4.4.49 Silně $\mathcal{NP}$-úplné úlohy. Ne všechny $\mathcal{NP}$-úplné úlohy mají stejné vlastnosti. Ukážeme si, že některé zůstávají $\mathcal{NP}$-úplné i když se omezíme jen na ty, které danou konstantu k mají pouze polynomiálně velkou vzhledem k velikosti zbylého vstupu, některé tuto vlastnost nemají. Ukažme si jeden příklad.

4.4.50 Příklad – celočíselný problém batohu *Úloha.* Jsou dána kladná přirozená čísla $a_1, a_2, \ldots, a_n$ a K.

Otázka. Existují celá čísla $x_1, x_2, \ldots, x_n$ taková, že $x_i \geq 0$ pro každé i a platí

$$\sum_{i=1}^{n} x_i\, a_i = K.$$

Existuje algoritmus, který řeší celeočíselný problém batohu v čase $\mathcal{O}(nK)$.

Vytvoříme graf orientovaný $G = (V, E)$, kde $V = \{0.1. \ldots, K\}$ a (i, j) je hrana právě tehdy, když $i < j$ a $j - i \in \{a_1, a_2, \ldots, a_n\}$.

Pak instance celočíselného problému batohu je ANO instance právě tehdy, když v grafu G existuje orientovaná cesta z vrcholu 0 do vrcholu K.

Přitom zjistit, zda v grafu G existuje orientovaná cesta z 0 do K, je možné v čase $\mathcal{O}(nK)$.

Uvědomte si, že toto není obecně polynomiální algoritmus, protože číslo K může být i exponenciální vůči číslu n, což je velikost instance. Zavedeme proto kromě velikosti instance, ještě číslo instance.

Číslo instance I, značíme ho $num(I)$, je největší číslo, které se v dané instanci vyskytuje.

4.4.51 Silně NP-úplné úlohy.

Definice. Úloha $\mathcal{U}$ je **silně $\mathcal{NP}$-úplná**, jestliže existuje polynom $p(n)$ pro který je úloha, omezíme-li se pouze na instance I s $num(I) \leq p(n)$, stále $\mathcal{NP}$-úplná.

- Problém klik, problém existence hamiltonovské kružnice/cyklu, TSP jsou silně $\mathcal{NP}$-úplné úlohy.
- Subsetsum, problém batohu, dělení kořisti nejsou silně $\mathcal{NP}$-úplné úlohy.

4.4.52 Pseudopolynomíální algoritmus

Definice. Algoritmus $\mathcal{A}$ je **pseudopolynomiální**, jestliže existuje polynom p takový, že $\mathcal{A}$ řeší úlohu v čase $\mathcal{O}(p(n, num(I))$.

4.4.53 Tvrzení. Kdyby pro některou silně $\mathcal{NP}$-úplnou úlohu existoval pseudopolynomiální algoritmus, který ji řeší, tak $\mathcal{P} = \mathcal{NP}$.

4.5 Heuristiky.

Jestliže je třeba řešit problém, který je $\mathcal{NP}$ úplný, musíme pro větší instance opustit myšlenku přesného nebo optimálního řešení a smířit se s tím, že získáme „dostatečně přesné“ nebo „dostatečně kvalitní“ řešení. K tomu se používají heuristické algoritmy pracující v polynomiálním čase. Algoritmům, kde umíme zaručit „jak daleko“ je nalezené řešení od optimálního, se také říká aproximační algoritmy.

4.5.1 Heuristika pro vrcholové pokrytí — 1. Uvažujme následující heuristický algoritmus, který pro daný neorientovaný graf najde jeho vrcholové pokrytí. Algoritmus je založen na „hladovém postupu“.

Vstup: neorientovaný graf $G = (V, E)$.
Výstup: vrcholové pokryti C grafu G.

```
begin
    C := ∅
        while E ≠ ∅ do
            vyber vrchol v s největším stupněm
            C := C ∪ {v}
            odstraň v spolu s hranami s ním incidentními
end
return C
```

Přestože algoritmus „vypadá rozumně“, v některých případech najde vrcholové pokrytí, které má podstatně víc vrcholů než nejméně početné pokrytí. Zde tím „podstatně“ rozumíme toto: existuje graf G, který má vrcholové pokrytí o k vrcholech, ale výše uvedený algoritmus najde vrcholové pokrytí o $\Theta(k \lg k)$ vrcholech.

4.5.2 Heuristika pro vrcholové pokrytí — 2. Uvažujme ještě jeden heuristický algoritmus, který pro daný neorientovaný graf najde jeho vrcholové pokrytí.

