Žilinská univerzita

Fakulta riadenia a informatiky

****

**Diskrétna optimalizácia**

**semestrálnA PRÁCA č.2**

2014/2015 Andrej Šišila 5ZI026

Obsah

[Zadanie semestrálnej práce: 2](#_Toc417217323)

[Zadanie H6F 2](#_Toc417217324)

[Popis riešeného algoritmu 2](#_Toc417217325)

[Popis tried 3](#_Toc417217326)

[Použité atribúty pre celú triedu Batoh: 3](#_Toc417217327)

[Použité metódy v jednotlivých triedach: 3](#_Toc417217328)

[Trieda Prvok: 3](#_Toc417217329)

[Trieda Batoh: 4](#_Toc417217330)

[Trieda Main: 4](#_Toc417217331)

[Výpisy z konzoly 5](#_Toc417217332)

[Štartovacie (neprípustné) riešenie: 5](#_Toc417217333)

[Riešenie po skončení algoritmu “Last Admissible“: 5](#_Toc417217334)

[Riešenie po skončení algoritmu “First Admissible“: 5](#_Toc417217335)

# 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

Mojou úlohou bolo riešiť obrátenú úlohu o batohu v ktorom sa muselo nachádzať maximálne 300 prvkov a hmotnosť batohu musela byť maximálne 15000 tj. kapacita.

Začínalo sa duálnou heuristikou s výhodnostnými koeficientami, ktorá začína v neprípustnom riešení, kde máme v batohu umiestnené všetky prvky tj. vektor zj je celý inicializovaný na jednotky. Potom pre docielenie prípustnosti riešenia sa snažíme prvky vykladať. Prvky vykladám podľa lokálneho kritéria ktoré bolo v mojom zadaní pomer ceny a hmotnosti. Cieľom je vykladať prvky, ktoré majú tento pomer čo najmenší.

Začali som výberom najmenšieho prvku so stratégiou Last Admissible, ktorá spočívala v tom, že som prechádzal vektor zj a pokiaľ sa daný predmet v batohu nachádzal a jeho lokálne kritérium tj. pomer ceny a hmotnosti bol menší alebo rovný doteraz najmenšiemu nájdenému minimu, prvok bol zvolený ako najbližší kandidát na vyhodenie z batohu.

Obdobne som vytvoril prehľadávanie aj stratégiou First Admissible, kedy musel byť pomer ostro menší ako doteraz najmenšie nájdené minimum.

Pokračoval som samotným výpočtom, kedy som na začiatok otestoval či sú splnené štrukturálne podmienky a pokiaľ neboli pokračoval som samotným vylúčením prvku z batohu. K tomuto postupu ma viedol fakt, že duálna heuristika začína vo počiatočnom neprípustnom riešení.

Po aplikovaní obidvoch stratégii duálnej heuristiky na vytvorený batoh sa výsledky hmotnosti batohu, vektor zaradených prvkov, hodnotu účelovej funkcie vypíše na konzolu. Úloha je riešená v programovacom prostredí NetBeans.

# Popis tried

V programe sa nachádzajú tri triedy. Trieda Batoh je najdôležitejšou triedou v ktorej je implementovaná samotná duálna heuristika. Inicializácia zo súboru do príslušných hlavných atribútov a hľadanie najlepšieho prípustného riešenia. Metóda vypisRiesenie ktorá sa následne používa v triede Main na výpis najlepšieho riešenia podľa oboch stratégii na konzolu. V triede Prvok sa nachádzajú atribúty vlastné pre každý prvok v batohu spolu s metódami na ich nastavenie a vrátenie, vloženie a vyloženie prvku z batohu a celkový výpis atribútov prvku metódou toString.

## Použité atribúty pre celú triedu Batoh:



## Použité metódy v jednotlivých triedach:

### Trieda Prvok:

* **public Prvok(int paHmotnost, int paCena)** – konštruktor triedy Prvok
* **public int dajHmotnostPrvku()** – vráti Hmotnosť
* **public int dajCenuPrvku()** – vráti cenu
* **public float dajPomerCenaHmotnostPrvku()** – vráti pomer ceny a hmotnosti tj. lokálne kritérium
* **public boolean patriPrvokDoBatohu()** – vráti či je prvok v batohu alebo nie
* **public void vyberPrvokZBatohu()** – prvok sa vyberie z batohu
* **public void vlozPrvokDoBatohu()** – prvok sa vloží do batohu
* **public String toString()** – Vypíše informácie o prvku

### Trieda Batoh:

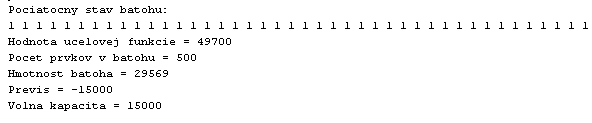
* **public Batoh(int paCelkovyPocetPrvkov, int paMaxPocetPrvkovVBatohu, int paKapacitaBatohu)** – konštruktor triedy batoh
* **public int dajAktualHmotPrvkovVatohu()** – vráti aktuálnu hmotnosť prvkov v batohu
* **public int dajAktualCenaPrvkovVatohu()** – vráti aktuálnu cenu prvkov v batohu
* **public int dajAktualPocPrvkovVatohu()** – vráti aktuálny počet prvkov v batohu
* **public int dajPrevis()** – vráti aktuálny previs (voľnú kapacitu) batohu
* **public void nacitaj(int paPocetPrvkov)** – načítanie hmotností a cien prvkov zo súborov
* **private int dajIndexPrvkuSNajmensimPomeromLA()** – vráti index prvku ktorý má spomedzi všetkých vložených prvkov v batohu má najlepší pomer ceny a hmotnosti a ten je menší alebo rovný doteraz nájdenému minimu
* **private int dajIndexPrvkuSNajmensimPomeromFA()** – vráti index prvku ktorý má spomedzi všetkých vložených prvkov v batohu má najlepší pomer ceny a hmotnosti a ten je ostro menší od doteraz nájdeného minima
* **private boolean splnaStruktPodmienky()** – vráti či sú štrukturálne podmienky splnené alebo nie
* **public int[] vypocitajPoA()** – uskutoční sa výpočet s výberom prvku so stratégiou Last Admissible, uskutočnuje výber prvkov kým nie sú splnené štrukturálne podmienky a vráti sa pole výsledného riešenia.
* **public int[] vypocitajPoB()** – uskutoční sa výpočet s výberom prvku so stratégiou First Admissible, uskutočnuje výber prvkov kým nie sú splnené štrukturálne podmienky a vráti sa pole výsledné riešenie.
* **private void dokonciTaktVypoctu(int paVyhadzovany)** – upraví vektor výsledného riešenia po vylúčení prvku, aktuálnu hmotnosť a cenu prvkov v batohu a previs.

### Trieda Main:

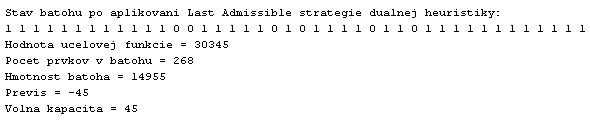
* **public static void main(String[] args)**
* **public static String vypisRiesenie(int [] pole)** – metóda vypisujúca riešenie na konzolu

## Výpisy z konzoly

### Štartovacie (neprípustné) riešenie:



### Riešenie po skončení algoritmu “Last Admissible“:



### Riešenie po skončení algoritmu “First Admissible“:

