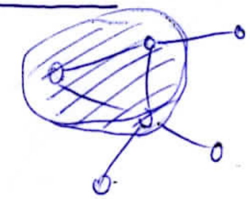
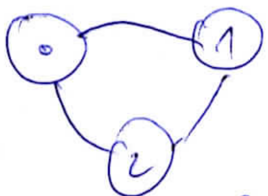
Problém kliky velikosti k v neorientovaném grafu



- klika: podgraf, který je úplným grafem
- problém kliky: Má daný graf G kliku o velikosti alespoň k?
- problém kliky lze charakterizovat jazykem

KLIKA = { <(G,k)> | G má kliku o velikosti alespoň k }.

- <.> reprezentuje vhodné kódování grafu a velikosti kliky
- graf můžeme kódovat např. jako posloupnost čísel uzlů v binárním kódu, následovaných posloupností dvojic čísel uzlů, mezi kterými existuje hrana. Vše vhodně odděleno.

Např.  lze kódovat jako: $\underbrace{00|01|10|(00,01)|(00,10)|(01,10)}_{n}$

- velikost kliky lze kódovat číslem v bin. kódu.

- KLIKA je NP-úplná. Ukažte!

I. Ukážeme, že KLIKA je v NP

Ukazuje-li, že P je C-úplný:

1. Ukázal P∈C

2. Ukázat P je C-těžký

Při rozhodování, zda $w \in$ KLIKA musíme postupovat takto:

1. Ověříme, zda w je platnou instancí problému klika:

   – kontrola správného použití oddělovačů – stačí jeden průchod vstupem a tedy složitost $O(n)$.

   – kontrola neexistence duplikovaných uzlů v každém grafu: Pro každý uzel na vstupu $(O(n))$, tento uzel přepíšeme na pomocnou pásku $(O(n))$ a projdeme ostatní uzly (těch je $O(n)$) a porovnáme je na shodu (test shody: $O(n)$). Celkově má tento krok složitost: $O(n) \cdot (O(n) + O(n) \cdot O(n)) = O(n^3)$.

   – kontrola neexistence duplikovaných hran (včetně možnosti záměny východzího a cílového uzlu) + kontrola, zda hrany nejsou nad neznámými uzly. Analogicky jako výše lze realizovat v polynomiálním čase s využitím pomocných pásek.

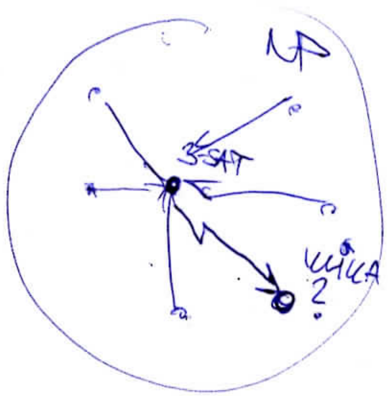
   – celkově je tedy ověření, zda w je platnou instancí problému kliky v polynomiálním čase.

2. Označíme náhodně k uzlů dodáním obvodné značky ke kódu uzlů: Lze realizovat tak, že na pomocnou pásku zapíšeme hodnotu k, tu následně postupně dekrementujeme a přitom procházíme seznam uzlů na vstupní pásce a při každém dekrementu jeden označíme. Provedeme O(n) dekrementací, přičemž dekrement je snadný v O(n) a průchod uzly na vstupní pásce je také v O(n). Dohromady je cena tohoto kroku evidentně polynomiální.

3. Ověříme, že v kroce 2 byla vybrána klika. To lze udělat tak, že postupně projdeme všechny uzly (O(n)) a pro každý označený uzel projdeme všechny ostatní uzly (O(n)) a pokud je takový jiný také označený, přepíšeme danou dvojici na pomocnou pásku. (O(n)). Následně projdeme seznam hran (O(n)) a podíváme se, zda existuje hrana odpovídající dané dvojici (O(n)). Celkově odpovídá složitost $O(n^4)$.