Vstup: neorientovaný graf $G = (V, E)$.
Výstup: vrcholové pokryti C grafu G.

```
begin
    C := ∅
        while E ≠ ∅ do
            vyber hranu {u, v}
            C := C ∪ {u, v}
            odstraň vrcholy u, v spolu se všemi hranami s nimi incidentními
end
return C
```

4.5.3 Tvrzení. Označme C_{min} nejméně početné vrcholové pokrytí grafu G. Pak druhá heuristika najde vrcholové pokrytí C takové, že

$$|C| \leq 2\,|C_{min}|.$$

Zdůvodnění. Označme F množinu všech hran, která byla vybrána algoritmem 4.5.2. Pak $|C| = 2\,|F|$. Ano, za každou vybranou hranu jsme do množiny C vložili dva vrcholy — krajní vrcholy této hrany. Navíc, žádné dvě hrany v množině F nemají společný vrchol; tudíž pro jejich pokrytí je třeba $|F|$ vrcholů. Proto $|C_{min}| \geq |F|$ a $|C| = 2\,|F| \leq 2\,|C_{min}|$.

4.5.4 Aproximační algoritmus.

Definice. Uvažujme optimalizační problém $\mathcal{U}$. Polynomiální algoritmus $\mathcal{A}$ se nazývá R *aproximační algoritmus*, jestliže existuje reálné číslo R takové, že pro každou instanci algoritmus $\mathcal{A}$ najde přípustné řešení ne horší než R krát hodnota optimálního řešení.

To znamená, že pro minimalizační úlohu najde řešení, které nemá hodnotu účelové funkce větší než R krát hodnotu optimálního řešení; pro maximalizační úlohu najde řešení, které nemá hodnotu účelové funkce menší než R krát hodnotu optimálního řešení.

Druhá heuristika pro nalezení vrcholového pokrytí je tedy 2 aproximační algoritmus pro problém vrcholového pokrytí.

Ne pro všechny úlohy, jejichž rozhodovací verze jsou NP úplné, aproximační algoritmy existují. Příkladem je problém obchodního cestujícího, jak ukazuje následující tvrzení.

4.5.5 Tvrzení. Kdyby existovala konstanta R a polynomiální algoritmus $\mathcal{A}$ takový, že pro každou instanci obchodního cestujícího I najde trasu délky $D \leq R\,OPT(I)$, kde $OPT(I)$ je délka optimální trasy instance I, pak

$$\mathcal{P} = \mathcal{NP}.$$

4.5.6 Zdůvodnění tvrzení 4.5.5. Za předpokladu tvrzení 4.5.5 bychom uměli polynomiálně vyřešit problém existence hamiltonovské kružnice. Naznačíme odpovídající převod.

Je dán neorientovaný graf $G = (V, E)$, $V = \{1, 2, \ldots, n\}$, a ptáme se, zda v něm existuje hamiltonovská kružnice. Zkonstruujeme instanci obchodního cestujícího takto: Pro města $\{1, 2, \ldots, n\}$ položíme

$$d(i,j) = \begin{cases} 1, & \{i,j\} \in E \\ R\,n + 1, & \{i,j\} \notin E \end{cases}$$

Trasa v instanci popsané výše může mít délku n, jestliže je tvořena všemi hranami délky 1. V tomto případě jsou všechny hrany hranami grafu G a trasa představuje hamiltonovskou kružnici. Nebo musí trasa mít délku alespoň $n - 1 + n\,R + 1 = n\,R + n = n\,(R + 11)$. To je v případě, že aspoň jedna spojnice v trase není tvořena hranou grafu G.

Tedy jestliže algoritmus $\mathcal{A}$ najde trasu délky jiné než n, pak v grafu G neexistuje hamiltonovská kružnice. Takto bychom polynomiálním algoritmem byli schopni rozhodnout existenci hamiltonovské kružnice. Protože existence hamiltonovské kružnice je $\mathcal{NP}$ úplný problém, platilo by $\mathcal{P} = \mathcal{NP}$.

4.5.7 Metrická instance TSP. Řekneme, že instance obchodního cestujícího je *metrická*, jestliže pro každá tři města i, j, k platí:

$$d(i,j) \leq d(i,k) + d(k,j).$$

4.5.8 Tvrzení. Jestliže instance I obchodního cestujícího je metrická, pak existuje polynomiální algoritmus $\mathcal{A}$, který pro I najde trasu délky D, kde $D \leq 2\,OPT(I)$. ($OPT(I)$ je délka optimální trasy v I.)

