Report: Problém batohu

Specifikace úlohy

Úkoly:

1. Vytvoření programu, který řeší konstruktivní *0/1* problém batohu (dále *problém*).
   a. Umí využít tyto algoritmy:
      - Dynamické programování
      - Jednoduchá greedy heuristika
      - Vylepšená greedy heuristika
      - Algoritmus FPTAS
2. Experimentální vyhodnocení výpočetní složitosti na velikosti instance
3. U FPTAS experimentální vyhodnocení závislosti opravdové chyby a výpočetního času na zvolené přesnosti algoritmu.

Výpočetní složitost je měřena počtem navštívených konfigurací, tedy vyhodnocených sestav věcí v batohu.

Pro testování byly zadány dvě sady náhodně vygenerovaných instancí řešeného *problému* pro *N=4 10 15 20 22 25 27 30 32 35 37 40*, kde hodnota *N* značí počet věcí, ze kterých lze v jednotlivých instancích vybírat. Každá sada také obsahuje všechna řešení konstruktivní verze jednotlivých instancí.

Plné zadání úlohy zde.

Program

Program byl napsán v jazyce Python a je rozdělen do několika souborů:

Soubor	Popis
knapsackSolver.py	Spuštění programu pro vyhodnocení jednoho souboru s instancemi *problému*.
solverStrategy.py	Obsahuje všechny využívané algoritmy pro vyřešení instancí *problému*.
output_generator.py	Slouží pro automatizované vyřešení všech problémů v sadě pomocí opakovaného volání vytvořeného programu.

Pro vyřešení *problému* lze využít tyto **algoritmy**:

- DP
- DPWeight
- Greedy
- GreedyOne
- FPTAS

Popis algoritmů

Algoritmus **DP** používá dekompozici problému podle ceny jednotlivých věcí. Vytváří tabulku velikosti S_MxN, kde N značí počet věcí a S_M značí součet cen všech věcí. Tento algoritmus je iterativní, tabulka je plněna odzdola nahoru.

Algoritmus **DPWeight** používá dekompozici problému podle hmotnosti jednotlivých věcí. Vytváří tabulku CxN, kde N značí počet věcí a C značí maximální kapacitu batohu. Tento algoritmus je rekurzivní, tabulka je plněna odshora dolů.

Algoritmus **Greedy** je heuristika, která plní batoh, dokud dostačuje kapacita. Nejdříve jsou v jednotlivých instancích všechny zadané věci seřazeny sestupně podle vyhodnocení $\frac{cena}{hmotnost}$. Maximální chyba výsledku nelze omezit.

Algoritmus **GreedyOne** je heuristika, která vloží do batohu právě jednu nejcennější věc, pro kterou dostačuje kapacita. Nejdříve jsou v jednotlivých instancích všechny zadané věci seřazeny sestupně podle ceny. Maximální chyba výsledku nelze omezit.

Algoritmus **FPTAS** je heuristika, která umožňuje určit přesnost, se kterou výsledek získáme. Zadanou maximální chybu použijeme k zanedbání nejméně významných bitů cen jednotlivých věcí. Takto zjednodušené zadání předáme algoritmu DP.

Předpokládané výsledky

Algoritmy Greedy a GreedyOne mají lineární složitost $O(n)$, proto očekávám velmi pomalý růst oproti ostatním algoritmům.

Algoritmy DP a FPTAS mají složitost $O(n^2 S_m)$. Protože FPTAS používá algoritmus DP a výrazně snižuje ceny všech věcí v jednotlivých instancích, očekávám jeho mnohem pomalejší růst oproti algoritmu DP.

Algoritmus DPWeight má složitost $O(nC)$, proto očekávám pomalejší růst oproti algoritmu DP a FPTAS.

Hardwarová konfigurace

Výsledná data byla vygenerována na operačním systému Linux, procesor 2,3 GHz Intel Core i5, RAM paměť 8 GB 2133 MHz LPDDR3.

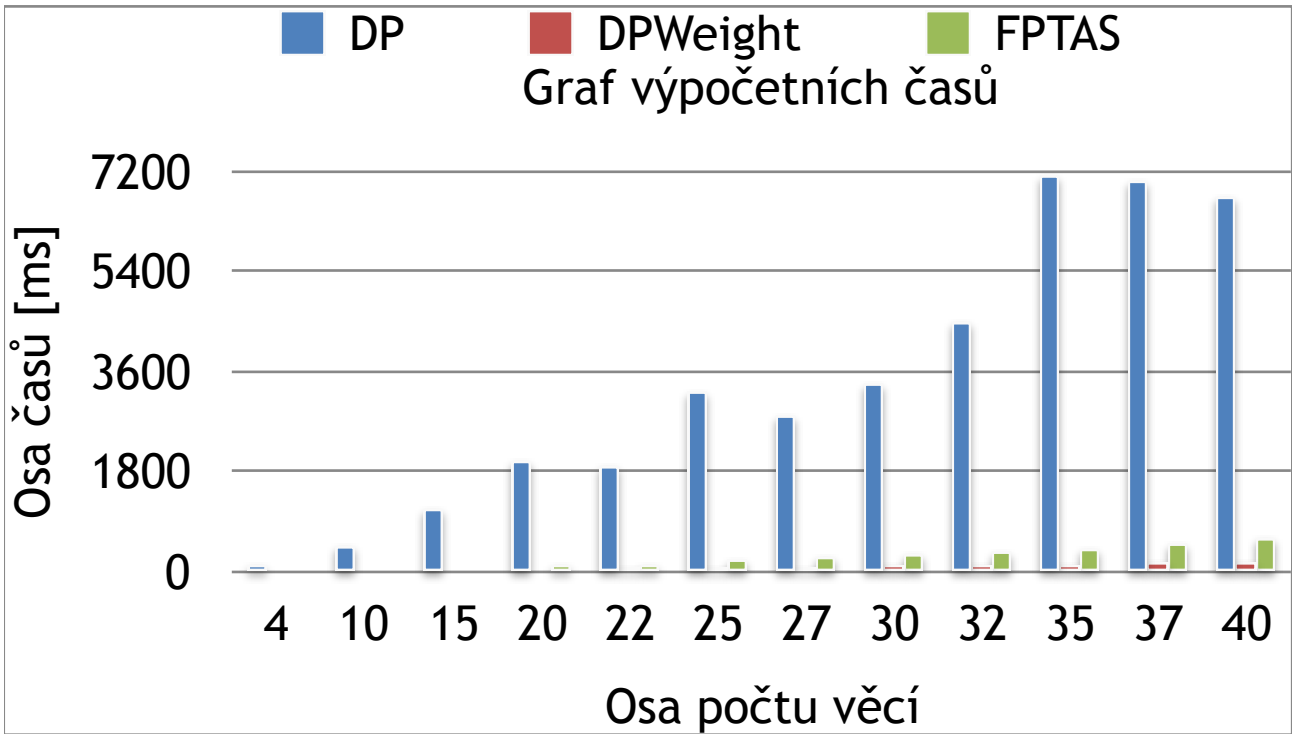
Naměřené výsledky

Všechny výsledky byli naměřeny na všech zadaných datasetech NK, ZKC a ZKW.