1. for each uzel
2. for each uzel'
   - přepiš
3. for each hrana
4. porovnej hrana a (uzel, uzel')

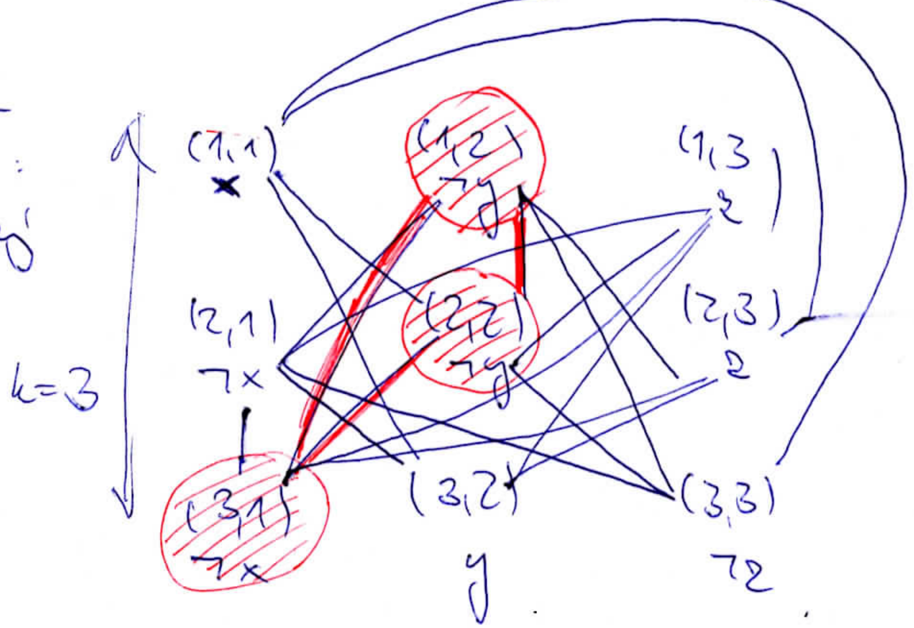
II. Ukážeme, že KLIKA je NP-těžká



- polynomiální redukce z problému 3-SAT
- 3-SAT: - máme $m \geq 1$ bool. proměnných $x_1, \dots, x_m$
  - literály L: $x_i / \neg x_i$ pro $1 \leq i \leq m$.
  - klauzule C: $L_{i_1} \vee L_{i_2} \vee L_{i_3}$ pro literály $L_{i_1}, L_{i_2}, L_{i_3}$
  - formule F: $C_1 \wedge \dots \wedge C_k$, $k \geq 1$,
- 3-SAT je známý NP-úplný problém.
- Princip redukce na příkladu: uvažme formuli

$$F = (x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee y \vee \neg z)$$

graf G_F přiřazený F



V G_F propojíme hranami uzly, které:
- si nekolidují
- jsou z různých klauzulí

- $F = C_1 \wedge \dots C_k$ je splnitelná $\iff$ G_F obsahuje kliku velikosti k.
- Ve výše uvedeném příkladě – klika je červeně vyznačená a odpovídá jí splňující přiřazení:
  $y = 0 \wedge x = 0 \wedge z = 0/1$ (z je libovolné)

- <u>Formálněji</u>:
  - Libovolnou danou instanci 3-SAT, tj. formuli $F = C_1 \wedge \dots \wedge C_k$ převedeme na ~~graf~~ graf $G_F = (V, E)$ následovně:

    $$V = \{ [\underset{\uparrow \text{klauzule}}{i}, \underset{\nwarrow \text{literál}}{j}] \mid 1 \leq i \leq k,\ 1 \leq j \leq 3 \}$$

    $$E = \{ \{ [i,j], [i',j'] \} \mid \underset{\uparrow \text{různé klauzule}}{i \neq i'} \wedge \underset{\uparrow \text{nekolidující literály}}{l_{ij} \neq \neg l_{i'j'}} \}$$

  - Tuto redukci lze evidentně implementovat úplným DTS pracujícím v polynomiálním čase:
    - lze užít podobné kódování grafů jako v bodě I, pro usnadnění dalších kroků lze každý uzel spojit

s kódem příslušného literálu (+/- číslo proměnné)

- Vygenerování seznamu uzlů znamená průchod vstupní páskou (O(n)) spojený s tím, že na pomocnou pásku přepisujeme jednotlivé nalezené literály (O(n)) a spojujeme je s čísly uzlů, které čteme z pomocné pásky, jejíž obsah je postupně inkrementován (čítač) (přepis čítače O(n), inkrementace (O(n))). Celkově O(n²)
- Seznam hran vygenerujeme tak, že projdeme seznam uzlů (O(n)) a ke každému možnému výchozímu uzlu projdeme všechny případné cílové (O(n)), porovnáme kódy literálů (O(n)) a pokud nekolidují a uzly jsou z jiných klauzulí, vypíšeme hranu (O(n)) - celková složitost O(n³)
- ke kódu grafu ještě přidáme k jako velikost kliky (k je počet klauzulí, který lze snadno inkrementací spočítat na pomocné pásce).

