozija

- Potenčna množica $P(V)$ je množica vseh podmnožic V
- $|P(V)| = 2^{|V|}$
- Osnovni (naivni) algoritem največje klike mora pregledati vse podmnožice V , torej vse elemente $P(V)$
 - Njegova kompleksnost je torej vsaj $O(2^n)$; $n=|V|$
 - Če ima graf 100 vozlišč, potem mora algoritem preveriti $\sim 1 \times 10^{30}$ podgrafov
- NP-poln problem

Razveji in omeji algoritmi

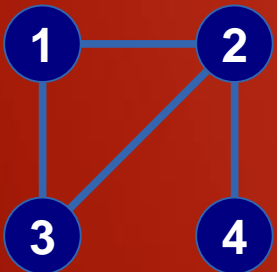
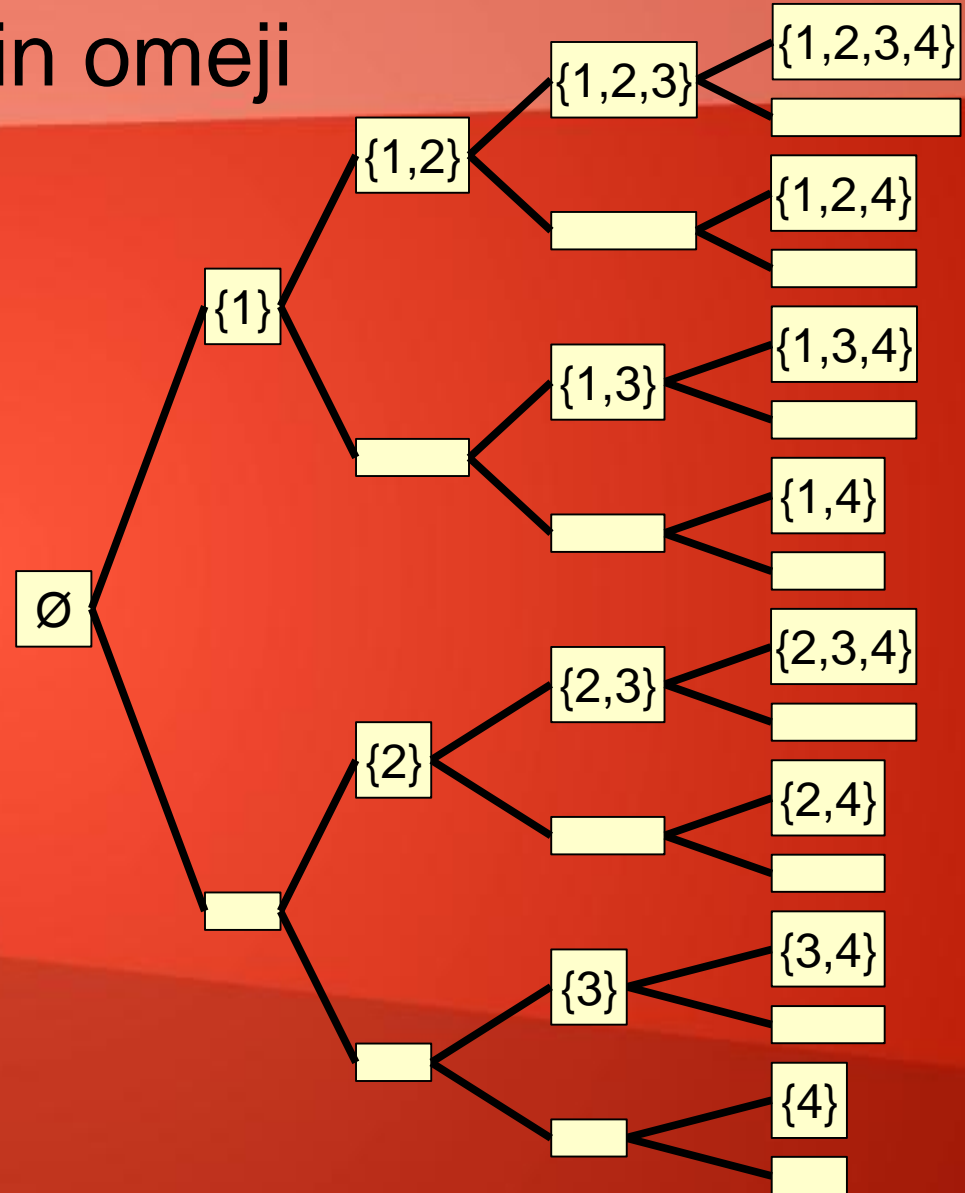
- 'Branch and bound'
- Algoritmi za obvladovanje kombinatorične eksplozije pri preiskovanju drevesa
- Z dodatnim znanjem se omeji področje iskanja
- Sestavljajo jih dve glavni funkciji
 - Razveji – trenutno stanje iskanja razcepi v več vej
Spremeni prostor iskanja v preiskovalno drevo
 - Omeji – veji določi meje iskane količine
omogoča rezanje vej in določanje vrstnega reda preiskovanja

Razveji in omeji največjo kliko

- Trenutno stanje:
 - Trenutna klika
 - Množica preostalih vozlišč
 - Največja do sedaj najdena klika
- Razveji:
 - Vzame prvo vozlišče iz množice preostalih vozlišč in naredi dve novi stanji:
 - Eno, kjer vozlišče doda trenutni kliki
 - Eno, kjer trenutna klika ostane enaka, vozlišče se le odstrani iz preiskave
- Omeji:
 - Če je velikost klike, ki lahko nastane iz trenutnega stanja + preostalih vozlišč \leq velikosti največje že najdene klike, potem zaustavi preiskovanje trenutne veje

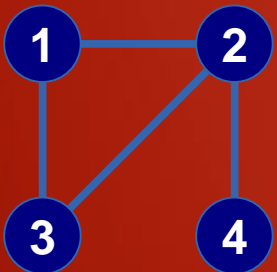
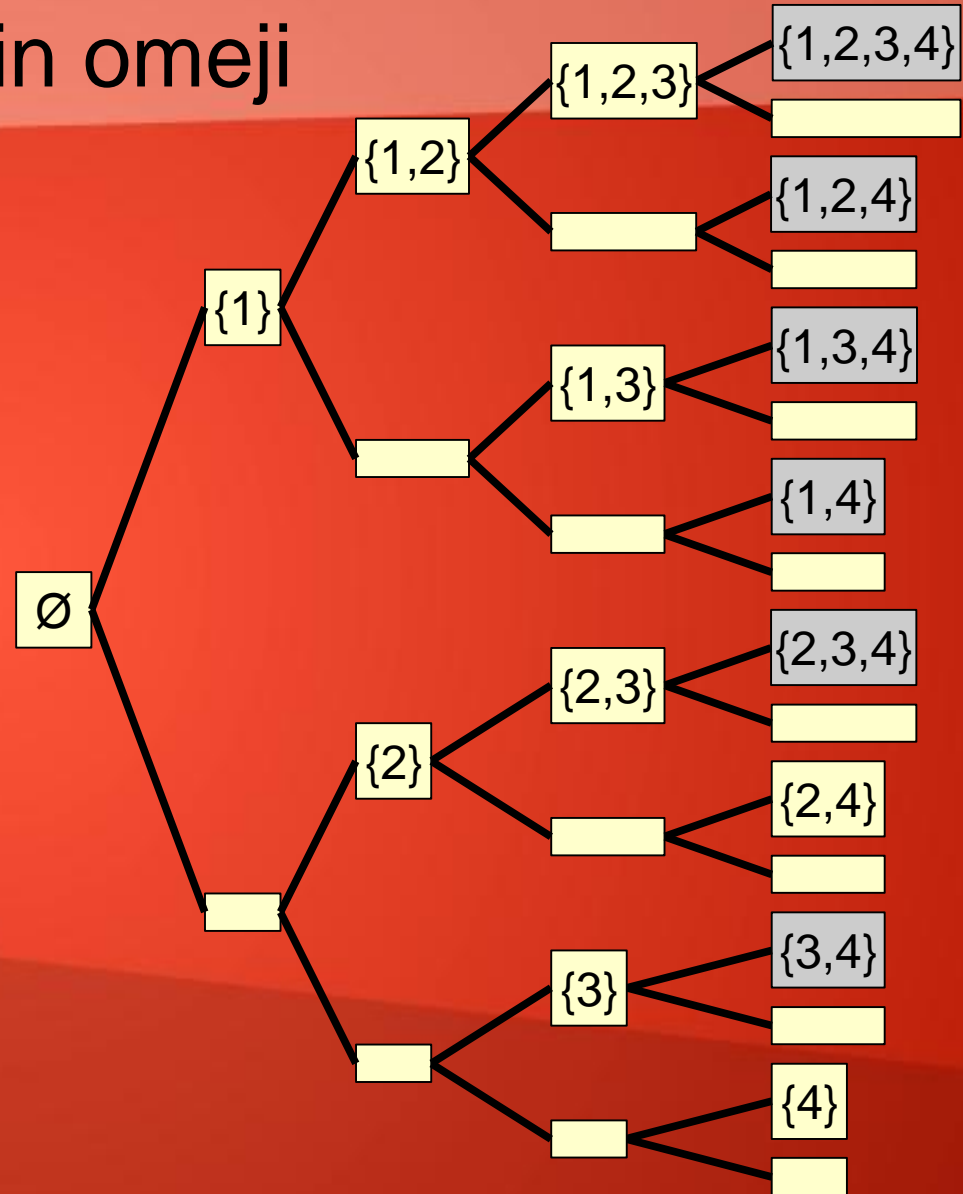
Razveji in omeji

