Hrátky s Archimédovým zákonem

odborná práce z fyziky

Dominika Gajdová

3. listopadu 2018

Obsah

1 Úloha batohu

Problém batohu *(0-1 knapsack problem)* je úloha kombinatorické optimalizace, jejímž cílem je umístění podmnožiny předmětů (předmět má specifikovánu svoji cenu a váhu) do přepravky omezené kapacity (nosnosti) tak, aby cena nákladu byla maximální. Předměty nelze dělit – buď jsou v přepravce umístěny celé, nebo zde nejsou vůbec. Rozhodovací NP-úplná varianta problému řeší otázku, zda-li lze do batohu dané kapacity umístit podmnožinu daných předmětů tak, aby byl součet cen těchto předmětů alespoň k.

1.1 Formulace pomocí lineárního programování

Pomocí celočíselného lineárního programování lze problém batohu formulovat jako:

$$\max \sum_{n=1 \in I} x_i \cdot c_i$$

Za podmínek:

$$\sum_{n=1 \in I} x_i \cdot w_i \leq W, x_i \in 0, 1$$

1.2 Pseudopolynomiální algoritmus

Problém batohu lze řešit pomocí dynamického programování v pseudopolynomiálním čase. Algoritmus funguje na bázi vyplňování tabulky. Její sloupce značí cenu doposud naloženého nákladu, řádky značí stav po naložení n-té položky, buňky označují váhu nákladu.

Algoritmus je v prvním kroku v buňce $[0, 0]$, což znamená, že byla zpracována nultá položka (tj. nebyla ještě zpracována žádná položka) a cena nákladu je proto 0 (obsah buňky). Algoritmus nyní zpracuje první položku (posun na další řádek tabulky).

Mohou nastat dvě situace – dojde proto k rozdělení řešení a algoritmus bude postupovat po obou větvích. První možností je, že algoritmus do batohu přidá danou položku a přejde na souřadnici $[y + 1, x + cost(y)]$, kde x je dosavadní součet cen položek obsažených v batohu a nastaví hodnotu pole na z + váha předmětu (z je dosavadní součet vah všech předmětů obsažených v batohu). Druhou možností je, že předmět do batohu nebude přidán – algoritmus přejde na souřadnici $[y + 1, x]$, kde hodnotu pole stanoví opět jako

součet vah všech obsažených předmětů (protože nebyl přidán žádný předmět, tak hodnotu pouze opíše ze zdrojového pole). Stejným způsobem algoritmus postupuje pro všechny dosažené buňky v následujícím řádku, dokud nejsou zpracovány všechny předměty (vyplněny všechny řádky).

Pokud dojde ke kolizi – algoritmus se dostane do situace, kdy se do dané buňky může dostat více cestami, tak dostane přednost to řešení, jehož váha nákladu je nižší. Pokud váha nákladu překročí kapacitu batohu, tak v dané větvi nemá smysl pokračovat, protože řešení je nepřípustné.

1.3 Složitost algoritmu

Algoritmus je pseudopolynomiální, přesněji má složitost $O(C \cdot n)$, což znamená, že lze jeho složitost omezit polynomem vzhledem k délce vstupu (n je počet předmětů), ale nikoliv vzhledem k velikosti vstupu (C může být až exponenciálně velké).

1.4 Příklad

Mějme batoh o kapacitě 8 a předměty, jejichž váhy jsou $[3, 6, 4, 1]$, tyto předměty mají hodnotu $[3, 5, 3, 1]$. Umístěte předměty do batohu tak, aby součet jejich cen byl maximální. Jako horní mez ceny použijte součet ceny všech předmětů.

$x_i / \sum$	0	1	2	3	4	5	6	7	8
$0(v/w)$	0								
$1(3/3)$	0			3					
$2(5/6)$	0			3		6			~~9~~
$3(3/4)$	0			3		6	7		~~10~~
$4(1/1)$	0	1		3	4	6	7	8	