- Zbývá ukázat, že $F = C_1 \wedge \dots \wedge C_k$ je splnitelná
$\Longleftrightarrow$ G_F má kliku velikosti k.

a) "$\Rightarrow$"

- Je-li F splnitelná, lze v každé klausuli C_i zvolit jeden literál, který může splnit, aniž by to mělo vliv na ostatní zvolené literály.
- Vzhledem k splněným literálům odpovídá k uzlů v G. Tyto uzly jsou všechny propojeny, protože (a) jsou z jiných klausulí a (b) jsou označeny nekolidujícími literály.
- G_F tedy má kliku velikosti k.

b) "$\Leftarrow$"

- Je-li v G_F klika velikosti k, je tvořena množinou uzlů obsahující jeden uzel z každé skupiny uzlů odpovídajících nějaké klausuli.
- Splňující ohodnocení pak lze získat tak, že splníme literály spojené s těmito uzly. To je možné, protože tyto literály nekolidují.

Současně splníme literál z každé klauzule a tedy i celou formuli.

$\square$

- Ukažte, že třída P je uzavřena vůči sjednocení.

Důkaz (idea):

- Uvažme libovolné jazyky $L_1, L_2 \in P$
- Víme, že L_1, L_2 lze přijímat DTS v polynomiálním čase.
- Současně víme, že ke každému DTS, který přijímá nějaký jazyk v polynom. čase, lze sestrojit DTS, který tento jazyk rozhoduje v polynom. čase.
- Nechť tedy M_1 a M_2 jsou DTS rozhodující L_1 a L_2 v polynom. čase.
- Sestrojíme DTS M, který rozhoduje $L_1 \cup L_2$.
- M:
  1. zkopíruje vstupní pásku na pomocnou pásku.
  2. Na vstupní pásce odsimuluje M_1. Pokud M_1 přijme, M přijme.
  3. Jinak na 2. pásce odsimuluje M_2. Pokud přijme, přijme, jinak odmítne.

- Pokud $P_1(n)$ a $P_2(n)$ jsou polynomy, ve kterých M_1 a M_2 rozhodují L_1 a L_2, pak M pracuje v čase $O(n) + P_1(n) + P_2(n) \in O(n^k)$ pro $k \geq 1$.
  ↑ kopie pásky + režie (předávání řízení) □

- Analogicky lze ukázat uzavřenost P vůči $\cap$.
- <u>Ukažte (stačí idea), že P je uzavřena vůči konkatenaci.</u>

<u>Důkaz (idea)</u>:

- Uvažme libovolné $L_1, L_2 \in$ P. K L_1, L_2 víme, že existují DTS M_1, M_2, které L_1, L_2 rozhodují v polynomiálním čase $P_1(n), P_2(n)$.
- Jazyk $L_1 \cdot L_2$ lze pomocí DTS M rozhodovat tak, že pro vstupní řetězec $w = a_1 a_2 \cdots a_n$ uvážíme $n+1$ možných rozdělení na řetězce w_1 a w_2 takové, že $w = w_1 \cdot w_2$: $w_1 = \varepsilon$ a $w_2 = w$, $w_1 = a_1$ a $w_2 = a_2 \cdots a_n$, $\ldots$, $w_1 = a_1 \cdots a_{n-1}$ a $w_2 = a_n$, $w_1 = w$ a $w_2 = \varepsilon$.

Pro každou dvojici w_1 a w_2 se pomocí M_1 a M_2 ověří, zda $w_1 \in L_1$ a $w_2 \in L_2$. M pak odpoví ano, ~~pouze~~ právě tehdy pokud taková dvojice existuje.