- Osnovno preiskovalno drevo za algoritem največje klike
- Število korakov = 16
 $|V| = 4; 2^4 = 16$



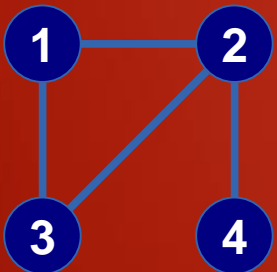
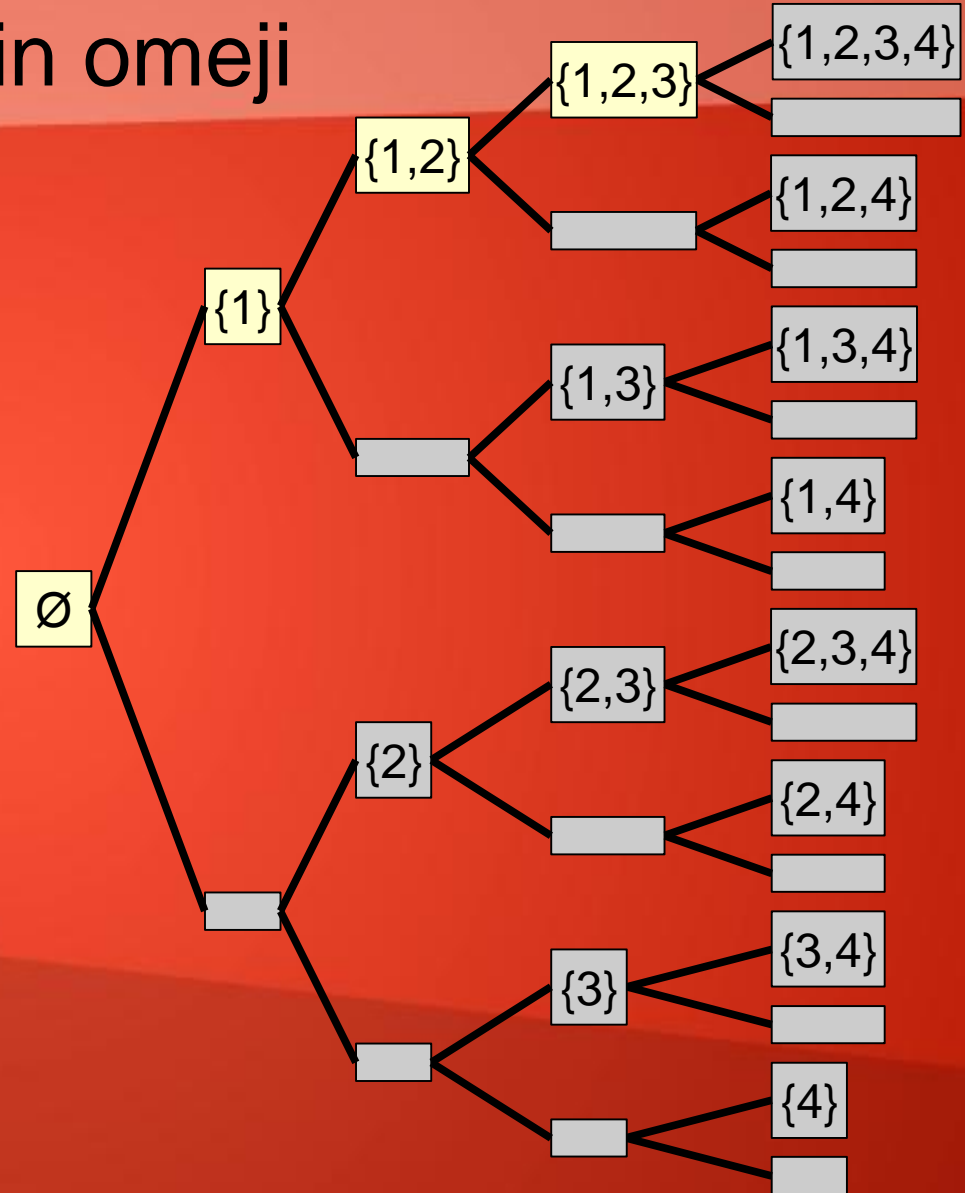
Razveji in omeji

