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.
      - Rozhodovací verzi umí řešit efektivně.
   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).

Pokud algoritmy BB a UBB řeší rozhodovací problém, tak nejsou prohledávany podstromy, kde maximální výsledná suma hodnot jejich věcí je nižší než cena hledaná.

Pokud je potřeba prověřit oba podstromy nějakého uzlu, algoritmy nejdříve prověří ten podstrom, ve kterém byla věc z daného uzlu do batohu přidaná.

Předpokládané výsledky

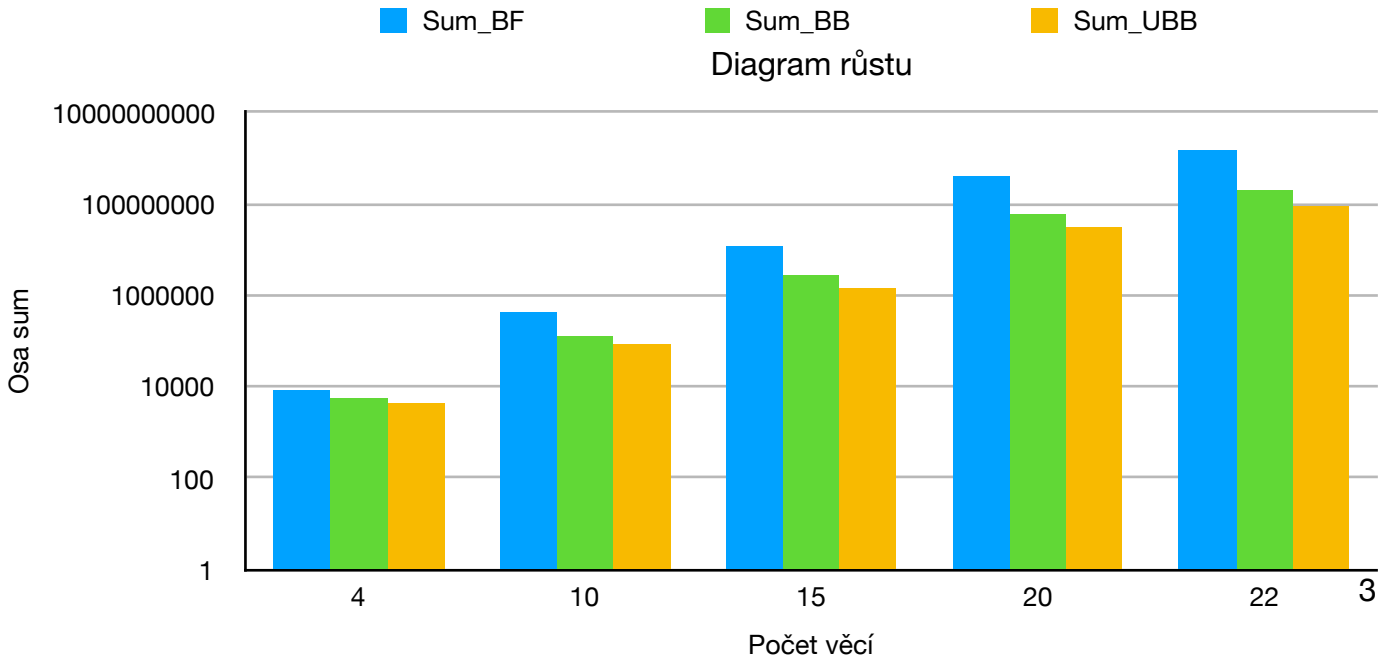
Všechny algoritmy mají v nejhorším případě výpočetní složitost $O(2^N)$. Proto na diagramu růstu očekáváme u všech algoritmů exponenciální růst. Očekáváme, že algoritmus *BF* bude růst rychleji než algoritmy *BB* a *UBB*.

