Martin Števko

MFF UK, Informatika, 1. ročník

Úloha 1.

Pre nájdenie najdlhšej neklesajúcej vybranej podpostupnosti v $O(n^2)$ budem postupovať od konca postupnsti. Zoberiem posledný nevyriešený prvok a pozriem sa na jeho hodnotu. Nájdem všetky prvky za ním (teda s vyšším indexom), ktoré sú väčšie alebo rovné než on. Vyberiem z nich prvok, ktorý má maximálnu dĺžku vybranej podpostupnosti začínajúcej týmto prvkom, túto hodnotu zvýšim o 1 a uložím do pomocného poľa s indekom prvku, ktorý práve riešim.

Týmto som k nemu uložil dĺžku maximálnej neklesajúcej vybranej podpostupnosti začínajúcej týmto prvkom, pretože postupnosť začína ním a pokračovať môže iba väčšími, alebo rovnými prvkami, na ktorých maximálne dĺžky vybraných podpostupností som sa už pozrel.

Takto vyrátam dĺžky maximálnych vybraných podpostupností pre každý prvok a na konci vyberiem maximum z týchto hodnôt. To je evidentne aj úplné maximum. Ak by sme naviac chceli aj vybrať prvky tejto podpostupnosti, začneme tentokrát zo začiatku, kde si nájdeme prvok s maximálnou hodnotou dĺžky vybranej podpostupnosti, vyberieme ho, nájdeme prvý väčší prípadné rovný prvok s hodnotou o 1 nižšou, vyberieme ho, atď.

Martin Števko

MFF UK, Informatika, 1. ročník

Úloha 2.

Nezávislosť je naprogramovaná v súbore *nezavislost.py* a dominancia zase v *dominancia.py*. Oba algoritmy by sa dali optimalizovať ešte viac, ak by som o vstupe vedel viacero údajov – napríklad, že šachovnica bude štvorcová. Pre rozmery šachovnice x a y, typ figúrky ľubovoľný okrem pešiaka a šachovnicu ako pneumatiku sa mi to podarilo optimalizovať najviac takto.

Využil som rekurzívne volanie funkcie na skúšanie všetkých perspektívnych možností, teda tých, ktoré ešte nie sú ohrozené žiadnou figúrkou. Pre každú figúrku som nastavil aj základný matematický invariant, ktorý určuje minimum/maximum, akurát nedokazuje, že je dosiahnuteľné a teda ak program nájde nejaké usporiadanie pre invariant, môže skončiť.

Využité invarianty:

- v jednom riadku ani stĺpci nemôžu byť 2 veže ani kráľovné,
- na podpolíčku 2×2 nemôžu byť 2 králi,
- každý kôň ohrozuje 8 políčok rozdielnej farby a každé políčko je ohrozované maximálne 8 koňmi
- najdlhšia uhlopriečka prechádza cez počet políčok rovný menšej strane šachovnice
- žiaden kráľ ani kôň neohrozí viac než 8 políčok
- žiadna veža neohrozí 2 riadky ani 2 stĺpce.

Martin Števko

MFF UK, Informatika, 1. ročník

Úloha 3.

Najprv dokážem, že pre konečný počet nádob existuje konečný počet stavov v ktorých môžu byť. Zoberme si 2 nádoby s kapacitami k a l a objemom vody v nich a a b. Potom prelievaním viem dostávať iba stavy:

- k a $a+b-k$,
- l a $a+b-l$,
- 0 a $a+b$,
- a a b (ak vodu neprelejem).

Avšak keďže nádob je konečný počet, mám konečný počet kapacít aj objemov a teda aj rôznych možných súčtov objemov, čo znamená, že aj konečný počet stavov.

Potom keďže chcem nájsť najrýchlejší spôsob prelievania, môžem použiť prehľadávanie grafu do šírky, pričom vrcholmi budú stavy nádob a hrany budú popisovať zmenu stavu, ktorú viem spraviť na 1 preliatie. (Analogicky cesty grafu na n preliatí.)

Stav v ktorom sa aktuálne nachádzam nech je S, pole prejdených stavov nazvem $CLOSED$ a frontu (pole stavov na prejdenie) nazvem $OPENED$. Na začiatku bude S rovné vstupu, $CLOSED$ prázdne a v $OPENED$ bude len S. Aloritmus bude vyzerať takto:

- Začni cyklus,
- ak je $OPENED$ prázdne, vypíš, že sa prelievaním nedá dostať do cieleného stavu a skonči,
- ináč odstráň prvý prvok z $OPENED$, ulož ho do S a pridaj ho na koniec do $CLOSED$,
- ak sa v S nachádza nádoba, ktorá má aktuálny objem vody taký, aký chceme dostať skonči a vypíš počet preliatí na ktoré sme sa do stavu dostali, ináč pokračuj,
- pre všetky rôzne dvojice nádob zisti do akého stavu sa preliatím vieme dostať (vždy to je z pravidla práve 1 stav z vyššie popísaných) a ak nie je v $OPENED$ ani $CLOSED$, pridaj ho na koniec do $OPENED$, ak je v $OPENED$ pozri sa na počet preliatí ktorým sa doň vieme dostať a ak je ten aktuálny počet menší, prepíš ho, ak je v $CLOSED$ pozri sa na počet preliatí ktorým sa doň vieme dostať a ak je ten aktuálny počet menší, prepíš ho,
- pokračuj na začiatok cyklu.

Takto nájdeme najkratší možný počet preliatí na ktorý sa do výsledného stavu vieme dostať ak existuje.

Martin Števko

MFF UK, Informatika, 1. ročník

Úloha 4.

Vytvorím si 2 polia, ktoré budem používať ako zásobníky. V jednom budem udržiavať nevyhodnotené znamienka a v druhom nevyhodnotené čísla.

Vstup budem čítať po znakoch, ak dostanem znamienko pozriem sa čo je na vrchu zásobníku. Ak je to znamienko s vyššou alebo rovnakou prioritou, vyhodnotím najprv to a potom zopakujem pokus o vloženie, až kým ho nevložím. Ak prečítam zo vstupu ľavú zátvorku, dám ju ku znamienkam, ak pravú, nájdem jej ľavú zátvorku v zásobníku a vyhodnotím znamienka medzi nimi. Ak dostanem číslo, vložím ho do zásobníka. Následne, akonáhle prečítam celý vstup, vyhodnotím každé znamienko v zásobníku postupne.

Zátvorku budem považovať za najvyššiu prioritu.

Program je v súbore "infix.py". Ako znamienka môžu byť uvedené $+$, $-$, $*$ a $/$, pričom delenie je len celočíselné (no nie je problém rozšíriť na klasické, akurát by som musel celý čas rátať s typom float). Výraz môže takisto obsahovať ľubovoľné uzátvorkovanie, ale iba z okrúhlych zátvoriek (,) (pre jednoduchosť). Čísla môžu byť ľubovoľné kladné, záporné program na vstupe nezvládne, avšak rátať s nimi dokáže.