Korespondenční Seminář z Programování

ZAČÁTEČNICKÁ KATEGORIE

36. ročník | **KSP-Z** | **Prosinec 2023**

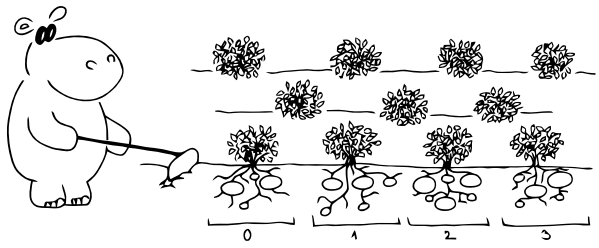
Řešení druhé série začátečnické kategorie 36. ročníku KSP

36-Z2-1 Kevin a krtiny

Při řešení využijeme faktu, že máme pozice krtin seřazené. Budeme si pamatovat, až do jaké pozice jsme odstranili veškeré krtiny v proměnné i. Na začátku se $i = 0$, protože jsme neodstranili krtinu žádnou. Stejně tak počítadlo použití $p = 0$.

Pozice budeme procházet od začátku. Když narazíme na krtinu, která je menší rovna i, nic neuděláme, protože jsme ji shrábli už dříve. V opačném případě hrábneme motyčkou tak, aby aktuální krtina byla úplně vlevo. To protože všechny krtiny nalevo už jsou vyřešené a tímto umístěním zahladíme největší množství krtin směrem doprava. Nastavíme tedy $i = k_j + M - 1$, kde k_j je pozice aktuální krtiny. Nyní je v i uloženo nejpravější políčko, které jsme zahrabali. Také si přičteme do počítadla p jedničku. Po projití všech krtin vypíšeme p.

Časová složitost je $\mathcal{O}(n)$, na každou krtinu se podíváme jen jednou. Pozice krtin si nemusíme ukládat, stačí nám je zpracovávat postupně rovnou ze vstupu. Paměťová složitost je tedy $\mathcal{O}(1)$.



Úlohu připravili: Petr Budai, Kačka Doubková, Janek Hlavatý

36-Z2-2 Oběd v továrně

Úlohu vyřešíme simulováním pohybu krabičky. Na začátku si pořídíme proměnnou s pozicí krabičky a inicializujeme ji na x, y ze vstupu.

V každém kroku algoritmu se podíváme, kterým směrem ukazuje šipka, kterou má krabice pod nohama. Tímto směrem krabici posuneme. Až se nám stane, že krabice vypadla z mapy, ohlásíme pozici těsně před vypadnutím.

Každý krok algoritmu trvá konstantní množství času. Ze zadání máme zároveň garantováno, že cesta krabice netvoří cyklus. Z toho plyne, že kroků algoritmu nemůže být více než políček na mapě.

Celková časová složitost algoritmu je tedy $O(RS)$. Paměti potřebujeme také $O(RS)$ na uložení pole ze vstupu.

Úlohu připravili: Prokop Randáček, Dan Skýpala

36-Z2-3 Počítačová hra

Naším cílem je obstarat co nejlevněji bronzovou sošku. To je přeci přímočaré – vyzkoušíme všechny recepty na bronzovou sošku, spočítáme ceny surovin a tu nejmenší porovnáme s cenou nákupu sošky rovnou za peníze. Jenže teď máme před sebou úplně stejný problém: určit ceny surovin. Tak když je úplně stejný, nešel by také řešit úplně stejně?

Zatím to vypadá beznadějně. Když budeme pořád převádět počítání ceny na počítání jiné ceny, můžeme se začít točit v kruhu a nikam se neposouvat. Naštěstí má zadání v sobě ukryté vysvobození. Recept vždy požaduje suroviny s větším číslem, než je číslo produktu. S tímto omezením žádný cyklus nemůže vzniknout. Nevěříte-li, nahlédněte do kuchařky o grafech,[1] části o *topologickém uspořádání*.

Důležité je, že předmět je vždy stejně drahé získat, ať už ho potřebujeme do jakéhokoli receptu. Již naceněné si tedy můžeme pamatovat. Formálně bude naším algoritmem rekurzivní funkce, nazvěme $f(i)$. Na vstupu vezme suroviny a na výstupu vydá cenu. Tu zjistí následujícím postupem

1. Pokud je k i již uložena cena, vrať ji.
2. Najdi nejlevnější recept, tedy pro každý recept R produkující i sečti ceny $f(j)$ všech předmětů j potřebných na recept R a vyber nejlevnější R
3. Vrať minimum z nejlevnějšího receptu a ceny přímého nákupu a ulož jej pro pozdější využití.

Pro rychlou implementaci bodu 2) je potřeba si recepty předem předroztřídit do nějaké rychlé implementace slovníku podle předmětů, které produkují.

Cena sošky je pak jen vyhodnocení $f(0)$.