- M pracuje v čase $(n+1)\cdot\left(P_1(n) + P_2(n) + O(n)\right) \in O(n^k)$

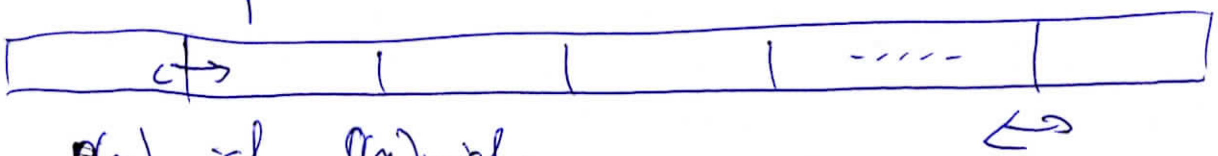
  $k \geq 1$.

  věci spojené s kopírováním, mazáním

$\square$

- Ukažte (stačí idea) uzavřenost P vůči $*$.

  - Pozor, nelze řešit tak, že vyzkoušíme rozdělení na libovolný počet podřetězců končících na libovolném místě. Vede na exponenciální složitost



$O(n)$ míst, kde může končit 1. podřetězec

$O(n)$ míst, kde končí 2. podřetězec

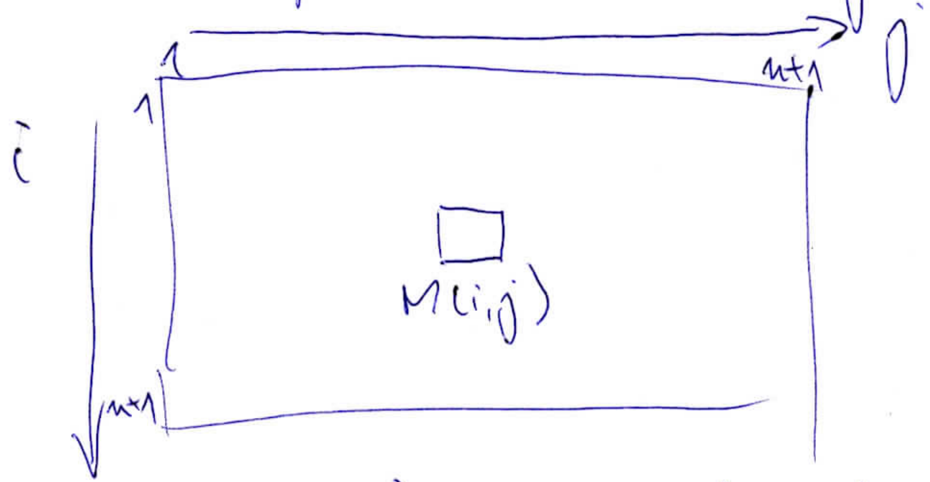
....

$O(n)$ podřetězců:

$$\underbrace{O(n)\cdot O(n)\cdot\ldots\cdot O(n)}_{O(n)\times\ \text{krát}} = O(n)^{O(n)}$$

[illegible]:

- Lze ale užít dynamické programování (cache) a sestavit pro řetězec délky n matici $(n+1)\times(n+1)$:



Význam $M(i,j) \in \{0,1\}$ bude takový, že
$M(i,j) = 1$, jestliže řetězec $a_i a_{i+1} \dots a_{j-1} \in L$, kde L je jazyk, jehož iteraci testujeme. Jinak $M(i,j) = 0$.
K sestavení této matice stačí čas $(n+1)^2 \cdot P(n)$, kde $P(n)$ je polynom charakterizující složitost, ve které jsme schopni pomocí DTS rozhodovat L.

- M reprezentuje relaci $R \subseteq \{1,\dots,n+1\} \times \{1,\dots,n+1\}$.
- Lze v čase ~~Warsh.~~ $O(n^3)$ Warsh. alg. spočíst relaci R^+ reprezentovanou maticí M^+.
- Otestujeme, zda platí $R^+(1,n+1)$, pokud ano přijmeme,

jej odmítneme. Tímto způsobem v polynomiálním čase rozhodujeme $L^\#$. $\square$

- <u>Uzavřenost NP vůči $\cup, \cap, \cdot, {}^*$:</u>

  Lze postupovat analogicky s tím, že je možno u $\cdot$ a * uhádnout rozdělení na podřetězce.