- Uporabimo znanje o povezavah
- Če med trenutno množico vozlišč in naslednjim vozliščem ni povezave, potem se vejo odreže
- Število korakov = 10



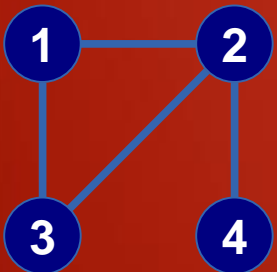
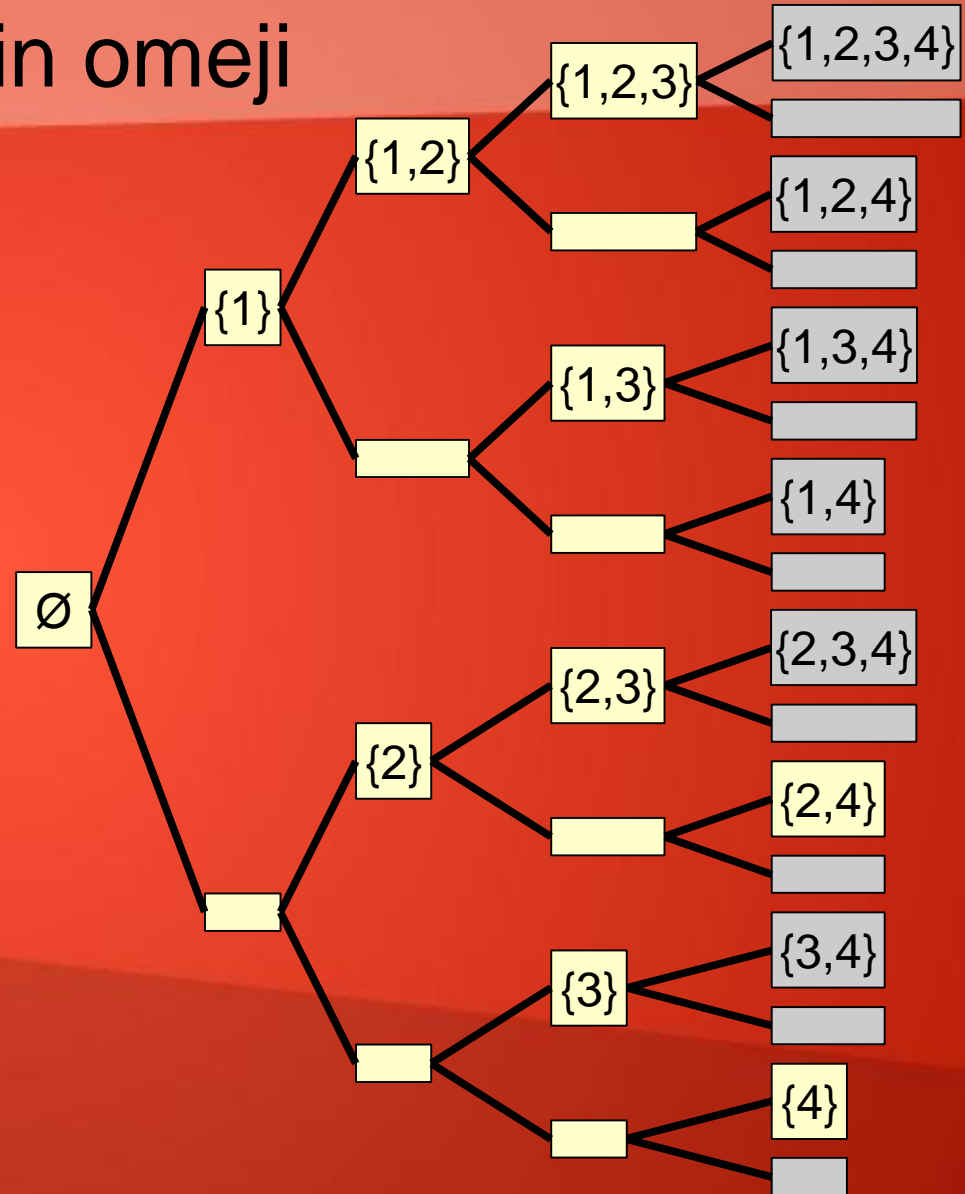
Razveji in omeji