Naměřené výsledky

Diagram růstu

Sumy

N	Sum_BF	Sum_BB	Sum_UBB
4	8429	5372	4581
10	411950	126868	86770
15	12528995	2583261	1449720
20	404835833	56554586	28960921
22	1548769136	195464456	90063209

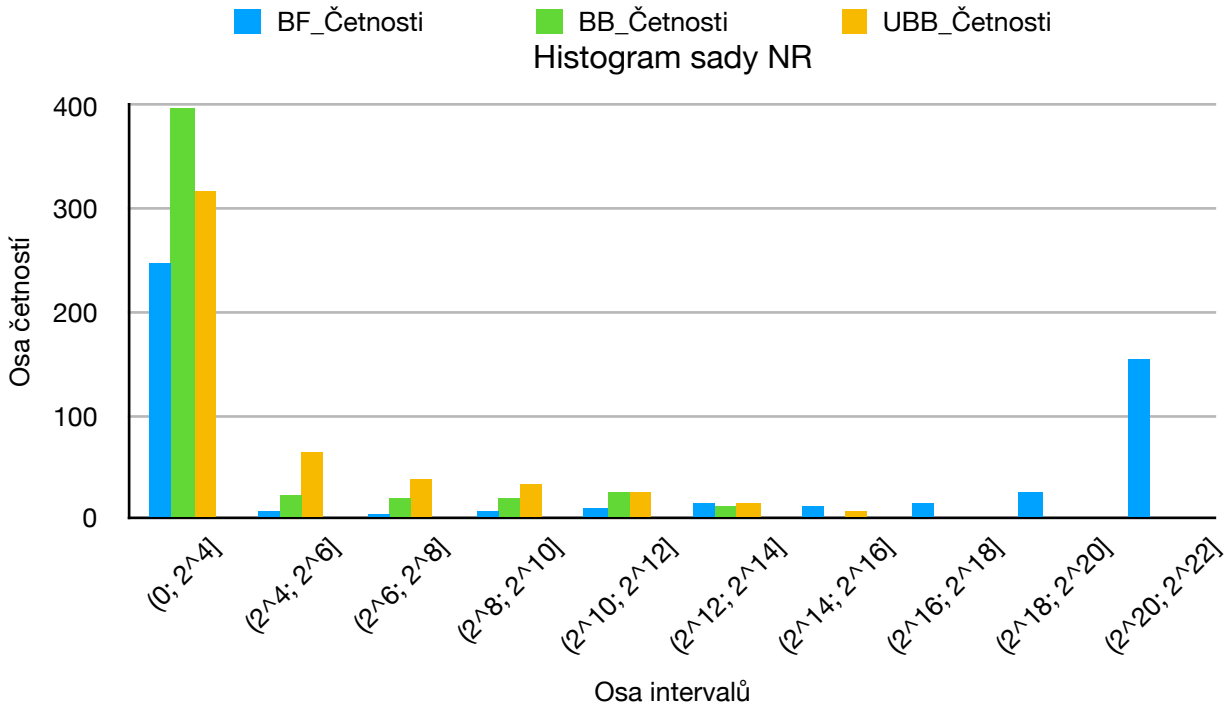


Histogramy

Pro hodnotu *N = 22* byli vytvořeny histogramy a tabulky četností hodnot pro sadu NR, ZR a pro sadu NR & ZR.

