Report: Problém batohu

Specifikace úlohy

Úkoly:

1. Vytvoření programu, který řeší optimalizační *0/1* problém batohu (dále *problém*).
   a. Primárně řeší rozhodovací verzi a optimálně i konstruktivní verzi.
   b. Umí využít tyto algoritmy:
      - Hrubá síla (Brute force)
      - Metoda větví a hranic (Branch & Bounds)
2. Experimentální vyhodnocení výpočetní složitosti na velikosti instance
   a. Vytvořit **diagram růstu** hodnot v závislosti na zvolené hodnotě *N*
   b. Pro jednu z hodnot *N* zjistit četnosti hodnot a vytvořit **histogram**

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**:

- Hrubá síla (dále *BF*)
- Metoda větví a hranic (dále *BB*)
- Neseřazená metoda větví a hranic (dále *UBB*)

U algoritmů *BF* a *BB* jsou v jednotlivých instancích všechny zadané věci seřazeny sestupně podle vyhodnocení $\frac{cena}{hmotnost}$.

Popis algoritmů

Algoritmy prohledávají binární strom, kde:

- Uzly jsou tvořeny zadanými věcmi
- Levé hrany vyjadřují, že daná věc nebyla do batohu přidána
- Pravé hrany vyjadřují, že daná věc byla do batohu přidána

Všechny algoritmy strom ořezávají *shora* (překročení kapacity batohu), algoritmy *BB* a *UBB* ořezávají strom i *zdola* (stávající řešení nemůže být lepší než nejlepší dosud nalezené).

Předpokládané výsledky

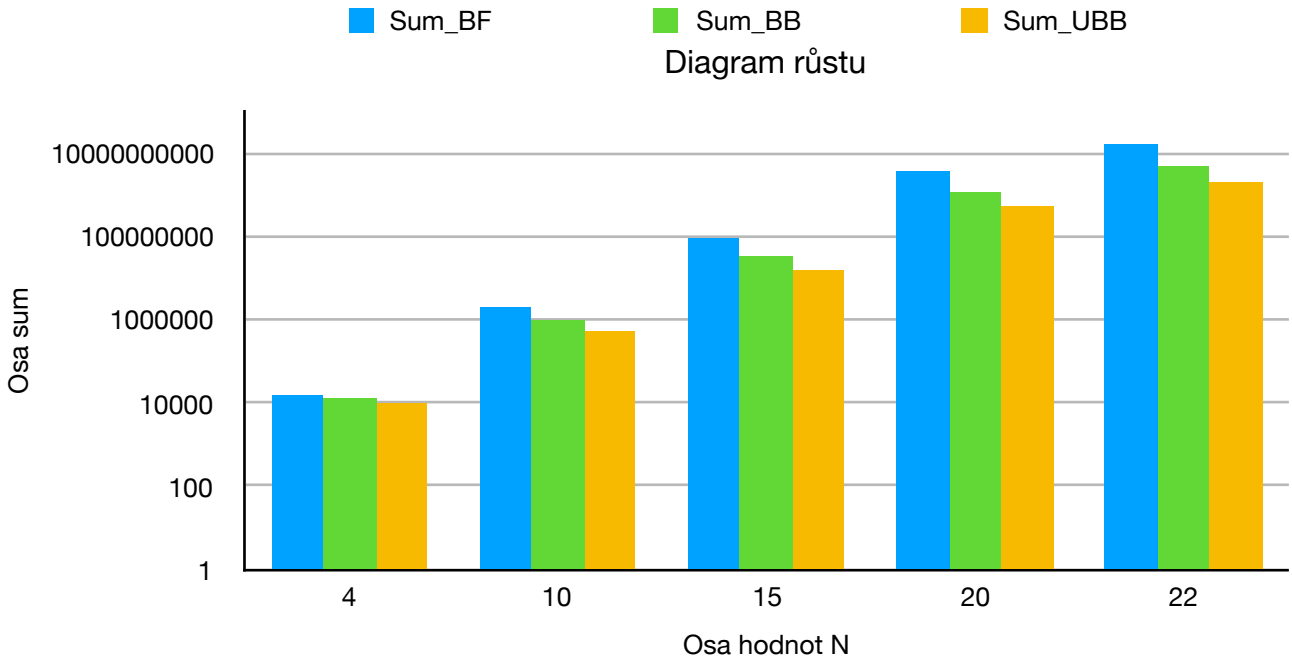
Všechny algoritmy mají v nejhorším případě výpočetní složitost $O(2^N)$. Proto na diagramu růstu očekáváme u všech algoritmů exponenciální růst. Očekáváme, že algoritmus *BF* bude růst rychleji než algoritmy *BB* a *UBB*.

