Fórum studentů MFF UK

Fórum pro všechny studenty matematicko-fyzikální fakulty UK, informatiky, fyziky i matematiky

Přejít na obsah

Hledej… Hledat
Pokročilé hledání

- Obsah fóra ‹ Informatika LS ‹ Výuka LS 2. ročník ‹ PRG005 Neprocedurální programování
- Změnit velikost textu
- Napsat e-mail
- Verze pro tisk

- FAQ
- Registrovat
- Přihlásit se

Zkouška 29.5.2017

Odeslat odpověď

Hledat v tomto tématu Hledat

Příspěvek: 1 • Stránka **1** z **1**

- Ohlásit tento příspěvek
- Odpovědět s citací

Zkouška 29.5.2017

od **vasek.rozhon** » 1. 6. 2017 22:01

Přikládám čtyři otázky z první části zkoušky.
Ve druhé části byl zadaný n-regulární hypergraf s m vrcholy takový, že každá dvojice hyperhran měla v průniku max. jeden vrchol a a) jsme chtěli najít validní obarvení takového hypergrafu a b) jsme chtěli všechny takové n-regulární hypergrafy na m vrcholech generovat. Měla se použít nějaká heuristika.

První část zkoušky:

Otázka 1)
Sestavte predikát termy/1, který postupně vrací termy složené z funktorů bin/2, un/1 a const/0. Výstupem bude tedy korektně sestavený term. Predikát by měl postupně vrátit všechna řešení, sice v libovolném pořadí, ovšem každé právě jednou.
termy(V).
V=const;
V=un(const);
V=bin(const,const);
V=un(un(const));
V=un(bin(const,const));
V=bin(un(const),un(const));

atd.

Otázka 2)
Multimnožinu lze specifikovat seznamem termů Prvek-Pocet. Sestavte predikát mensi/2, který porovná multimnožiny A a B následovně: mensi(A,B) je True právě tehdy, pokud v B existuje nějaký prvek, co není v A takový, že je větší než všechny prvky z A, které nejsou v B.
mensi([c-3,b-2,a-1],[d-1,b-3]) True
mensi([c-3,b-2,a-1],[c-1,b-3]) False

Otázka 3)
Navrhněte datový typ Graf a pro reprezentaci konečného neorientovaného grafu s vrcholy typu a.
Definujte funkci troj::Graf a -> Int, která k takovému grafu vrátí počet všech jeho trojúhelníků.

Otázka 4)
Dán datový typ
data Bag a = Item a | Items [Bag a]
a) Definujte funkci
fold :: (a->b) -> ([b]->b) -> Bag a -> b
pro obecný průchod touto datovou strukturou (to (a->b) tam zastupuje počáteční hodnotu v normálním foldu)
b) Pomocí funkce fold definujte funkci
listy::Bag a -> [a]
která posbírá všechny hodnoty z položek Item ze všech úrovní zleva doprava
listy (Items [Item 1,Items [Item 2, Item 3], Items [Items [Item 4]]])
[1,2,3,4]

ŘEŠENÍ:

% abychom opravdu nagenerovali kazdy term, tak budeme postupne generovat vsechny termy velikosti 1, pak velikosti 2,3, atd.
termy(V) :- termy(V,1).
termy(V,N) :-
stejnevelke(V,N).
termy(V,N) :-
NN is N+1,
termy(V,NN).

%velikost 1 ma jen konstantni term
stejnevelke(const,1).

% pro velikost aspon dva generujeme termy s vnejsim funktorem un
stejnevelke(un(T),N) :-
N>1,
NN is N-1,
stejnevelke(T,NN).