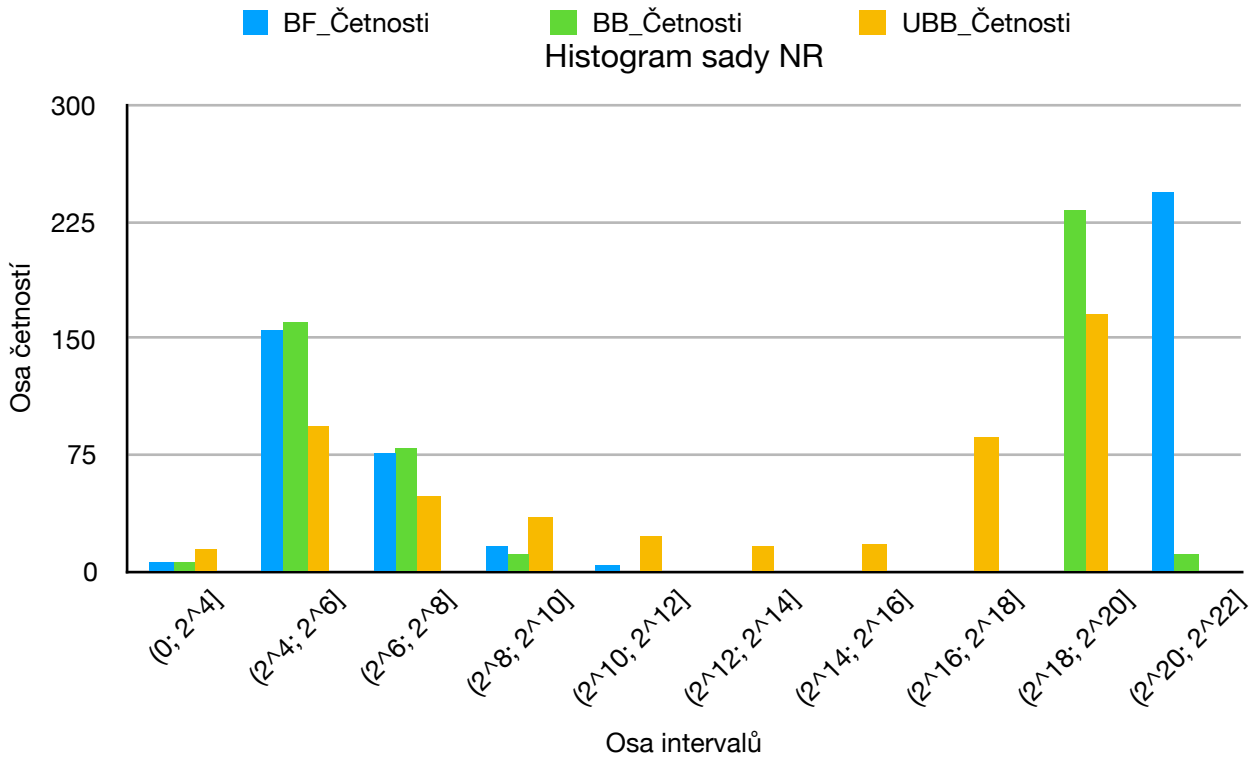
Tabulka četností pro sadu NR

Rozmezí	BF_Četnosti	BB_Četnosti	UBB_Četnosti
(0; 2^4]	248	397	317
(2^4; 2^6]	8	22	65
(2^6; 2^8]	5	20	39
(2^8; 2^10]	7	21	32
(2^10; 2^12]	10	26	25
(2^12; 2^14]	14	13	16
(2^14; 2^16]	12	1	6
(2^16; 2^18]	16	0	0
(2^18; 2^20]	24	0	0
(2^20; 2^22]	154	0	0