Naměřené výsledky

Diagram růstu

Sumy

N	Sum_BF	Sum_BB	Sum_UBB
4	15780	12232	9019
10	1913365	962925	516218
15	89005050	34506060	16338198
20	3885240002	1189417462	532552802
22	16351272376	4780155214	2043031405

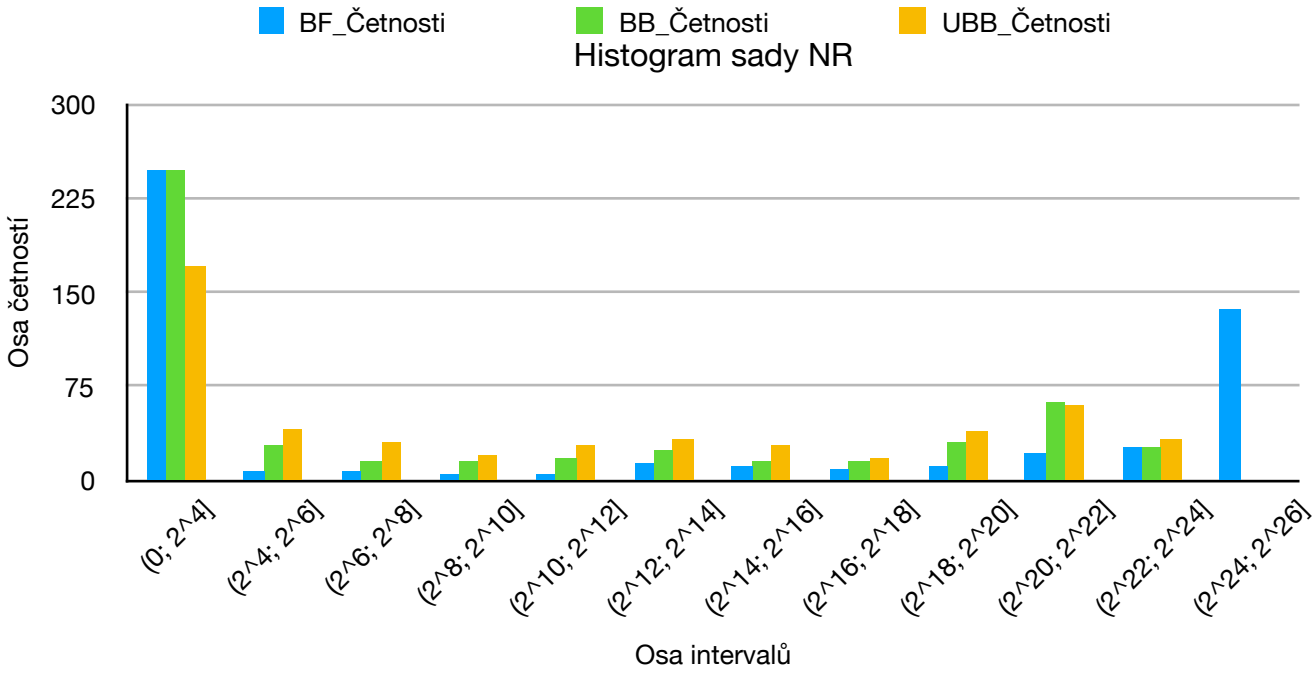


Histogramy

Pro hodnotu *N* = *22* byli vytvořeny histogramy a tabulka četností hodnot.