- Uporabimo znanje o številu preostalih vozlišč in trenutni velikosti klike
- Če v veji ni možno najti večje klike od že najdene, se vejo odreže
- Število korakov = 4



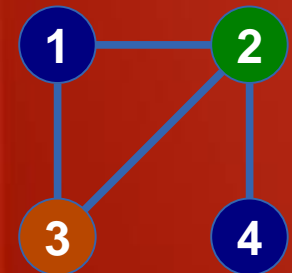
Razveji in omeji

- Veliko je možnosti za vrstni red preiskave
- Število korakov = 10



Dodamo kanček hevristik

- Vozlišča se pred začetkom sortira po njihovi stopnji
 - Stopnja vozlišča = število povezav na to vozlišče
 - Iz primera: $D_2=4$, $D_1=3$, $D_3=3$, $D_4=1$
- Izboljša se oceno o največji možni klikki trenutne veje z barvanjem vozlišč
 - Število barv predstavlja zgornjo mejo za velikost klike
 - Barvanje je sicer tudi NP-poln problem
 - Obstaja požrešni algoritem ki vrne približen (sub-optimalen) rezultat; časovna kompleksnost je $O(n^2)$



Prva vzporednost

- Množice vozlišč predstavimo kot bitne množice

$$\begin{aligned}\{\} &= [0000] \\ \{1\} &= [1000] \\ \{2\} &= [0100]\end{aligned}$$

$$\{\dots, 1, 2, 3, 4\} = [1111]$$

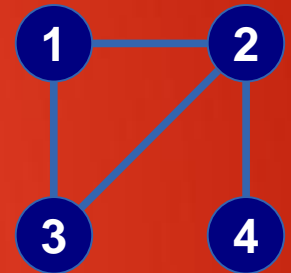
- Povezanost vozlišč predstavimo kot bitne množice

$$\begin{aligned}1: \{(1,2), (1,3)\} &: [0110] \\ 2: \{(1,2), (2,3)\} &: [1010]\end{aligned}$$

- Rezanje vej je izvedeno kot logični IN med povezanostjo dveh vozlišč:

$$[0110] \text{ in } [1010] = [0010]$$

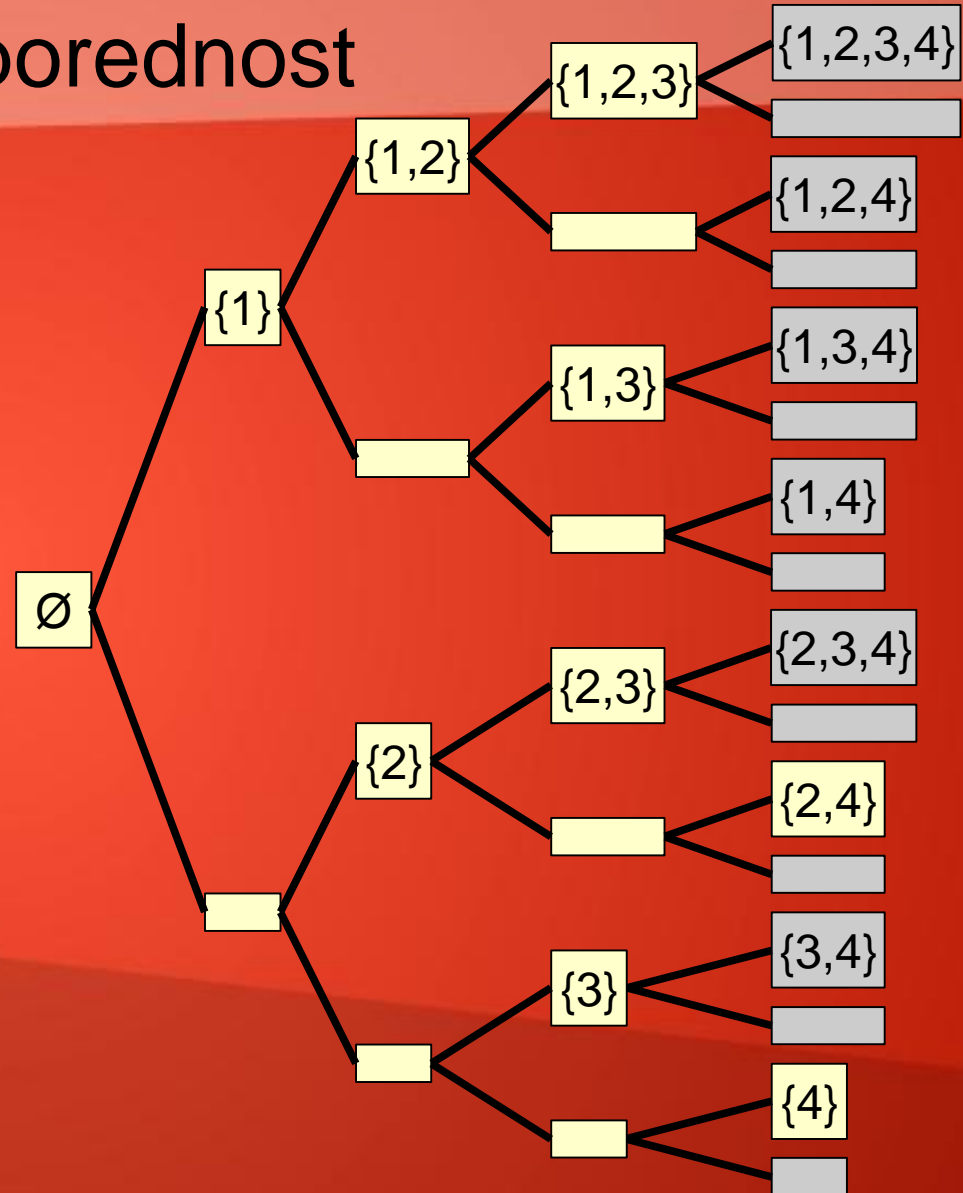
- Izkoriščamo strojne logične operacije nad 64-bitnimi operandi



Druga vzporednost

Paralelizacija z nitmi

- Prva nit je glavna in začne izvajanje
- Vsako 'polno' vejo dobi v pregled druga nit (do max št. niti)
- Ko niti zmanjka dela, si vzame novo vejo iz spiska še nedodeljenih polnih vej
- Komunikacija preko dveh globalnih spremenljivk
- Drevo mora biti veliko, da se taka delitev izplača



Niti, sinhronizacija

- Slabosti vzporednega algoritma
 - En proces mora prevzeti glavno vlogo in imeti pregled nad iskanjem
 - Globalna spremenljivka – velikost največje že najdene klike
 - Ena veja je lahko bolj zahtevna kot vse ostale skupaj
 - Nepredvidljivost
- Prednosti vzporednega algoritma
 - Več vzporedno preizkovanih vej lahko pozitivno spremeni potek preiskovanja
 - Zelo malo komunikacije

Učinkovitost algoritma

- Zelo malo sinhronizacije zaradi zaščitnih spremenljivk
- Avtomatski load-balancing
 - Vej je toliko kot vozlišč (razmeroma dobro razdrobljen problem)
 - Veje (poddrevesa) se deli od največjih do najmanjše



