Časté chyby

Úloha 1

- Ryby se často pohybovali po přímých čarách s dokonalým odrazem jako spořič DVD přehrávače. Uvítal bych spíše opravdu nahodilý pohyb, namísto nahodilého počátečního stavu, od kterého je vše deterministické.
- Mnoho z vás v Java/C#/Py použili pro předání stavu aplikace globální/statické proměnné. Ačkoliv to není vysloveně chyba, je velmi špatný zlozvyk "házet všechno do globálu". Do globálního scope patří pouze konstanty, konfigurace a pár dalších věcí jako třeba připojení do databáze. Jakýkoliv vnitřní stav logiky programu by se měl předávat místně přes parametry funkcí/metod za pomoci jednotlivých hodnot nebo struktur/tříd.
- Mnoho z vás pletli mezi sebou vykreslování a změnu stavu nebo čekání vláken. Animace je obecně dobré implementovat následovně:
    - Funkce, která vykreslí aktuální stav
    - Funkce, která změní aktuální stav
    - Timer, který tyto funkce volá (pokud takový konstrukt v použitém jazyce existuje)
- Mnoho z vás měli potíže s generováním náhodnosti. V Java/C# je třída `Random`, jejíž instance generují (zdánlivě) náhodná čísla. V rámci programu teoreticky vždy stačí jen jedna instance `Random`, kterou opakovaně voláte. Pokud při každém volání znovu vytvoříte novou instanci, tak už ta čísla nebudou tak náhodná. V C# je `Random` jednoduchý generátor, který (není-li řečeno jinak) odvozuje počáteční vnitřní stav z aktuálního času v sekundách. Takže pokud během jedné sekundy vygenerujete 100 čísel, každý s novou instancí `Random`, tak budou všechna stejná.

Úloha 2

Referenční řešení používá algoritmus Bellman-Ford. Používá tabulku sousednosti s počátečními hodnotami samotnými hranami. Pro každý počáteční vrchol následně N-krát zohlední každou hranu, pokud nezkracuje nějakou z cest. V grafu bez negativních smyček v poslední iteraci nic nezkrátí. Pokud ano, jedná se to zápornou smyčku. Zároveň při každém zkrácení si uloží informaci do tabulky relaxací - "tento vrchol byl zkrácen tímto sousedem". Na konci vybere jeden vrchol, který byl v poslední iteraci zkrácen a za pomoci relaxací zrekonstruuje zbytek smyčky.

Další možné řešení, které ale pouze spolehlivě určí, zda nějaká záporná smyčka existuje, ale už neurčí její vrcholy ani váhu je algoritmus Floyd-Warshall. Ten má také tabulku sousednosti a pro každý pár vrcholů A,B zohledňuje všechny ostatní vrcholy C, zda existují už nějaké cesty (A,C) a (C,B) a zda je jejich celková váha nižší než existující cesta (A,B). Na konci by tabulka sousednosti měla mít v diagonále (na souřadnicích (0,0), (1,1), ...) pouze nuly. Pokud ne, existuje záporná smyčka. Dá se výsledkem určit, které vrcholy se podílí na nějaké záporné smyčce, ale už se nedají určit konkrétní smyčky, ani správná hodnota této smyčky. V referenčním řešení je tento algoritmus dodatečně použit jako kontrola.

- Mnoho z vás jste implementovali buď rekurzivní procházení grafem nebo DFS. Rekurzivní procházení je velmi pomalé, ačkoliv korektní řešení. DFS nemusí nutně najít všechny smyčky, jelikož jakmile dojde do nějakého vrcholu, už do něj odjinud vcházet odmítá. Tím ale nikdy nezohlední všechny hrany (krom případů, kdy neexistují žádné smyčky v grafu).

Úloha 3

- Opět problémy se statickým/globálním stavem.
- Mnozí z vás nekontrolovali korektní vstup a to ať ze souboru karet, tak od uživatele.
- Mnoho z vás zapomnělo na uživatelem určené množství karet (vstup jich obsahoval 100, což je moc na jednu hru)
- Mnozí z vás implementovali rozhraní v CLI slovy "zadejte pozici, kam chcete kartu umístit". Pak je ale nutné vysvětlit odkud se čísluje, jestli tím je myšleno před nebo za kartu na zadané pozici atd. Ideálně toto buď vysvětlit v readme nebo po spuštění nebo vypsat mezi kartami očíslované mezery.

Úloha 4

Referenční řešení obsahuje Třídu, která obsahuje vnitřní stav pokladny a metodu `run`, která zpracuje jeden řádek. Vnitřní stav obsahuje paragon, předchozí váhu, předchozí sken a aktuální stav automatu. `run` na základě stavu automatu a prvního slova určí co má dělat, zkontroluje vstupy a zpracuje příkaz.

- Mnoho z vás nekontrolovala samotný příkaz `PLATBA 0`, který z povahy úlohy vůbec nedává smysl.
- Problémy s kontrolou korektního vstupu. Musíte jednak kontrolovat že první slovo je jedno z uvedených, ale taky, že zbytek řádky obsahuje právě tolik slov, kolik má, a že jsou číselné.

- Dle zadání "přírůstek navážené hmotnosti musí odpovídat hmotnosti položky z akce SKEN +-5%". Pokud kontrolujete pouze celkovou jednotkovou váhu (součet vah z akce SKEN), pak vám může přijít několik produktů, které jsou trochu lehčí než jejich jednotkové váhy a pak jeden, který je výraně těžší (o více než 5%) a vy byste tento vstup nesprávně přijali. Někteří jste kontrolovali váhu ještě jinak, že pokud vám přišel kus -5% a pak jeden kus +5%, tak jste vyvolali chybu. Referenční řešení si uloží poslední naměřenou váhu a poté porovnává jednotkovou váhu s `next - previous`.