Domácí úloha č. 1 – Knapsack problém

Specifikace úlohy

Cílem domácí úlohy byla implementace řešení problému batohu a to hrubou silou a dále pak jednoduchou heuristikou. K dispozici byla vstupní data pro 4 až 40 položek batohu a rovněž soubory s výsledky.

Nástroje k řešení

K implementaci jsem využil programovací jazyk **Java** pod prostředím **NetBeans**. Všechny výpočty běželi na procesoru Intel Core 2 Duo 3.00 GHz a pod operačním systémem Microsoft Windows 7. Výsledky byly zpracovány tabulkovým procesorem Microsoft Excel.

Výsledný zdrojový kód je spouštěn ze souboru **Main.java**, zbytek kódu je přehledně rozdělen do tříd.

K měření času jsem použil funkci **System.currentTimeMillis().**

Rozbor variant řešení

Úkolem bylo:

- Naprogramovat řešení problému **pomocí hrubé síly** a sledovat závislost výpočetního času na N.
- Naprogramovat řešení problému **pomocí jednoduché heuristiky** podle poměru cena/váha. Sledovat závislost výpočetního času na N a dále relativní chybu oproti metodě hrubou silou.

Popis postupu řešení

Algoritmus hrubou silou

Cílem tohoto algoritmu je projít celý stavový prostor. Je tedy úkolem projít úplně všechny možné konfigurace a pro každou konfiguraci vyzkoušet, jestli jí lze naplnit batoh a jakou bude mít cenu. Každá konfigurace určuje, jaké položky v batohu jsou a jaké ne, proto jsem pro reprezentaci každé konfigurace použil pole bitů, kde každý bit určuje přítomnost položky. Pole bitů je dlouhé podle toho, s kolika věcmi pracujeme. Procházení dalších stavů se děje tak, že akorát přičítám jedničku k poli bitů, takže procházím celý prostor iterativně, ne rekurzivně přes zásobník (což není nejspíš ideální metoda, protože se špatně dělá back-track, když už vím, že takové řešení nevede k lepšímu řešení).

U každé konfigurace se tedy sleduje cena řešení a pokud je cena lepší než aktuálně nejlepší cena, tak se tato cena uloží a považuje se za aktuálně nejlepší. Počet stavů je 2^N, (protože každá věc v batohu buď je, nebo není) a udává tak složitost algoritmu.

Algoritmus s heuristikou dle poměru cena/váha

Princip tohoto algoritmu je ten, že si spočítám u každé položky poměr cena/váha a iterativně se pokouším přidávat do batohu položky, dokud není batoh plný. Pokud nelze nějaká položka přidat, protože se nevejde, tak pokračuji v přidávání, protože dále může být položka, co se již vejde.

Tímto algoritmem se neprochází celý stavový prostor a je tak časově a paměťově méně náročná, což je zřejmé i z následujících grafů. Složitost algoritmu je n • log n, protože se v každém kroku zabýváme všemi předměty které lze vložit do batohu, ale v každém kroku se tento počet sníží o jedničku.

Naměřené výsledky

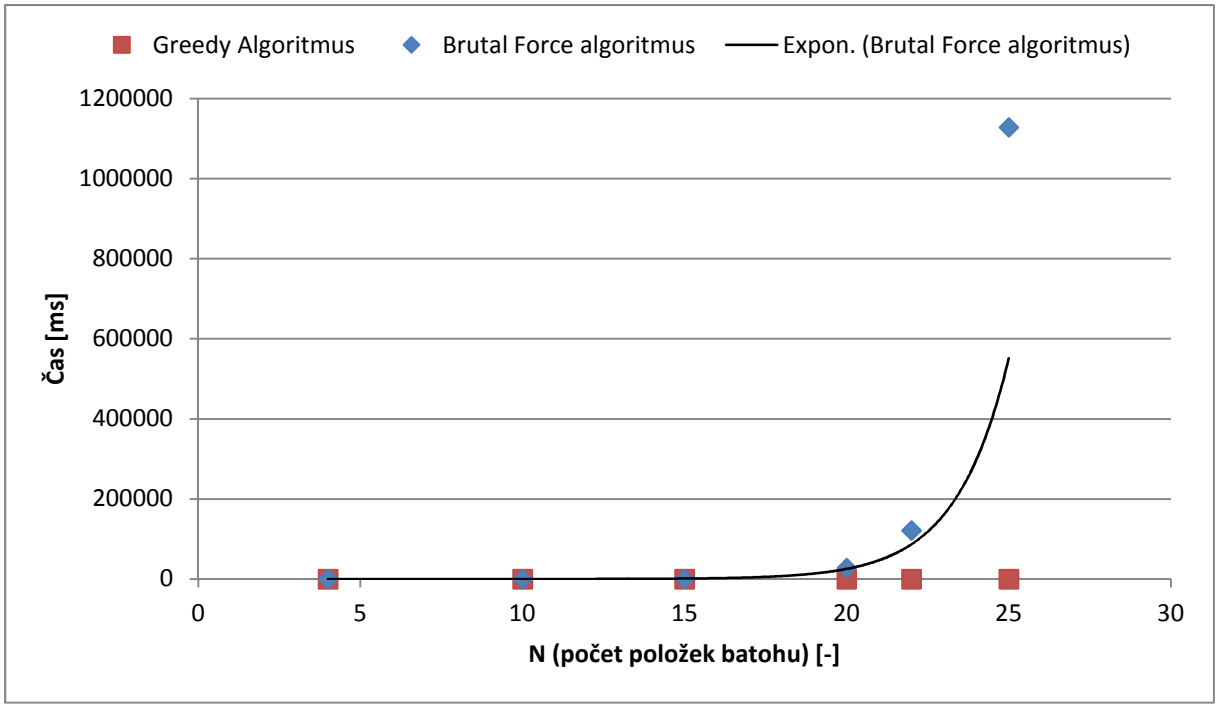
Cílem tedy bylo sledovat závislost výpočetního času na N pro oba zmíněné algoritmy a dále relativní odchylku. Naměřená data zobrazíme v přehledné tabulce následující grafem. Výpočty pro nižší N jsem opakoval vícekrát, abych získal přesnější čas na jeden výpočet.

Závislost výpočetního času na N

Následující tabulka zachycuje naměřené časy v ms. V pravé části jsou časy přepočteny na jeden průběh algoritmu. Hodnoty jsou naměřené pouze do N=27, více již výpočet trval neúnosně dlouho.

N	# opakování	BruteForce [ms]	GreedyAlg [ms]	BruteForceOne [ms]	GreedyAlgOne [ms]
4	1000	5501	5016	5,501	5,016
10	1000	17049	6267	17,049	6,267
15	500	137524	3267	275,048	6,534
20	20	559602	152	27980,1	7,6
22	5	606503	54	121300,6	10,8
25	1	1128557	11	1128557	11
27	1		24		24
30	1		11		11
32	1		13		13
35	1		12		12
37	1		13		13
40	1		14		14

Následující graf zobrazuje závislost výpočetního času na N. Je zde vidět, že algoritmus BrutalForce má exponenciální závislost, což jsem i naznačil spojnicí trendu (Expon.).



