KSI 2014/2015

Úloha 2-5: Sobí stáj

Jan Horáček
Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

14. prosince 2014

1 Úvod

Při analýze zadaného problému jsem došel k následujícím závěrům:

1. K ukládání sobů se vyloženě hodí `MinHalda`, která efektivně umožňuje řazení sobů dle priorit.
2. Implementovat v programu podporu pro příkaz `udelej_obraz` není (oproti zbytku příkazů) vůbec triviální problém. Tento problém lze řešit v podstatě dvěma cestami:
   (a) pamatovat si historii příkazů a při každém požadavku stav haldy v daném kroku rekonstruovat,
   (b) pamatovat si stav haldy po každém kroku,

   či kompromis mezi těmito dvěma přístupy.

   Přemýšlel jsem například o tom, že bych si pro každý k-tý příkaz ukládal stav haldy a stavy, které nejsou k-násobky bych rekonstruoval pomocí uloženého seznamu příkazů. Toto řešení jsem uvažoval pro to, že velikost haldy je řádově větší, než velikost seznamu příkazů. Na druhou stranu, rekonstrukce haldy z daných příkazů trvá dlouho — každý příkaz přidání či odebrání soba trvá $O(log(n))$, kde n je aktuální počet prvků haldy. A příkazů může být opravdu hodně.

2 Popis řešení

Po uvážení výše zmíněných přístupů k problému jsem se nakonec rozhodl pro ryzí ukládání kopie haldy po každém příkazu. A to především z toho důvodu, že obecně platí "čas procesoru je dražší, než paměť".

Samotný algoritmus je pak poměrně triviální: na základě příkazu v parametru provedeme buď přidání sobů do haldy, nebo odebrání sobů z haldy. Samotná halda zajišťuje to, že odebereme vždy ty soby, které mají nejnižší prioritu.

Po každém příkazu si uložíme hloubkovou kopii haldy do listu. Při požadavku na výpis stavu stáje (haldy) po n krocích pak jednoduše vrátíme $n-1$. prvek tohoto listu.

3 Závěr

Paměťová složitost řešení je definována počtem prvků v listu historie příkazu. V našem případě je lineární vůči počtu příkazů: $O(n)$, kde n je počet příkazů.

Časová složitost se liší u jednotlivých operací:

- Časová složitost přidání soba do haldy je dána implementací a principem haldy. Pakliže platí, že halda je v `binary2_5` implementována opravdu jako halda, je náročnost přidání prvku v O notaci $O(log(n))$, kde n je aktuální počet prvků haldy. Pro přidání k sobů do haldy je tedy časová složitost této operace $O(\sum_{i=1}^{k} log(n_i))$.
- Podobně časová složitost odebrání k sobů z hlady plyne z vlastností haldy a je dána stejným vztahem $O(\sum_{i=1}^{k} log(n_i))$.
- Časová složitost příkazu `vypis_epochu` je konstantní ($O(1)$), protože jednoduše stačí přistoupit k prvku v seznamu o daném indexu. To je taky hlavní důvod, proč jsem tuto metodu vybral.

V případě zájmu o zmenšení absolutní paměťové náročnosti lze při hloubkové kopii haldy výstup ukládat například do listu. Tím ušetříme místo na ukazatele. Tuto možnost zadání připouští díky tomu, že není striktně dáno, že záznam stáje musí být seřazen dle priorit sobů. Pro implementaci této funkcionality bych ale musel vytvořit vlastní metodu, která zajišťuje hloubkovou kopii, což je podmíněno absolutním přístupem k haldě, který nemám. Samozřejmě, mohl bych implementovat vlastní haldu, ale tímto odstavcem jsem spíše chtěl naznačit, že by to šlo "o chlup úsporněji", než se pouštět do vlastní implementace ☺.