Srovnání výpočetních časů

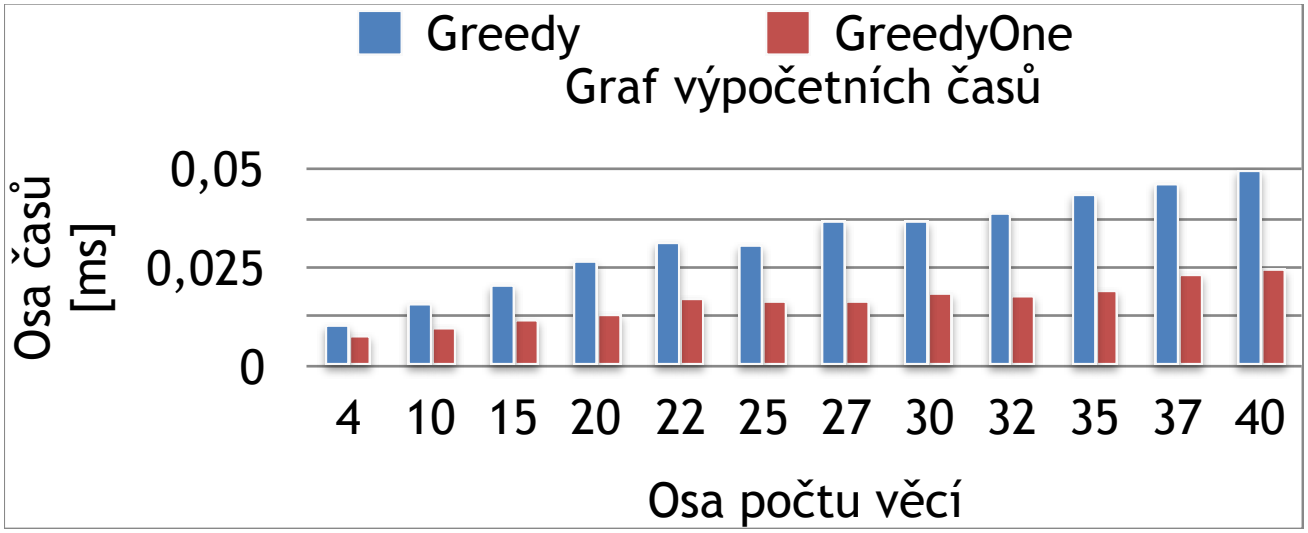
Průměrný čas [ms] podle počtu věcí

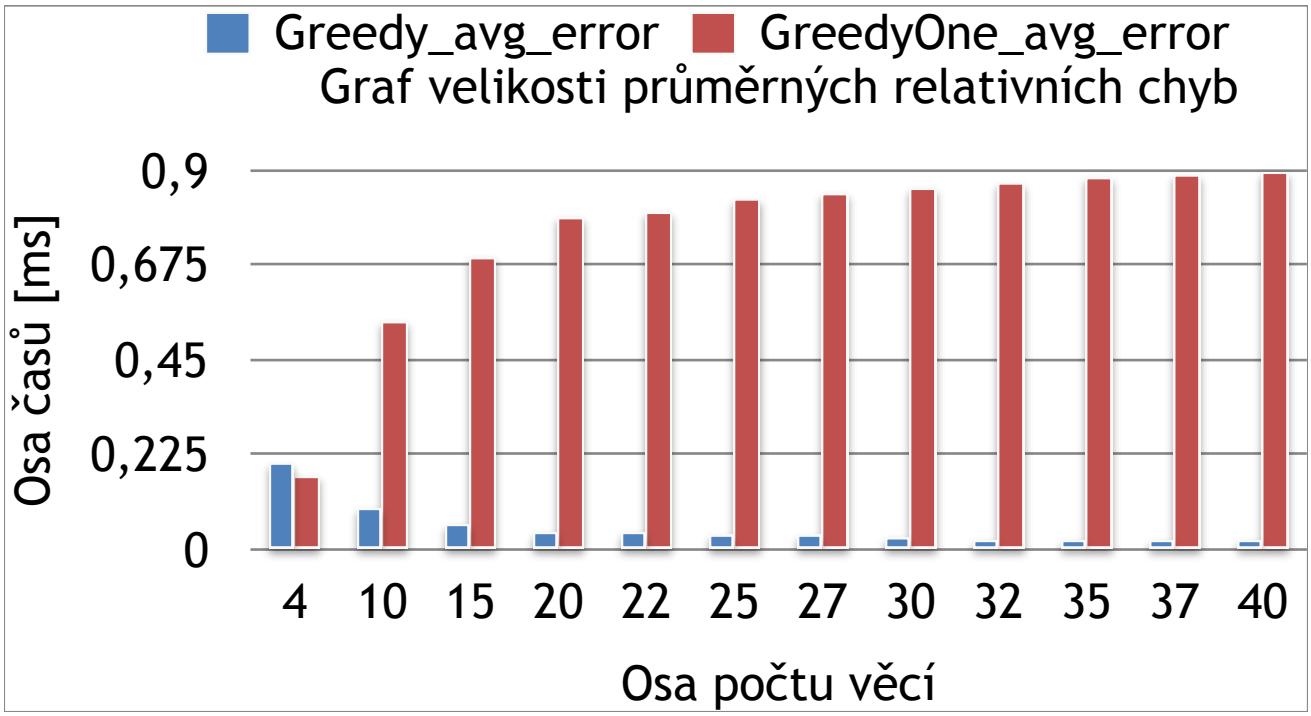
item_count	DP	DPWeight	FPTAS	Greedy	GreedyOne
4	93,6982	1,4304	0,8519	0,0105	0,0079
10	456,8244	7,7131	10,227	0,016	0,0099
15	1106,8275	21,3662	32,6015	0,0204	0,0119
20	1950,3325	37,899	83,623	0,0269	0,0133
22	1876,8124	51,397	116,6782	0,0317	0,0171
25	3221,8843	61,1677	175,6165	0,0305	0,0163
27	2785,1095	72,0347	222,2353	0,0372	0,0168
30	3349,9526	85,181	292,5677	0,037	0,0186
32	4461,4599	103,9543	331,5438	0,0387	0,0179
35	7109,8726	119,1089	400,9741	0,044	0,0194
37	7009,423	136,0131	460,7465	0,0463	0,0236
40	6734,9957	147,9735	586,6446	0,0497	0,0247