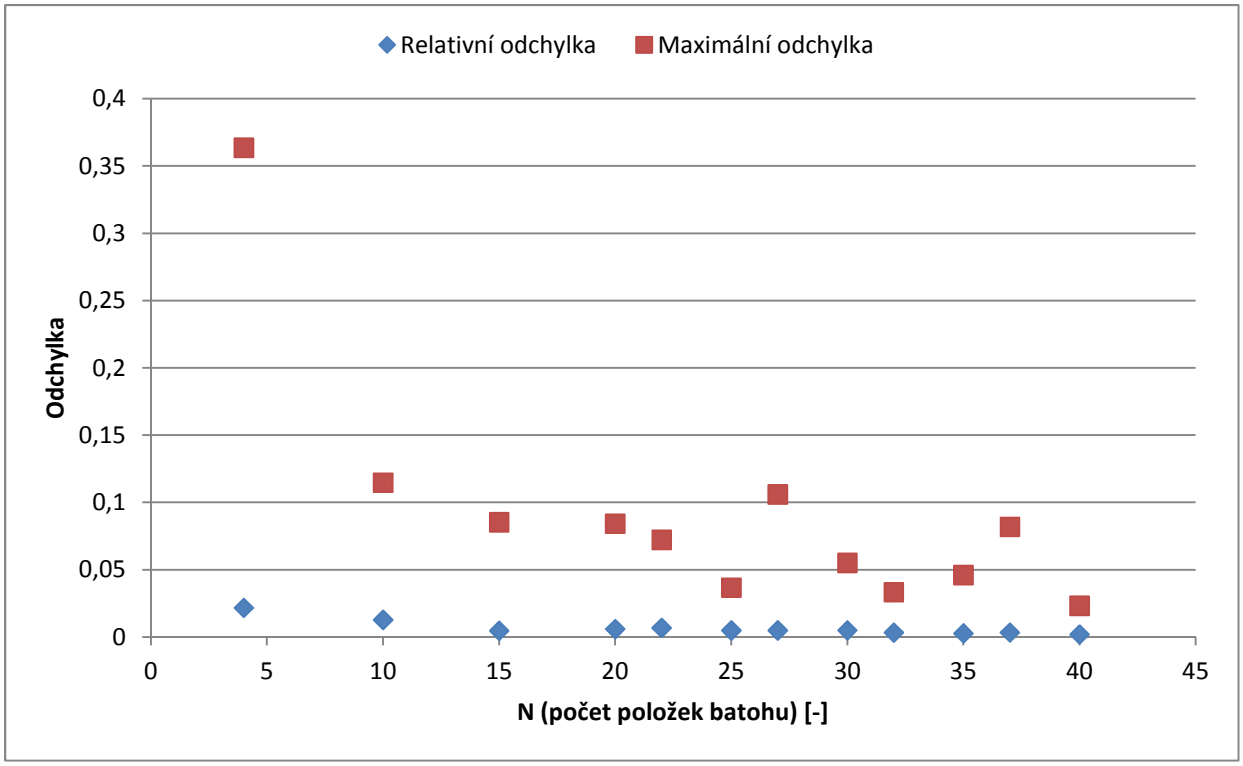
Relativní chyba použitých algoritmů

Dalším úkolem bylo zjistit relativní chybu použité heuristiky. Relativní odchylka byla spočítána jako průměrná hodnota všech naměřených odchylek pro každou instanci daného N.

Opět výpis naměřených dat tabulkou následovanou grafickým znázorněním.

N	Relativní odchylka	Maximální odchylka
4	0,021745081	0,363636364
10	0,012861986	0,114800759
15	0,004758918	0,085427136
20	0,006000856	0,084337349
22	0,006866936	0,072289157
25	0,004983556	0,036789298
27	0,0050165	0,106017192
30	0,005074467	0,055137845
32	0,003411919	0,033407572
35	0,002801854	0,046092184
37	0,003436086	0,081967213
40	0,001994868	0,023372287

Grafické znázornění relativní a maximální odchylky:



Závěr

Cílem bylo vyzkoušet si vyřešit kombinatorický problém pomocí metody hrubou silou a pomocí jednoduché heuristiky.

Na této úloze se krásně ukázalo, že pokud nepotřebujeme nejlepší možný výsledek, heuristiky nám dokážou poskytnout relativně dobrý výsledek (nebo i optimální) ve velice krátkém čase ve srovnání s algoritmem hrubé síly. V praxi lze narazit na velice mnoho aplikací, kde se upřednostňuje rychlost dodání informací před jejich správností, např. real-time přenosy, logistické operace, různé odhady potřebné ke strategickému rozhodování, předpovědi počasí a podobné.