4.5.9 Slovní popis algoritmu z tvrzení 4.5.8. Instanci I považujeme za úplný graf G s množinou vrcholů $V = \{1, 2, \ldots, n\}$ a ohodnocením d.

1. V grafu G najdeme minimální kostru (V, K).
2. Kostru (V, K) prohledáme do hloubky z libovolného vrcholu.
3. Trasu T vytvoříme tak, že vrcholy procházíme ve stejném pořadí jako při prvním navštívení během prohledávání grafu. T je výstupem algoritmu.

Zřejmě platí, že délka kostry K je menší než $OPT(I)$. Ano, vynecháme-li z optimální trasy některou hranu, dostaneme kostru grafu G. Protože K je minimální kostra, musí být délka K menší než $OPT(I)$ (předpokládáme, že vzdálenosti měst jsou kladné). Vzhledem k tomu. že instance je metrická, je délka T menší nebo rovna dvojnásobku délky kostry K.

4.5.10 Christofidesův algoritmus. Jestliže instance I obchodního cestujícího je metrická, pak následující algoritmus najde trasu T délky D takovou, že $D \leq \frac{3}{2}\,OPT(I)$.

Instanci I považujeme ze úplný graf G s množinou vrcholů $V = \{1, 2, \ldots, n\}$ a ohodnocením d.

1. V grafu G najdeme minimální kostru (V, K).
2. Vytvoříme úplný graf H na množině všech vrcholů, které v kostře (V, K) mají lichý stupeň.
3. V grafu H najdeme nejlevnější perfektní párování P.
4. Hrany P přidáme k hranám K minimální kostry. Graf $(V, P \cup K)$ je eulerovský graf. V grafu $(V, P \cup K)$ sestrojíme uzavřený eulerovský tah.
5. Trasu T získáme z eulerovského tahu tak, že vrcholy navštívíme v pořadí, ve kterém jsme do nich poprvé vstoupili při tvorbě eulerovského tahu.

Platí, že délka takto vzniklé trasy je maximálně $\frac{3}{2}$ krát větší než délka optimální trasy.

4.5.11 Poznámka. Odhad délky trasy, kterou jsme získali v 4.5.9, i odhad pro trasu získanou Christofidesovým algoritmem není možné zlepšit.

4.6 Třída co-$\mathcal{NP}$

4.6.1 Pozorování. Je-li jazyk L ve třídě $\mathcal{P}$, pak i jeho doplněk $\overline{L}$ patří do třídy $\mathcal{P}$. Obdobné tvrzení se pro jazyky třídy $\mathcal{NP}$ neumí dokázat.

4.6.2 Definice. Jazyk L patří do třídy co-$\mathcal{NP}$, jestliže jeho doplněk patří do třídy $\mathcal{NP}$.

4.6.3 Příklady.

- Jazyk $USAT$, který je doplňkem jazyka SAT splnitelných booleovských formulí, leží ve třídě co-$\mathcal{NP}$. (Jazyk $USAT$ se skládá ze všech nesplnitelných booleovských formulí a ze všech slov, které neodpovídají booleovské formuli.)
- Jazyk $TAUT$, který se skládá ze všech slov odpovídajících tautologii výrokové logiky, patří do třídy co-$\mathcal{NP}$.

4.6.4 Otázka, zda co-$\mathcal{NP} = \mathcal{NP}$, je otevřená.

4.6.5 Lemma. Mějme dva jazyky L_1 a L_2, pro které platí $L_1 \lhd_p L_2$. Pak platí také $\overline{L_1} \lhd_p \overline{L_2}$, (kde $\overline{L}$ je doplněk jazyka L).

Zdůvodnění. Jestliže $L_1 \lhd_p L_2$, $L_1 \subseteq \Sigma^\star$, $L_2 \subseteq \Pi^\star$, pak existuje polynomiální algoritmus $\mathcal{A}$, který pro každé slovo $w \in \Sigma^\star$ zkonstruuje slovo $\mathcal{A}(w) \in \Pi^\star$ a to tak, že

$$w \in L_1 \quad \text{právě tehdy, když} \quad \mathcal{A}(w) \in L_2.$$

To ale znamená, že

$$w \notin L_1 \quad \text{právě tehdy, když} \quad \mathcal{A}(w) \notin L_2,$$

a tedy $\overline{L_1} \lhd_p \overline{L_2}$.

4.6.6 Tvrzení. Platí co-$\mathcal{NP} = \mathcal{NP}$ právě tehdy, když existuje $\mathcal{NP}$ úplný jazyk, jehož doplněk je ve třídě $\mathcal{NP}$.

Zdůvodnění. Jestliže co-$\mathcal{NP} = \mathcal{NP}$, pak každý doplněk nějakého $\mathcal{NP}$ úplného jazyka leží ve třídě $\mathcal{NP}$.

Předpokládejme, že existuje $\mathcal{NP}$ úplný jazuk L, jehož doplněk $\overline{L}$ leží ve třídě $\mathcal{NP}$. Ukážeme, že co-$\mathcal{NP} \subseteq \mathcal{NP}$ a $\mathcal{NP} \subseteq$ co-$\mathcal{NP}$.

Vezměme libovolný jazyk L_1 ze třídy co-$\mathcal{NP}$. Pak $\overline{L_1} \in \mathcal{NP}$. Protože L je $\mathcal{NP}$ úplný jazyk, $\overline{L_1} \lhd_p L$ a podle předchozího lemmatu $L_1 \lhd_p \overline{L}$, kde $\overline{L} \in \mathcal{NP}$. Odtud $L_1 \in \mathcal{NP}$.

Vezměme libovolný jazyk L_2 takový, že $L_2 \in \mathcal{NP}$. Pak $L_2 \lhd_p L$ a tedy (opět podle předchozího lemmatu) $\overline{L_2} \lhd_p \overline{L}$. Protože $\overline{L} \in \mathcal{NP}$, je $\overline{L_2} \in \mathcal{NP}$ a $L_2 \in$ co-$\mathcal{NP}$.

4.7 Třídy $\mathcal{P}$SPACE a $\mathcal{NP}$SPACE

4.7.1 Je dán Turingův stroj M (deterministický nebo nedeterministický). Připomeňme, že M pracuje s paměťovou složitostí $p(n)$ právě tehdy, když pro každé slovo délky n nepoužije paměťovou buňku větší než $p(n)$. Uvědomte si: jestliže Turingův stroj přijímá jazyk s polynomiální časovou složitostí, pak se musí zastavit na **každém** vstupu (ať už leží v přijímaném jazyce, nebo ne); tj. Turingův stroj jazyk rozhoduje. Podobné tvrzení neplatí pro Turingův stroj,

který nějaký jazyk přijímá s polynomiální paměťovou složitostí; ano, Turingův stroj se může zacyklit na slově, které nepřijímá. Neterministický Turingův stroj se může zacyklit i na slově, které v jazyce $L(M)$ leží.

4.7.2 Třída $\mathcal{P}$SPACE. Jazyk L patří do třídy $\mathcal{P}$SPACE jestliže existuje deterministický Turingův stroj M, který přijímá jazyk L a pracuje s polynomiální paměťovou složitostí.

4.7.3 Tvrzení. Platí

$$\mathcal{P} \subseteq \mathcal{P}\text{SPACE}.$$

4.7.4 Třída $\mathcal{NP}$SPACE. Jazyk L patří do třídy $\mathcal{NP}$SPACE jestliže existuje nedeterministický Turingův stroj M, který přijímá jazyk L a pracuje s polynomiální paměťovou složitostí.

4.7.5 Tvrzení. Platí

$$\mathcal{NP} \subseteq \mathcal{NP}\text{SPACE}.$$

4.7.6 Věta. Je dán Turingův stroj M (deterministický nebo nedeterministický), který přijímá jazyk L s paměťovou složitostí $p(n)$ (kde p je nějaký polynom). Pak existuje konstanta c taková, že M přijme slovo w délky n po nejvýše $c^{p(n)+1}$ krocích.

4.7.7 Myšlenka důkazu věty 4.7.6. Konstantu c volíme tak, abychom měli zajištěno, že Turingův stroj M má při práci se vstupem délky n méně než $c^{p(n)+1}$ různých situací (ID). Zajímají nás totiž pouze takové výpočty, ve kterých se situace (ID) neopakují.

Označme t počet páskových symbolů Turingova stroje M a označme s počet stavů M. Pak M má $p(n)\, s\, t^{p(n)}$ různých situací. Ano, stroj se může nacházet v s různých stavech, hlava může skenovat jedno z $p(n)$ polí pásky, a páska může mít jeden z $t^{p(n)}$ různých obsahů.

Položme $c = t + s$. Z binomické věty vyplývá, že

$$c^{p(n)+1} = (t+s)^{p(n)+1} = t^{p(n)+1} + (p(n)+1)\, t^{p(n)}\, s + \dots.$$

Odtud $c^{p(n)+1} > p(n)\, t^{p(n)}\, s$.