Analýza algoritmů Greedy a GreedyOne

item_count	Greedy_max_error	GreedyOne_max_error	Greedy_avg_error	GreedyOne_avg_error
4	0,99422	0,663519	0,201841	0,17552
10	0,980897	0,8689	0,094145	0,542395
15	0,9119	0,906724	0,057076	0,692689
20	0,870702	0,925672	0,04121	0,789942
22	0,986863	0,934436	0,04063	0,797961
25	0,882412	0,945159	0,033858	0,828923
27	0,869567	0,945211	0,031298	0,842392
30	0,68206	0,943471	0,025137	0,856279
32	0,916667	0,952653	0,024302	0,870382
35	0,839994	0,955039	0,02082	0,881199
37	0,909057	0,95637	0,020359	0,891331
40	0,887376	0,959202	0,018921	0,893496





Analýza FPTAS

item_count	relative_error	max_measured_relative_error	avg_measured_relative_error	avg_time[ms]
	0,1	0,021425	0,000288	1,523912
	0,2	0,197895	0,001096	1,039187
	0,3	0,197895	0,001224	0,7192
	0,4	0,197895	0,003718	0,558511
4	0,5	0,258182	0,006181	0,418535
	0,1	0,01622	0,000148	20,020886
	0,2	0,078431	0,00058	12,142093
	0,3	0,078431	0,001035	8,668218
	0,4	0,078431	0,001736	5,968464
10	0,5	0,102442	0,002845	4,335126
	0,1	0,009777	0,000112	63,008369
	0,2	0,013001	0,000337	40,068404
	0,3	0,064089	0,000662	24,891422
	0,4	0,064089	0,001004	19,612318
15	0,5	0,064089	0,001319	15,426815
	0,1	0,001941	0,000077	171,676342
	0,2	0,017655	0,000229	98,489571
	0,3	0,017655	0,000402	64,791133
	0,4	0,026297	0,000597	49,313236
20	0,5	0,079371	0,000966	33,736134
	0,1	0,007236	0,000079	252,081153
	0,2	0,007236	0,000203	144,339305
	0,3	0,01651	0,000347	83,9031
	0,4	0,01651	0,000502	61,78381
22	0,5	0,019069	0,000643	41,283844
	0,1	0,00085	0,000064	395,883673
	0,2	0,017145	0,000193	195,020061
	0,3	0,017145	0,000279	127,763369
	0,4	0,01769	0,000427	91,304992
25	0,5	0,017145	0,000536	68,110587
	0,1	0,000708	0,000049	508,38764
	0,2	0,003119	0,000158	225,284156
	0,3	0,005409	0,000263	156,429968
	0,4	0,005075	0,000353	121,02214
27	0,5	0,013829	0,000496	100,052581

