Žilinská univerzita

Fakulta riadenia a informatiky

****

**Diskrétna optimalizácia**

**semestrálnA PRÁCA č.2**

2014/2015 Andrej Šišila 5ZI026

Obsah

[Zadanie semestrálnej práce: 2](#_Toc418242379)

[Zadanie H6F 2](#_Toc418242380)

[Popis riešeného algoritmu 2](#_Toc418242381)

[Popis tried 3](#_Toc418242382)

[Atribúty jednotlivých tried 3](#_Toc418242383)

[Atribúty Triedy ‚Prvok‘ 3](#_Toc418242384)

[Atribúty Triedy ‚Batoh‘ 3](#_Toc418242385)

[Atribúty Triedy ‚Main‘ 3](#_Toc418242386)

[Použité metódy v jednotlivých triedach: 3](#_Toc418242387)

[Trieda Prvok 3](#_Toc418242388)

[Trieda Batoh 4](#_Toc418242389)

[Trieda Main 4](#_Toc418242390)

[Výpisy riešení 4](#_Toc418242391)

[Štartovacie (neprípustné) riešenie: 4](#_Toc418242392)

[Riešenie po skončení duálnej heuristiky so stratégiou “Last Admissible“: 5](#_Toc418242393)

[Riešenie po skončení výmennej heuristiky so stratégiou “First Admissible“: 5](#_Toc418242394)

# Zadanie semestrálnej práce:

## Zadanie H6F

a/ **Duálnou heuristikou s výhodnostnými koeficientmi** riešite úlohu danú modelom (úloha o batohu s kapacitou *K* a obmedzeným počtom predmetov v batohu *r*). Riešte úlohu pre *n*=500, *r*=300, *K*=15000 a pre **lokálne kritérium** „*Odstráň prvok z dosiaľ nespracovaných prvkov, ktorý* ***má najmenší pomer koeficientov cj/aj*** *(najmenší výhodnostný koeficient)*“. Východiskové riešenie položte rovné batohu, v ktorom sú vložené všetky prvky (neprípustné riešenie). Súčasťou zadania sú súbory **H6\_a.txt** a **H6\_c.txt**, ktoré obsahujú *n* údajov koeficientov *aj* a *cj* pre *j*=1..*n* potrebných pre riešenie zadanej úlohy.

*Max cj zj*

za podmienok *zj* ≤ *r*

*aj zj* ≤ *K*

*zj ∈ {0,1} ∀ j=1..n*

b/ Riešenie vylepšite výmennou heuristikou so stratégiou najlepší vhodný (best admissible).

# Popis riešeného algoritmu

Našou úlohou bolo riešiť obrátenú úlohu o batohu v ktorom sa muselo nachádzať maximálne 300 prvkov a maximálna hmotnosť batohu (kapacita) sa rovnala 15000.

Najprv používame duálnu heuristiku s výhodnostnými koeficientmi, ktorá štartuje v neprípustnom riešení, kde máme v batohu umiestnené všetky prvky tj. vektor riešení je celý inicializovaný na jednotky. Potom pre docielenie prípustnosti riešenia sa snažíme prvky z batoha vyhadzovať. Prvky vykladám podľa lokálneho kritéria ktoré bolo v danom zadaní pomer ceny a hmotnosti. Cieľom je vyhadzovať prvky, ktoré majú tento pomer čo najmenší.

Pri výpočte sa neustále kontrolujú štrukturálne podmienky. Akonáhle sú porušené, výpočet sa skončí. Výsledkom je optimálne prípustné riešenie.

Potom sme sa snažili hodnotu účelovej funkcie vylepšiť výmennou heuristikou so stratégiou ‘First Admissible‘. Ako vstup sme použili prípustné riešenie vytvorené duálnou heuristikou. Výmenná heuristika spočívala v tom, že sme prechádzali vektor riešení a vymieňali sme predmety tak, že sme si prvky rozdelili na dve podmnožiny: zaradené do batohu a nezaradené. Aby sme jeden prvok z batohu vyhodili a nahradili ho iným zo skupiny nezaradených predmetov, musel vkladaný predmet spĺňať nasledovné podmienky: vkladaný predmet musel byť ľahší alebo rovnako ťažký ako predmet vyberaný z batoha a zároveň cena vkladaného prvku musela byť ostro väčšia ako cena vyhadzovaného prvku.

Po dokončení oboch heuristík na vytvorený batoh sa zadanie a výsledky uložia do súborov a vypíšu na terminál.

# Popis tried

## Atribúty jednotlivých tried

### Atribúty Triedy ‚Prvok‘

**private final int aHmotnost;**

**private final int aCena;**

**private final float aPomerCenaHmotnost;**

**private int aIndexVoVektoreRieseniDH = -1;**

### Atribúty Triedy ‚Batoh‘

**private int[] aVysledneRiesenie; –** vektor riešení pozostáva z núl a jednotiek. Jednotka na n-tom mieste predstavuje, že sa n-tý predmet nachádza v batohu

**private Prvok[] aPrvky;** samotné pole prvkov

**private int aKapacitaBatohu;** K=15000

**private int aCelkovyPocPrvkov;** n=500

**private int aMaxPocPrvkovVBatohu;** r=300

**private int aAktualHmotPrvkovVBatohu = 0; –** aktuálna hmotnosť batohu

**private int aAktualCenaPrvkovVBatohu = 0; –** hodnota účelovej funkcie

**private int aAktualPocPrvkovVBatohu = 0; –** koľko prvkov v batohu aktuálne mám

**private int aPrevis;** aktuálna kapacita batoha – celková kapacita batoha

### Atribúty Triedy ‚Main‘

Trieda ‚Main‘ nemá žiadne atribúty, iba lokálne premenné. Sú to:

**private static int aPocPrvkov = 500;** celkový počet prvkov

**private static int aMaxPocPrvkov = 300;** maximálny počet prvkov v batohu

**private static int aKapacita = 15000;** maximálna hmotnosť batohu

**private static Batoh batoh;** samotný batoh, ktorý chceme naplniť

## Použité metódy v jednotlivých triedach:

### Trieda Prvok

* **public Prvok(int paHmotnost, int paCena)** – parametrický konštruktor triedy Prvok; každý prvok musí mať pri vytvorení svoju hmotnosť a cenu
* **public int dajCenuPrvku()** – vráti cenu prvku
* **public int dajHmotnostPrvku()** – vráti hmotnosť prvku
* **public int dajIndexVoVektoreRieseniDH()** – vráti index prvku vo vektore riešení
* **public float dajPomerCenaHmotnostPrvku()** – vráti pomer ceny a hmotnosti ako desatinné číslo
* **public void nastavIndexZVektoraRieseniDH(int paIndexVoVektoreRieseniDH)** – nastaví atribút aIndexVoVektore RieseniDH na index, na ktorom sa daný prvok vo vektore riešení nachádza

### Trieda Batoh

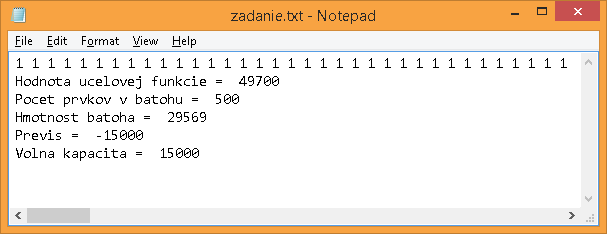
* **Batoh(int paCelkovyPocetPrvkov, int paMaxPocetPrvkovVBatohu, int paKapacitaBatohu)** – vytvorí batoh o zadanom počte prvkov a maximálnou kapacitou a inicializuje prvky,
* **private void nacitaj (int paPocetPrvkov)** – napĺňa pole aPrvky inštanciami objektu ‚Prvok‘, ktorý sa vytvára pomocou hodnôt zo súborov. Súbory sú uložené v koreňovom priečinku programu v priečinku s názvom ‚data‘.
* **private int dajAktualCenaPrvkovVBatohu()** – vracia aktuálnu cenu prvkov v batohu
* **private int dajAktualHmotPrvkovVBatohu()** – vracia aktuálnu hmotnosť prvkov v batohu
* **private int dajAktualPocPrvkovVBatohu()** – vracia aktuálny počet prvkov v batohu
* **private int dajPrevis()** = aktuálna hmotnosť batohu – maximálna kapacita batohu
* **private int dajVolnaKapacita()** – vracia záporne vzatý previs
* **public String dajStav()** – vypíše aktuálny stav atribútov batohu
* **private int dajIndexPrvkuSNajmensimPomeromDualna()** – vráti index prvku s najmenším výhodnostným koeficientom
* **private boolean splnaStruktPodmienky()** – vracia true, ak sú štrukturálne podmienky splnené
* **public int[] dualnaHeuristika()** – vracia vektor riešení po duálnej heuristike so stratégiou Last Admissible
* **public void initVymenna(int[] pole)** – inicializácia výmennej heuristiky: prvky sa rozdelia na *zaradené* a *nezaradené* do batohu.
* **public int[] vymennaHeuristika()**  - vracia vektor riešení po výmennej heuristike so stratégiou First Admissible

### Trieda Main

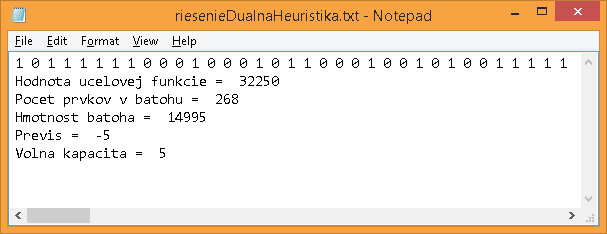
* **public static void main(String[] args)**
* **private static void zapisDoSuboru(String paNazovSuboru, String paCoZapisat)** – metóda zapisujúca do textového súboru s názvom ‚paNazovSuboru‘ to, čo je uvedené v parametri ‚paCoSaMaZapisat‘.

# Výpisy riešení

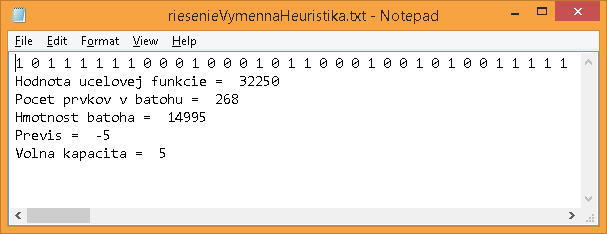
## Štartovacie (neprípustné) riešenie:



## Riešenie po skončení duálnej heuristiky so stratégiou “Last Admissible“:



## Riešenie po skončení výmennej heuristiky so stratégiou “First Admissible“:



Z obrázkov je zrejmé, že duálna heuristika našla také optimálne riešenie, ktoré už vkladacia heuristika nedokázala zlepšiť.