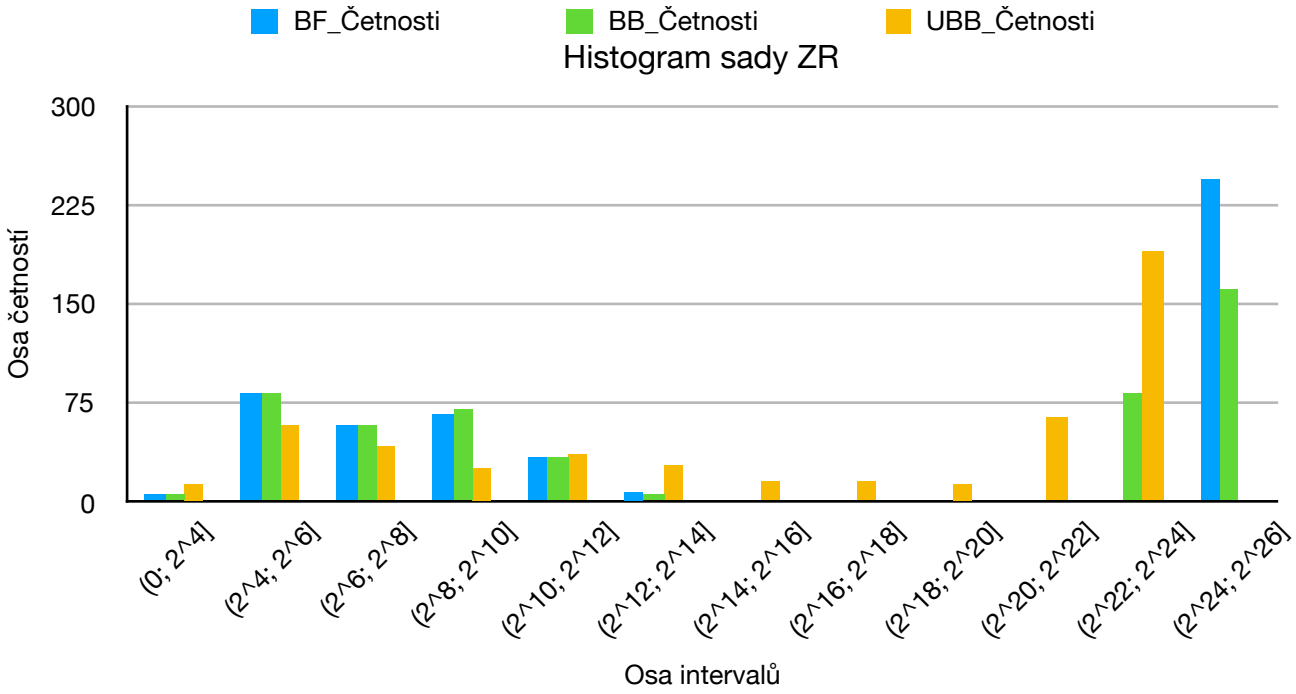
Tabulka četností pro sadu NR

Rozmezí	BF_Configs	BB_Configs	UBB_Configs
(0; 2^4]	248	248	171
(2^4; 2^6]	7	29	40
(2^6; 2^8]	6	15	31
(2^8; 2^10]	5	15	20
(2^10; 2^12]	5	17	27
(2^12; 2^14]	14	24	32
(2^14; 2^16]	10	16	28
(2^16; 2^18]	9	15	17
(2^18; 2^20]	11	31	39
(2^20; 2^22]	21	63	59
(2^22; 2^24]	25	25	33
(2^24; 2^26]	137	0	0