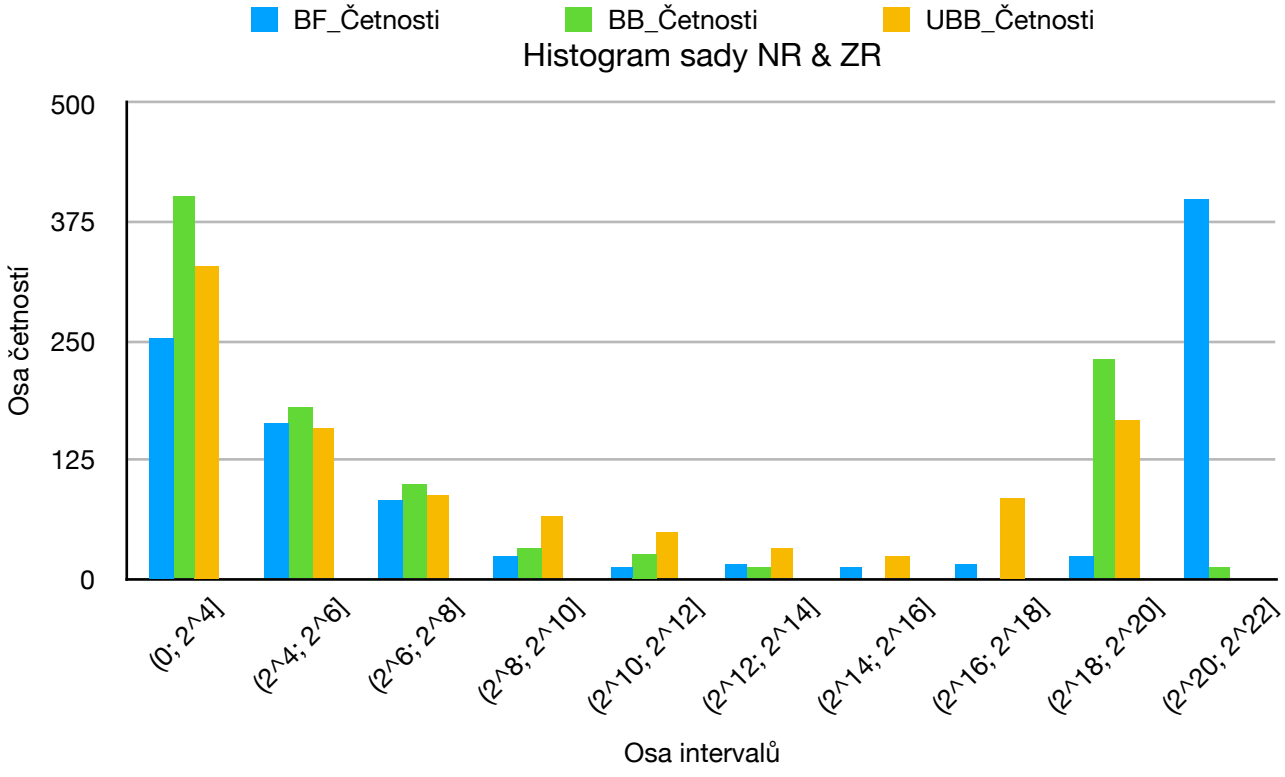
Tabulka četností pro sadu ZR

Rozmezí	BF_Četnosti	BB_Četnosti	UBB_Četnosti
(0; 2^4]	5	5	13
(2^4; 2^6]	155	160	93
(2^6; 2^8]	76	79	49
(2^8; 2^10]	16	11	34
(2^10; 2^12]	3	0	23
(2^12; 2^14]	0	0	16
(2^14; 2^16]	0	0	18
(2^16; 2^18]	0	0	86
(2^18; 2^20]	0	232	166
(2^20; 2^22]	244	11	0



Tabulka četností pro sady NR a ZR

Rozmezí	BF_Četnosti	BB_Četnosti	UBB_Četnosti
(0; 2^4]	253	402	330
(2^4; 2^6]	162	181	157
(2^6; 2^8]	81	99	88
(2^8; 2^10]	23	32	66
(2^10; 2^12]	13	26	48
(2^12; 2^14]	14	13	32
(2^14; 2^16]	12	1	24
(2^16; 2^18]	16	0	86
(2^18; 2^20]	24	231	166
(2^20; 2^22]	398	11	0



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 velká část instancí (cca 25 % – 30 %) se nachází v prvním sloupci. Toto je důsledek testování rozhodovací verze programu, která může být pro danou instanci ukončena dříve kvůli nízké vyhledávané hodnotě a která u algoritmů BB a UBB zásadně prořezává binární strom. 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 trochu rychleji než algoritmus *UBB*. Je možné, že algoritmus BB pracující se seřazenými daty bude fungovat efektivněji, pokud bude zvolen jiný způsob řazení a/nebo jiný způsob vyhodnocování binárního stromu.

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.