1.5.20 Připomeňme z minulé přednášky, že:

1. Časová složitost deterministického Turingova stroje M je parciální zobrazení $T(n)$ z množiny všech přirozených čísel do sebe definované:

   Jestliže pro nějaké vstupní slovo délky n se Turingův stroje nezastaví, $T(n)$ není definováno. V opačném případě je $T(n)$ rovno maximálnímu počtu kroků, po nichž dojde k zastavení Turingova stroje, kde maximum se bere přes všechny vstupy délky n.

2. Paměťová složitost deterministického Turingova stroje M je parciální zobrazení $S(n)$ z množiny všech přirozených čísel do sebe definované:

   Jestliže pro nějaké vstupní slovo délky n se Turingův stroje nezastaví, $S(n)$ není definováno. V opačném případě je $S(n)$ rovno největšímu rozdílu pořadových čísel buněk, které byly během výpočtu použity než došlo k zastavení Turingova stroje, kde maximum se bere přes všechny vstupy délky n.

Tyto pojmy nyní rozšíříme i na nedeterministické Turingovy stroje.

1.5.21 Časová složitost nedeterministického Turingova stroje M je parciální zobrazení $T(n)$ z množiny všech přirozených čísel do sebe definované:

Jestliže pro nějaký výpočet nad některým vstupem délky n se M nezastaví, $T(n)$ není definováno. V opačném případě je $T(n)$ rovno maximálnímu počtu kroků, po nichž dojde k zastavení M, kde maximum se bere přes všechny vstupy délky n a všechny výpočty nad nimi.

1.5.22 Paměťová složitost nedeterministického Turingova stroje M je parciální zobrazení $S(n)$ z množiny všech přirozených čísel do sebe definované:

Jestliže pro nějaký vstup délky n a nějaký jeho výpočet se M nezastaví, $S(n)$ není definováno. V opačném případě je $S(n)$ rovno největšímu rozdílu pořadových čísel buněk, které byly během některého výpočtu použity než došlo k zastavení M, kde maximum se bere přes všechny vstupy délky n a všechny výpočty nad nimi.

1.5.23 Pozorování. Každý deterministický Turingův stroj $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ můžeme považovat za nedeterministický — je-li $\delta(q, X)$ definováno, ztotožníme ho s jednoprvkovou množinou $\{\delta(q, X)\}$.

1.5.24 Věta. Ke každému nedeterministickému Turingovu stroji M existuje deterministický Turingův stroj N takový, že přijímají stejné jazyky; tj.

$$L(M) = L(N).$$

Důkaz této věty spočívá v tom, že stylem „prohledávání do šírky“ deterministický Turingův stroj N prohledává všechny možné výpočty nedeterministického Turingova stroje M. Jestliže existuje přijímající výpočet nedeterministického Turingova stroje M nad slovem w, deterministický Turingův stroj N na něj narazí; jestliže přijímající výpočet neexistuje, deterministcký Turingův stroj N slovo w nepřijme.

Poznamenejme, že zkonstruovaný deterministický Turingův stroj N může potřebovat exponencielně více kroků než původní nedeterministický Turingův stroj M.

1.6 Rozhodovací úlohy

1.6.1 Teorie složitosti pracuje zejména s tzv *rozhodovacími* úlohami. Rozhodovací úlohy jsou takové úlohy, jejichž „řešením“ je buď odpověď „ANO“ nebo odpověď „NE“.

1.6.2 Příklad. *SAT – splňování Booleovských formulí*: Je dána výroková formule φ v CNF. Rozhodněte, zda je φ splnitelná.

Na danou formuli φ je tedy odpověď (tj. řešení) buď „ANO“ nebo „NE“. Všimněte si, že v tomto případě se neptáme po ohodnocení, ve kterém je formule pravdivá – zajímá nás pouze fakt, zda je splnitelná.

1.6.3 Řada praktických úloh není podobného druhu jako uvedený příklad. Často se jedná o tzv. optimalizační úlohy, tj. úlohy, kde mezi přípustnými řešeními hledáme přípustné řešení v jistém smyslu optimální. Obvykle to bývá tak, že je dána účelová funkce, která každému přípustnému řešení přiřadí číselnou hodnotu, a úkolem je najít přípustné řešení, pro které je hodnota účelové funkce optimální, tj. buď největší nebo naopak nejmenší. V dalším textu se s řadou těchto úloh setkáme. Nyní uvedeme jeden příklad.

