

FLP 2022/2023 – funkcionální projekt: Haskell

Daniel Uhříček
iuhricek@fit.vut.cz

Úvod

Toto je zadání funkcionálního projektu do předmětu Funkcionální a logické programování 2022/2023. Za projekt zodpovídá Ing. Daniel Uhříček, konzultace jsou možné kdykoliv po předchozí domluvě.

Obecné požadavky

Svá řešení odevzdávejte v předepsaném tvaru (zkomprimovaný archiv zip, viz Závazné pokyny pro projekty) prostřednictvím VUT IS. Odevzdaný projekt musí obsahovat zdrojové texty v Haskellu. V hlavní složce musí být soubor `Makefile`, který program přeloží a výsledný binární kód umístí také do hlavní složky. Dále se doporučuje přiložit stručný popis všeho, co nebylo dořešeno nebo naopak bylo implementováno nad rámec zadání.

Například soubor `flp-fun-xlogin00.zip` po rozbalení obsahuje:

```
Makefile
src
├── Main.hs
├── Minimize.hs
├── ParseInput.hs
└── Types.hs
doc
├── README.md
└── test-description.txt
test
├── test01.in
├── test01.out
├── test02.in
└── test02.out
```

Pro překlad použijte kompilátor `ghc` s volbou `-Wall`. Cílový program pojmenujte **flp22-fun**. Můžete využít standardní knihovny z balíku `base`, případně knihovny z balíků `containers`, `parsec`, `vector`, `split`, `directory`, a `random`. Za referenční verzi Haskellu je považována verze 7.6.3 (server merlin) nebo verze 9.2.5. Před odevzdáním si ověřte, zda lze zip rozbalit a program přeložit a spustit, například:

```
unzip flp-fun-xlogin00.zip
make
./flp22-fun -2 test.in
```

Hodnocení

Za vypracovaný projekt lze získat až 12 bod. Hodnocena bude míra splnění zadání, kvalita řešení, čistota a kvalita kódu. Za obzvláště kvalitní řešení lze získat prémiové body navíc. Vyvarujte se nestandardních funkcí, které obcházejí bezpečné otypování nebo obcházejí zapouzdření vedlejších efektů – nepoužívejte nic, co má ve svém jméně slovo `unsafe`. Snažte se psát čistý kód respektující různé varování a připomínky, které zazněly na přednáškách a cvičeních. Prosím také o dodržení, ať: úvodní řádky zdrojových kódů obsahují název projektu, login, jméno autora a rok řešení; součástí definic na globální úrovni jsou typové anotace a stručný a výstižný komentář; je zřejmé, co a z jakých modulů importujete¹. Projekt vypracovávejte samostatně.

Následuje popis jednotlivých zadání.

¹https://wiki.haskell.org/Import_modules_properly

2 Knapsack problem

Vytvořte program, který řeší optimalizační verzi 0-1 problému batohu (knapsack problem).

2.1 Rozhraní programu

Program bude možné spustit:

```
flp22-fun volby [vstup]
```

kde

- *vstup* je jméno vstupního souboru (pokud není specifikován – stdin)
- *volby* jsou parametry ovlivňující chování programu:
 - i ze vstupu načte informace o instanci knapsack do vaší vnitřní reprezentace. Na stdout jí vypíše zpět (očekává se, že tento výpis bude řešen instancí typové třídy **Show** pro váš datový typ reprezentující knapsack).
 - b ze vstupu načte informace o knapsack instanci. Na stdout vypíše řešení nalezené prohledáváním stavového prostoru hrubou silou. V případě, že řešení nebylo nalezeno, vypíše **False**.
 - o ze vstupu načte informace o knapsack instanci. Na stdout vypíše řešení nalezené některou z následujících optimalizačních metod: (1) genetic algorithm, (2) simulated annealing, (3) ant colony optimization, nebo (4) particle swarm optimization. Algoritmy by mimo jiné měly být známé z předmětů IZU, SFC, EVO, a SUI. V případě, že řešení nebylo nalezeno, vypíše **False**.

2.2 Očekávaný vstup a výstup

V následujících ukázkách je <informace o knapsack instanci> zaměněno za:

```
Knapsack {  
  maxWeight: 46  
  minCost: 324  
  items: [  
    Item {  
      weight: 36
```

```
    cost: 3
  }
  Item {
    weight: 43
    cost: 1129
  }
  Item {
    weight: 202
    cost: 94
  }
  Item {
    weight: 149
    cost: 2084
  }
]
}
```

2.2.1 -i

Vstup:

<informace o knapsack instanci>

Výstup:

<informace o knapsack instanci>

2.2.2 -b

Vstup:

<informace o knapsack instanci>

Výstup:

Solution [0 1 0 0]

2.2.3 -o

Stejný formát jako v případě -b.