% pro velikost aspon tri generujeme termy s vnejsim funktorem bin, vnitrni termy nabyvaji v souctu velikosti N-1
stejnevelke(bin(T1,T2),N) :-

```
N>2,
NN is N-2,
vsechnacislado(NN,Cisla),
member(M,Cisla),
O is N-1-M,
stejnevelke(T1,M),
stejnevelke(T2,O).

%vygeneruje seznam vsech cisel od 1 do N
vsechnacislado(1,[1]).
vsechnacislado(N,[N|List]) :-
N>1,
NN is N-1,
vsechnacislado(NN,List).
```

```
%nejprve spocitame rozdil seznamu, nasledne v obou rozdilech najdeme nejvetsi prvek a ty nakonec porovname
mensi(A,B) :-
rozdil(A,B,R1),
rozdil(B,A,R2),
nejvetsi(R1,MA),
nejvetsi(R2,MB),
MB@>MA.

%rozdil(+A,+B,-Res), rozdil obsahuje prvky, co jsou v A a ne v B (uz nepotrebujeme jejich pocty)
rozdil([], _, []).

%B neobsahuje prvni prvek z A, pridame ho tedy do vysledku
rozdil([K-_|T], B, [K|Res]) :-
\+ member(K-_, B),
rozdil(T,B,Res).

%B obsahuje prvni prvek z A, ale je ho tam mene -- stejne jako v predchozim pripade prvek pridame do vysledku
rozdil([K-APocet|T], B, [K|Res]) :-
member(K-BPocet, B),
APocet>BPocet,
rozdil(T,B,Res).

%B obsahuje prvni prvek z A a je ho tam aspon tolik, co v A
rozdil([K-APocet|T], B, Res) :-
member(K-BPocet, B),
APocet=<BPocet,
rozdil(T,B,Res).

% ze seznamu termu vybereme ten nejvetsi
nejvetsi([],0). %tady by se hodilo mit term, ktery je v @usporadani nejmensi; jestlize takovy neexistuje, porad by se to dalo pro neprazdne seznamy zachranit tim, ze rekurzi ukoncime na jednoprvkovych seznamech, tedy pridame nejvetsi([A],A).
nejvetsi([H|T], Res) :-
nejvetsi(T,Ress),
maximum(H,Ress,Res).
```

```
maximum(T1,T2,T1) :- T1 @>= T2.
maximum(T1,T2,T2) :- T1 @< T2.
```

```
data Graf a = G ([a], [(a,a)]) -- dvojice obsahuje seznam vrcholu typu a a pak seznam hran, kde kazda hrana je dvojice acek

troj :: Eq a => Graf a -> Int
troj g = troj' 1 1 1 g

troj' ::Eq a => Int -> Int -> Int -> Graf a -> Int -- dostane tri vrcholy (rsp. indexy do seznamu vrcholu) a vrati pocet trojuhelniku takovych, ze jejich tri vrcholy jsou v seznamu vrcholu grafu lexikograficky >= teto trojici (pro ty tri promenne plati nerovnosti u<=v<=w, specialne kazdy trojuhelnik zapocitame prave jednou)
troj' u' v' w' (G (vrcholy, hrany)) =
let
n=length vrcholy
u=last $ take u' vrcholy
v=last $ take v' vrcholy
w=last $ take w' vrcholy
tentotrojuhelnik = if ((obsahujehranu (u,v) hrany) && (obsahujehranu (w,v) hrany) && (obsahujehranu (u,w) hrany)) then 1 else 0 -- podivame se, zda stavajici vrcholy tvori trojuhelnik
in tentotrojuhelnik +
(if (u'==n)
then 0
else (
if (v'==n)
then (troj' (u'+1) (u'+1) (u'+1) (G (vrcholy, hrany)))
else (
if (w'==n)
then (troj' u' (v'+1) (v'+1) (G (vrcholy, hrany)))
else (troj' u' v' (w'+1) (G (vrcholy, hrany))))))
-- podle toho, ktere promenne jsou uz na konci se zeptame na trojici, ktera nasleduje lexikograficky za tou aktualni

obsahujehranu :: Eq a => (a,a) -> [(a,a)] -> Bool -- vrati, zda seznam obsahuje zadanou hranu
obsahujehranu e [] = False
obsahujehranu (u,v) ((x,y):r) = if ((u==x && v==y) || (u==y && v==x)) then True else (obsahujehranu (u,v) r)
```

```
data Bag a = Item a | Items [Bag a]

fold :: (a->b) -> ([b]->b) -> Bag a -> b
fold f1 f2 (Item x) = f1 x -- v listu aplikujeme f1
fold f1 f2 (Items l) = f2 (map (fold f1 f2) l) -- ve vnitrnich vrcholech na syny rekurzivne mapujeme fold a vysledky skladame funkci f2

listy :: Bag a -> [a]
listy t = fold (\x -> [x]) (foldl (++) []) t -- v listech vyrobime singletonovy seznam, ve vnitrnich vrcholech spojujeme seznamy do kupy, foldl je klasicky fold pro seznamy
```

[vasek.rozhon]

Matfyz(ák|ačka) level I

Příspěvky: 5
Registrován: 25. 1. 2017 16:34
Typ studia: Informatika Bc.

Nahoru

Odeslat odpověď
Příspěvek: 1 • Stránka **1** z **1**

Zpět na PRG005 Neprocedurální programování

Přejít na: PRG005 Neprocedurální programování Přejít

Kdo je online

Uživatelé procházející toto fórum: Žádní registrovaní uživatelé a 1 návštěvník

- Obsah fóra
- Tým • Smazat všechny cookies z fóra • Všechny časy jsou v UTC + 1 hodina