1.6.4 Problém obchodního cestujícího – TSP. Jsou dána města 1,2, $\ldots, n$. Pro každou dvojici měst i, j je navíc dáno kladné číslo $d(i,j)$ (tak zvaná vzdálenost měst i, j). *Trasa* je dána permutací π množiny $\{1, 2, \ldots, n\}$ do sebe. Cena trasy odpovídající permutaci π je

$$\sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)) + d(\pi(n), \pi(1)).$$

Neformálně, trasa je pořadí měst, ve kterém má obchodní cestující města projít, a to tak, aby každé město navštívil přesně jednou a vrátil se do toho města, ze kterého vyšel. Cena trasy je pak součtem všech vzdáleností, které při své cestě urazil.

1.6.5 TSP – rozhodovací verze. Kromě čísel $d(i,j)$ z 1.6.4 je dáno číslo C. Existuje trasa π ceny nejvýše C?

1.6.6 TSP – vyhodnocovací verze. Jsou dána čísla $d(i,j)$ a 1.6.4. Najděte cenu optimální trasy, tj. trasy s nejmenší možnou cenou.

1.6.7 TSP – optimalizační verze. Jsou dána čísla $d(i,j)$ a 1.6.4. Najděte optimální trasu, tj. trasu s nejmenší možnou cenou.

1.7 Třídy $\mathcal{P}$ a $\mathcal{NP}$

1.7.1 Instance úlohy jako slovo nad vhodnou abecedou. Instance libovolné rozhodovací úlohy můžeme zakódovat jako slova nad vhodnou abecedou. Ukažme si to na příkladě problému SAT a úlohy nalezení nejkratší cesty v daném orinetovaném ohodnoceném grafu.

- Pro problém SAT (splňování booleovských formulí) je instancí libovolná formule φ v konjunktivním normálním tvaru (CNF). Označme jednotlivé logické proměnné formule φ jako x_1, x_2, $\ldots, x_n$. Pak φ můžeme zakódovat jako slovo nad abecedou $\{x, 0, 1, (,), \vee, \wedge, \neg\}$ takto: proměnná x_i se zakóduje slovem xw, kde w je binární zápis čísla i, ostatní symboly jsou zachovány.

  Na příklad formuli $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_4)$ odpovídá slovo
  $$(x1\ \vee\ \neg\, x10 \vee x11)\ \wedge\ (\neg\, x1\ \vee\ x100).$$

- U úlohy nalezení nejkratší cesty z vrcholu r do vrcholu c můžeme postupovat takto: Instanci tvoří matice délek daného orientovaného ohodnoceného grafu, dvojice vrcholů r a c a číslo k. Matici není těžké zakódovat jako slovo, za ní pak následuje pořadové číslo vrcholu r, pořadové číslo vrcholu c a číslo k, vše oddělené např. symbolem $\#$.

1.7.2 Úloha jako jazyk nad abecedou. Protože řešením rozhodovací úlohy je buď „ANO“ nebo „NE“, rozdělíme instance úlohy na tzv. „ANO-instance“ a „NE-instance“. *Jazyk* úlohy $\mathcal{U}$, značíme jej $L_{\mathcal{U}}$, se skládá ze všech slov odpovídajících ANO-instancím úlohy $\mathcal{U}$.

Uvědomte si, že některá slova nad abecedou Σ nemusí odpovídat žádně instanci dané úlohy. Tato slova chápeme jako „ NE-instance“. Můžeme proto říci, že množina všech NE instancí tvoří doplněk jazyka $L_{\mathcal{U}}$, tj. je to $\Sigma^\star \setminus L_{\mathcal{U}}$.

1.7.3 Třída $\mathcal{P}$. Řekneme, že rozhodovací úloha $\mathcal{U}$ leží ve třídě $\mathcal{P}$, jestliže existuje deterministický Turingův stroj, který rozhodne jazyk $L_{\mathcal{U}}$ a pracuje v polynomiálním čase; tj. funkce $T(n)$ je $\mathcal{O}(p(n))$ pro nějaký polynom $p(n)$.

1.7.4 Příklady.

- Minimální kostra v grafu. Je dán neorinetovaný graf G s ohodnocením hran c. Je dáno číslo k. Existuje kostra grafu ceny menší nebo rovno k?
- Nejkratší cesty v acyklickém grafu. Je dán acyklický graf s ohodnocením hran a. Jsou dány vrcholy r a c. Je dáno číslo k. Existuje orientovaná cesta z vrcholu r do vrcholu c délky menší nebo rovno k?
- Toky v sítích. Je dána síť s horním omezením c, dolním omezením l, se zdrojem z a spotřebičem s. Dále je dáno číslo k. Existuje přípustný tok od z do s velikosti alespoň k?
- Minimální řez. Je dána síť s horním omezením c, dolním omezením l. Dále je dáno číslo k. Existuje řez, který má kapacitu menší nebo rovnu k?

Uvedli jsme všechny úlohy v rozhodovací verzi. Velmi často se mluví i o jejich optimalizačních verzích jako o polynomiálně řešitelných úlohách.