1 MI-PAA, Úloha 1: eení problému batohu metodou hrubé síly a jednoduchou heuristikou

Marián Hlavá, 18 Oct 2017 (hlavam30)
marian.hlavac@fit.cvut.cz
https://github.com/mmajko/knapsack-problem

2 Zadání úlohy

- Naprogramujte eení problému batohu hrubou silou (tj. exaktn). Na zkuebních datech pozorujte závislost výpoetního asu na n.
- Naprogramujte eení problému batohu heuristikou podle pomru cena/váha. Pozorujte závislost výpoetního asu na n, prmrnou a maximální relativní chybu (tj. zhorení proti exaktní metod) v závislosti na n.

2.1 Moné varianty eení

Problém batohu je moné eit hrubou silou, heuristicky, dynamickým programováním, algoritmem "meet-in-the-middle" a dalími zpsoby. Heuristická metoda eení se dále dlí i podle zvolené heuristiky, nap. jednoduchá greedy heuristika upednostuje nejdraí pedmty. Takových heuristik existuje více, lií se rychlostí i komplexitou.

Zvolenou variantou pro první úlohu jsou eení hrubou silou a jednoduchou heuristikou vyuívající pomr cena/váha u jednotlivých pedmt.

2.2 Popis postupu eení

Algoritmus a celý program poskytující výsledky je napsán v jazyce *Rust*. Tento program nate instance z pedpipravených datových soubor urených pro tuto úlohu a vypote eení hrubou silou a eení za pomoci heuristiky. Zapíe délku provádní výpotu a vechna data poskytne v CSV formátu.

Druhým nástrojem je pak *Jupyter Notebook*, ve kterém se poskytnutá data zpracují a vizualizují, zapíou se výsledky tchto mení výpot a sepíe se zpráva.

2.2.1 Kostra algoritmu

Kompletn celý algoritmus je k nahlédnutí ve zdrojových souborech programu. Pro rychlou pedstavu je níe uveden krátký náhled na algoritmus výpotu za pomoci heuristiky v jazyce Rust, který je aktuáln pouit pro výsledky uvedené níe.

```
...
fn solve_heuristic(knap: &Knapsack) -> (u16, u16, u32) {
    let mut items: Vec<(usize, &KnapItem)> = knap.items.iter().enumerate().collect();
    items.sort_unstable_by(|a, b| (a.1.price / a.1.weight).cmp(&(b.1.price / b.1.weight)));

    let mut result_items: Vec<&KnapItem> = vec![];
    let mut total_weight = 0;
    for item in items {
        if item.1.weight + total_weight <= knap.capacity {
            result_items.push(item.1);
```

```
            total_weight += item.1.weight;
        } else {
            break;
        }
    }
...
```

Pi výpotu hrubou silou jsou pro kadou jednotlivou instanci vyzkoueny vechny kombinace umístní pedmt do batohu a následn je vybrána ta nejlepí vhodná (optimální). U této metody si meme být jisti, za pedpokladu, e je výpoet kompletní, e jsme nalezli optimální ení.

Implementaní detaily ení lze nalézt ve zdrojových kódech. Byla pouita bitová maska pítomnosti pedmtu v batohu.

Výpoet heuristikou pak spoívá v seazení pole pedmt podle kritéria heuristiky. Z tohoto pole jsou pak vybírány pedmty do vyerpání jeho kapacity.

2.3 Surová namená (raw) data

Níe uvedená tabulka je náhled na kompletní surová výstupní data z programu. Data mete sami (nap. pro kontrolu) získat jednoduchým zpsobem - sputním skriptu `generate.sh`, který vytvoí soubor `results.csv` obsahující tato data.

2.3.1 Sloupce

Názvy sloupc se vyskytují i dále v textu, zde je jejich struný popis:

- **knap_id** - identifikátor instance
- **item_count** - poet pedmt (konfigurace instance)
- **capacity** - kapacita batohu
- **method** - metoda výpotu
- *Bruteforce* je výpoet hrubou silou, *Heuristic* je heuristický výpoet (heuristika pomru váha/cena)
- **price** - vypotená celková cena batohu
- **weight** - vypotená celková váha batohu
- **bitmask** - bitmaska (jednoznaný identifikátor, maska pítomnosti pedmtu) ení
- **elapsed_ms** - doba výpotu v milisekundách
- **optimal_price** - optimální cena batohu

```
Out[1]:      knap_id  item_count  capacity      method  price  weight     bitmask  \
        0       9000           4       100  Bruteforce    473      63          11
        1       9000           4       100   Heuristic    415      91          12
        2       9001           4       100  Bruteforce    326      66          12
        3       9001           4       100   Heuristic    326      66          12
        4       9002           4       100  Bruteforce    196      89           1
        ..       ...         ...       ...         ...    ...     ...         ...
        715     9357          30       400   Heuristic   1923     400   751821317
        716     9358          30       400  Bruteforce   3773     400   960495591
        717     9358          30       400   Heuristic   1978     400   802359034
```

```
718     9359          30       400  Bruteforce   3786     396  1055772601
719     9359          30       400   Heuristic   1944     399   292762847

        elapsed_ms  optimal_price
0         0.010436            473
1         0.000834            473
2         0.001787            326
3         0.000440            326
4         0.001451            196
..             ...            ...
715       0.006197           3659
716  316909.000000           3773
717       0.020630           3773
718  317232.000000           3786
719       0.005381           3786

[720 rows x 9 columns]
```

2.4 Výsledky mení

Níe jsou uvedeny výsledky derivované z dat. Rychlosti eení jsou seskupeny podle potu pedmt a zprmrovány. Výsledné prmrné asy jsou ke kadé metod eení uvedeny jak tabulkou, tak grafem.

Mítko graf je lineární na vertikální ose.

2.4.1 Rychlost eení hrubou silou

Mení hrubou silou je výpoetn náron metoda. Sloitost je `O(2^n)`, take lze oekávat rapidn vzrstající trend doby nutné pro dokonení výpotu touto metodou.

Out[2]:
```
                elapsed_ms
item_count
4                 0.001835
10                0.215534
15                9.060000
20              273.200000
22             1075.540000
25             8828.420000
27            35933.460000
30           304364.000000
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x108cc68d0>
```

Na výsledných datech lze pozorovat velmi rychle vzrstající asovou závislost na potu pedmt.

Díky sloitosti `O(2^n)` obecn platí, e pidání jednoho dalího pedmt zdvojnásobí celý výpoetní as. Na výsledných datech lze tuto vlastnost snadno pozorovat.

Kupíkladu pro 20 pedmt byla prmrná doba výpotu `266 ms`. Pro 22 pedmt `1100 ms`. To je 4.13 násobek pvodního ísla, lo o rozdíl dvou pedmt, tudí dvojnásobek asu za kadý pidaný pedmt skuten odpovídá.

2.4.2 Rychlost eení jednoduchou heuristikou

Rychlost eení heuristikou by mla být podstatn mén asov nároná, ne výpoet hrubou silou.

Jeliko je v prbhu eení azeno pole podle heuristiky, lze oekávat, e se vzrstajícím potem pedmt v poli se prodlouí i doba azení tohoto pole.

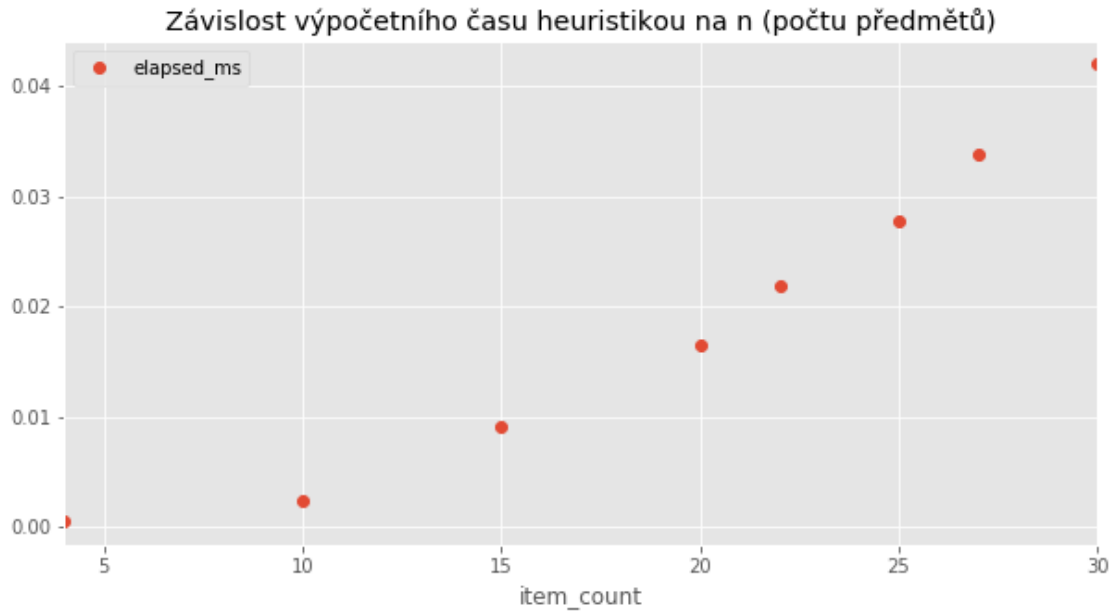
azení pole má na starosti funkce jazyka *Rust* `Vec::sort_unstable_by()`, která slibuje sloitost O(n log n) a je zaloena na pattern-defeating quicksortu.

Zbytek algoritmu po seazení je lineární (sloitost v této ásti nezávisí na **n**, tedy potu pedmt, ale na kapacit batohu a velikosti pedmt - irelevantní pro nae pozorování).

Out[4]:

item_count	elapsed_ms
4	0.000428
10	0.001849
15	0.006769
20	0.007424
22	0.005415
25	0.005903
27	0.006018
30	0.008205

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x108cac5c0>



Na grafu lze pozorovat lineární vzrst výpoetního asu pi vzrstu potu pedmt. Rozdíl je vak v relativn nepatrný. Na grafu je dleité si vimnout rozsahu vertikální osy, která se se pohybuje v desetinách milisekund. Píina pozorovaného vzrstu je pak dána nutností adit delí pole pedmt, co výpoetní as ovlivuje relativn minimáln.

Pokud bychom následovali lineární trend, citelnou asovou prodlevu bychom mohli pozorovat u pi nap. 100 000 pedmtech, které by zpsobily piblin `133 ms` dlouhou prodlevu, za pedpokladu, e takovou prodlevu pedpokládáme za citelnou (tvrzení je spíe subjektivní záleitostí, nelze jednoznan íct, co je dlouhá prodleva).

2.4.3 Relativní chyby pi výpotu heuristikou

Jednoduchá heuristika, jako ta, která byla pouita v této úloze, s nejvtí pravdpodobností nebude schopná urit optimální eení v kadé instanci.

Relativní chybou lze urit úspnost heuristického výpotu vi exaktního výpotu hrubou silou. Relativní chyby jsou uvedeny v procentech.

Out[6]:

	knap_id	item_count	optimal_price	price	error	relative_error_%
1	9000	4	473	415	58	12.262156
3	9001	4	326	326	0	0.000000
5	9002	4	196	196	0	0.000000
7	9003	4	545	437	108	19.816514
9	9004	4	243	243	0	0.000000
..	...	...	...	...	...	...

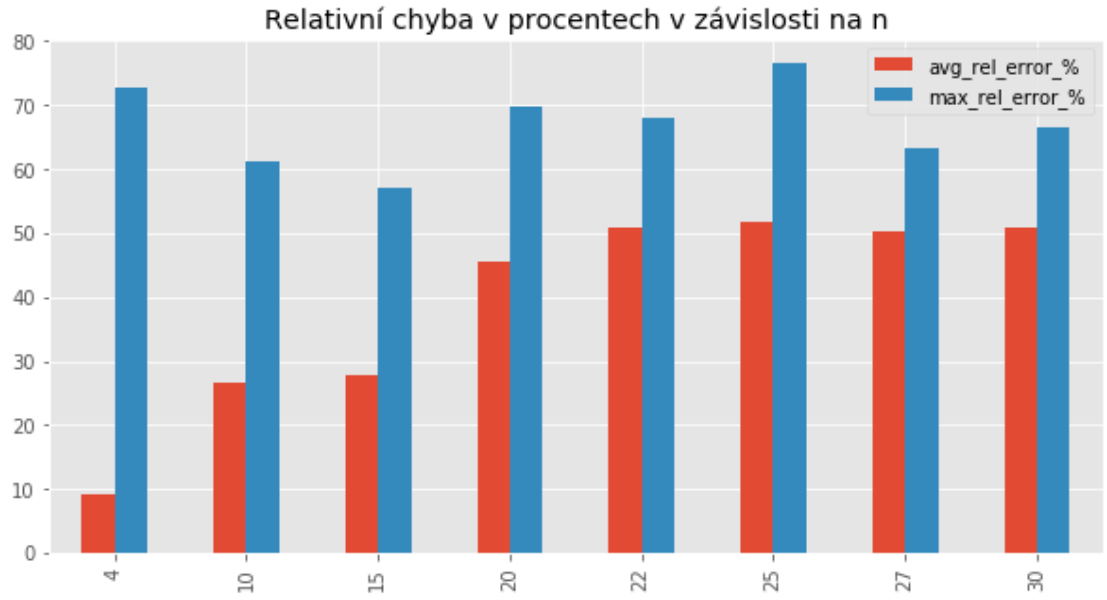
```
711     9355          30          3968    2201    1767          44.531250
713     9356          30          3757    2000    1757          46.766037
715     9357          30          3659    1923    1736          47.444657
717     9358          30          3773    1978    1795          47.574874
719     9359          30          3786    1944    1842          48.652932

[360 rows x 6 columns]
```

Out[7]:

	avg_rel_error_%	max_rel_error_%
4	9.125447	72.727273
10	26.621721	61.094819
15	27.724194	56.969697
20	45.451606	69.780220
22	50.792898	67.945310
25	51.872149	76.540688
27	50.407790	63.391390
30	50.760173	66.676292

Out[8]: `<matplotlib.axes._subplots.AxesSubplot at 0x10bb52390>`



Out[9]:

	avg_rel_error_%
mean	39.094497
std	15.980391

```
min          9.125447
25%         27.448576
50%         47.929698
75%         50.768354
max         51.872149
```

Na grafu lze pozorovat mírn vzrstající trend prmrné relativní chyby pi vzrstajícím potu pedmt. O lineární vzrst vak s nejvtí pravdpodobností nepjde a dalo by se spíe pedpokládat, e hodnota prmrné chyby se limitn blíí k 50%. Lepí odhad by poskytl vtí vzorek dat, pro vtí poty pedmt.

Grafická reprezentace maximální relativní chyby nenese ádnou podstatnou informaci (z grafu nelze vyíst nic pouitelného pro závr).

2.5 Závr

Prvotní pedpoklad, e výpoet problému batohu pomocí jednoduché heuristiky bude ádov rychlejí, ne výpoet hrubou silou, se potvrdil. Na datech lze vidt dsledky sloitosti algoritmu `O(2^n)`.

Prmrná relativní chyba pi eení pomocí heuristiky se ukázala, e je spíe vyí a tak se podstatn lií i kvalita eení obou metod. Metoda eení pomocí heuristiky toti vrací spíe mén kvalitní eení (asto `<50%`).

Lze tvrdit, e výpoet heuristikou se vyplácí a v bod, kdy relativní chyba eení nevzrstá, protoe víme, e asová náronost eení hrubou silou bude vzrstat zaruen vdy. K potvrzení tohoto tvrzení by byly vhodné dalí vzorky dat pro vyí poty pedmt, aby mohl být trend relativní chyby jednoznan uritelný. Výpoet dalích vzork dat je vak asov nároný a pesahuje hranice této úlohy.

Zdrojové soubory úlohy lze najít na GitHubu. Link je uveden v hlavice zprávy.