	0,1	0,000643	0,000048	630,025433
	0,2	0,008142	0,000143	310,941009
	0,3	0,008142	0,000223	226,196843
	0,4	0,008142	0,000295	171,039532
30	0,5	0,008142	0,000384	124,635713
	0,1	0,000496	0,000042	634,933349
	0,2	0,001318	0,000118	395,783814
	0,3	0,001879	0,000196	272,122216
	0,4	0,003943	0,000277	202,263714
32	0,5	0,020924	0,000383	152,615923
	0,1	0,000455	0,000038	695,349177
	0,2	0,002092	0,000102	470,472357
	0,3	0,006166	0,00017	349,406849
	0,4	0,006166	0,000237	275,986883
35	0,5	0,006166	0,000323	213,655086
	0,1	0,000366	0,000033	803,760764
	0,2	0,004175	0,00009	473,522842
	0,3	0,00807	0,000159	455,809392
	0,4	0,00807	0,00021	319,671895
37	0,5	0,00807	0,000263	250,967818
	0,1	0,000407	0,00003	1075,745142
	0,2	0,000908	0,000082	623,060743
	0,3	0,000915	0,000133	503,617797
	0,4	0,001882	0,000185	417,551873
40	0,5	0,001882	0,000243	313,247428

Závěr

Z grafů výpočetních časů je jasně viditelné, že algoritmy Greedy a GreedyOne rostou řádově pomaleji než ostatní algoritmy. Je jasně viditelná mnohem nižší rychlost růstu potřebného výpočetního času algoritmu FPTAS oproti algoritmu DP.

Také je jasně viditelné, že potřebný výpočetní čas algoritmus DPWeight roste pomaleji než algoritmy DP a FPTAS. Tento fakt je způsoben i tím, že v testovacích datech roste celkový součet věcí mnohem rychleji než kapacita batohu. Proto je algoritmus DPWeight mnohem efektivnější.

V *analýze algoritmů Greedy a GreedyOne* vidíme, že výpočetní čas algoritmu GreedyOne roste pomaleji oproti algoritmu Greedy. Ale velikost průměrné relativní chyby algoritmu Greedy s velikostí instance klesá, u algoritmu GreedyOne roste. Protože oba tyto algoritmy mají stejnou výpočetní složitost, rozdíl výpočetních časů není tak významný, a proto je mnohem výhodnější algoritmus Greedy, jehož průměrná relativní chyba s velikostí instance klesá.

V *analýze FPTAS* vidíme, že průměrná velikost relativní chyby se s počtem věcí snižuje. Pro velký počet věcí má tedy FPTAS mnohem nižší výpočetní čas a zároveň se jeho průměrná chybovost blíží nule. Také vidíme, že maximální naměřená relativní chyba nikdy nepřekročila zvolenou maximální relativní chybu, tedy algoritmus FPTAS umožňuje maximální relativní chybu bezpečně volit. Také průměrné velikosti relativních chyb jsou u algoritmu FPTAS mnohem nižší než u algoritmů Greedy a GreedyOne.

Pro n = 40 věcí a zvolené relativní chybě 10 % má algoritmus FPTAS skutečnou průměrnou relativní chybu 0.003 % a zároveň je jeho průměrný výpočetní čas roven cca 16 % průměrného výpočetního času, který vyžadoval algoritmus DP.

Přiložené soubory

Název	Popis
analysisOutput	Složka s výstupnímy hodnotami programu.
data	Sady instancí řešeného *problému*.
src	Zdrojové kódy programu a pomocných skriptů.
tests	Unit testy programu.
analysis.ipynb	Jupyter notebok použitý pro zpracování výstupních dat.