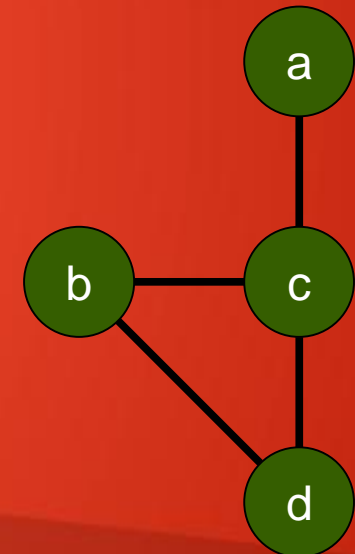
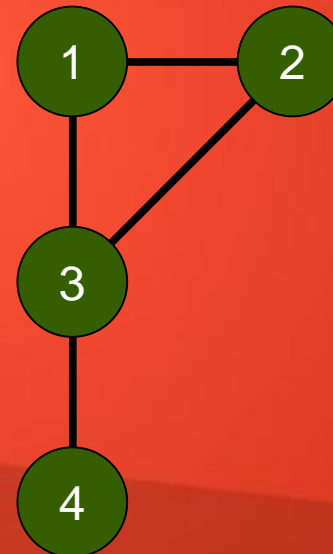
- Popolnoma neodvisni podproblemi
- Enostavnost

Primer uporabe

- Dva grafa
 - Npr: dve molekuli
- Ali sta grafa ekvivalentna, oziroma, kako podobna sta si?

4 == a
1 == b
(2 == b

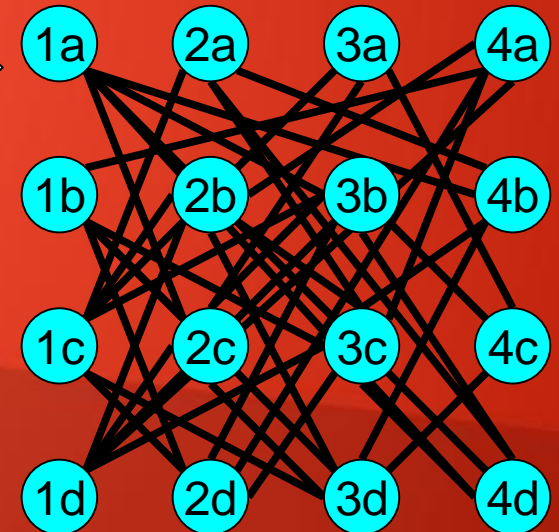
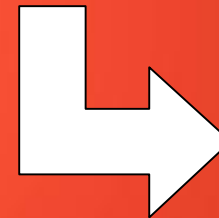
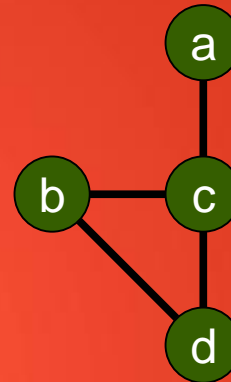
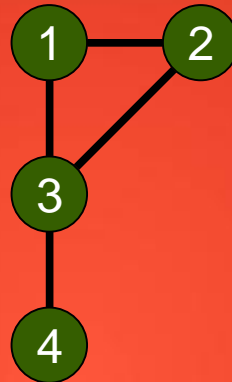
3 == c
2 == d
1 == d)



Primer uporabe

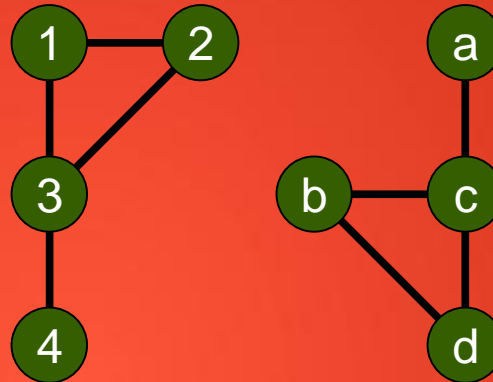
Produktni graf

- Vozlišča so kombinacije vozlišč vhodnih grafov
- Povezave so tam, kjer so ekvivalentno povezana vozlišča vhodnih grafov



Primer uporabe

Produktni graf



- Vsaka klika v produktnem grafu predstavlja ekvivalentne podgrafe
- Največja klika predstavlja najboljšo preslikavo med grafoma