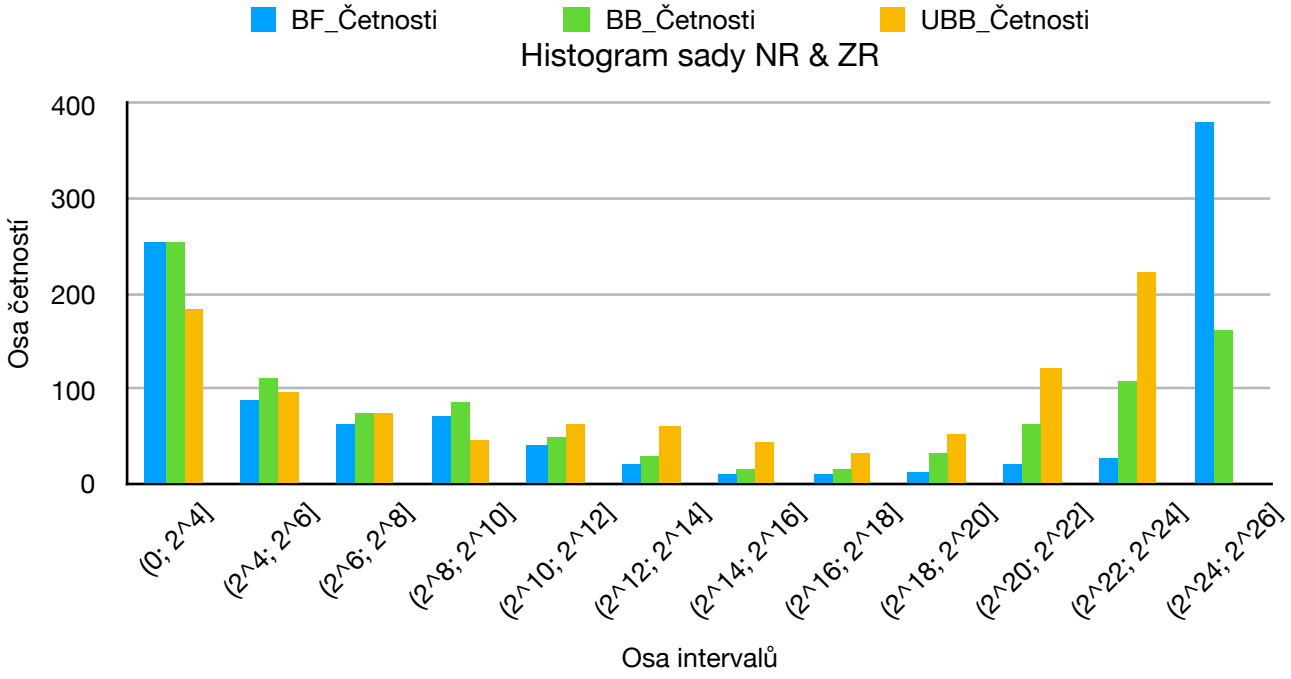
Tabulka četností pro sadu ZR

Rozmezí	BF_Četnosti	BB_Četnosti	UBB_Četnosti
(0; 2^4]	5	5	13
(2^4; 2^6]	83	83	58
(2^6; 2^8]	59	59	43
(2^8; 2^10]	66	70	26
(2^10; 2^12]	34	33	35
(2^12; 2^14]	8	5	27
(2^14; 2^16]	0	0	15
(2^16; 2^18]	0	0	16
(2^18; 2^20]	0	1	13
(2^20; 2^22]	0	1	64
(2^22; 2^24]	0	82	189
(2^24; 2^26]	244	160	1



Tabulka četností pro sady NR a ZR

Rozmezí	BF_Četnosti	BB_Četnosti	UBB_Četnosti
(0; 2^4]	253	253	184
(2^4; 2^6]	89	111	97
(2^6; 2^8]	64	73	74
(2^8; 2^10]	70	84	46
(2^10; 2^12]	39	50	62
(2^12; 2^14]	22	29	59
(2^14; 2^16]	10	16	43
(2^16; 2^18]	9	15	33
(2^18; 2^20]	11	32	52
(2^20; 2^22]	21	64	123
(2^22; 2^24]	25	107	222
(2^24; 2^26]	381	160	1



Závěr

Na *diagramu růstu* je jasně viditelný exponenciální růst všech algoritmů. Také je zřejmé, že algoritmus *BF* roste mnohem rychleji než algoritmy *BB* a *UBB*.

Na *histogramu sady NR & ZR* je patrné, že většina instancí se nachází v prvním a posledním sloupci. Toto je důsledek testování rozhodovací verze programu, která může být pro danou instanci ukončena dříve kvůli nízké vyhledávané hodnotě. Proto se na *histogramu sady ZR* v prvním sloupci nachází velmi málo hodnot, instance této sady byly záměrně zvoleny tak, aby jejich výpočet trval déle.

Z diagramu růstu je viditelné, že algoritmus *BB* roste rychleji než algoritmus *UBB*. *Z histogramu sady NR* je patrné, že u algoritmů *BF* a *BB* řazení lehce zvýšilo počet instancí vyřešených velmi rychle, ale v *histogramu sady ZR* je patrné, že kvůli řazení se u algoritmu *BB* cca 30 % instancí vyhodnotilo v intervalu $(2^{24},2^{26}]$.

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.