Jak dlouho to trvá? Díky kešování v bodě 1) si můžeme všimnout, že cenu každého receptu počítáme jen jednou a to v čase úměrném počtu přísad. Přesně takto jsou zadané na vstupu, proto je i časová složitost tohoto algoritmu lineární s délkou vstupu.

Úlohu připravili: Vojta Káně, Vojta Lančarič, Jirka Sejkora, Ondra Sladký

[1] `http://ksp.mff.cuni.cz/viz/kucharky/grafy`

36-Z2-4 Stará trať

Můžeme začít tím, že budeme simulovat cestu vlaku a najdeme, kde dochází k problémům. Ty mohou nastat dvou typů.

První je, když projedeme zpomalovákem, rychlost klesne na nulu a vlak zastaví. Abychom tomu zabránili, musíme deaktivovat alespoň jeden zpomalovák před tímto místem (včetně toho, kterým jsme právě projeli). Pokud bychom deaktivovali cokoliv za tímto místem, rychlost tady by to neovlivnilo. Pokud bychom deaktivovali zrychlovák, rychlost by se jenom snížila.

Představme si tedy, že jsme jeden zpomalovák deaktivovali. Rychlost je teď jedna a pokračujeme v simulaci. Řekněme, že se toto stalo a-krát.

Když dojedeme na konec, může nastat druhý typ problému – pokud je rychlost větší než jedna, vlak vykolejí. Koncovou rychlost můžeme snížit jedině deaktivováním některých zrychlováků. Pokud je rychlost například $1+b$, musíme jich deaktivovat alespoň b, abychom ji snížili na jedna.

Po odsimulování se tedy dozvíme, že musíme deaktivovat minimálně a zpomalováků a b zrychlováků. Teď potřebujeme vybrat které.

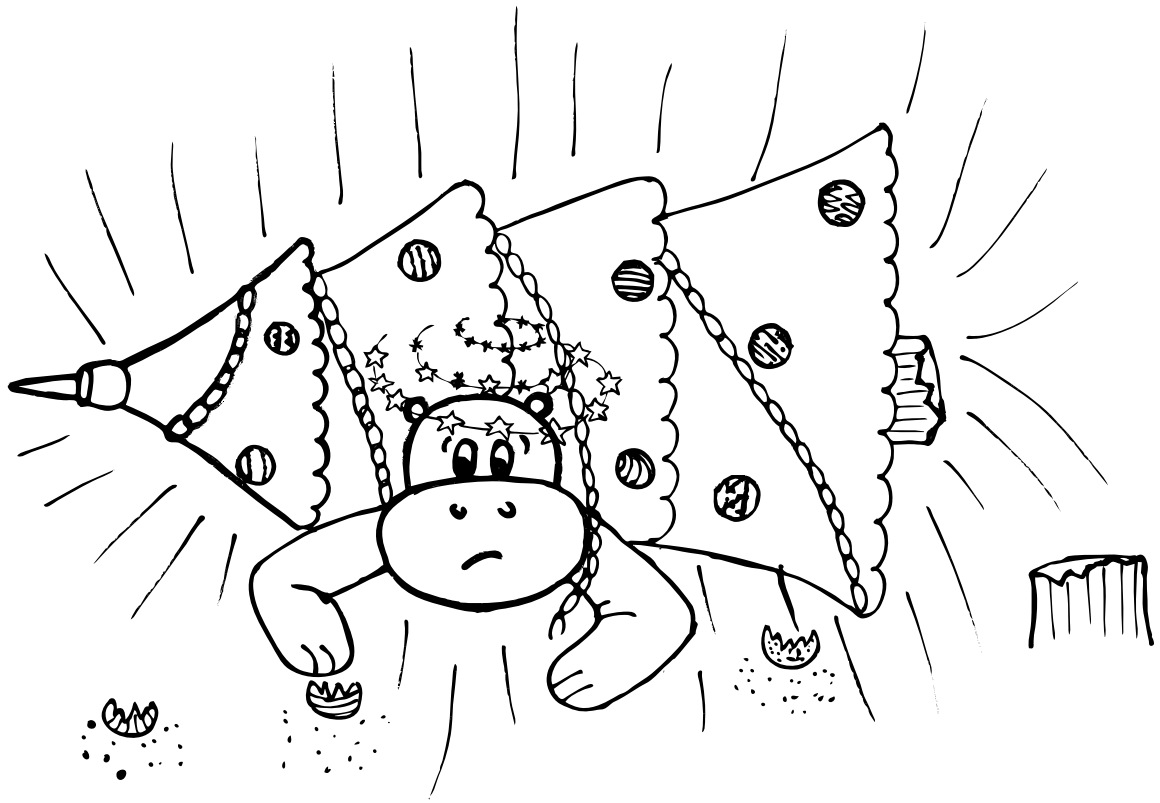
Pokaždé, když narazíme na problém vyžadující deaktivaci zpomalováku, víme, že to musí být ten, na kterém rychlost klesla na nulu, nebo nějaký dřívější. Jelikož volba neovlivní zbytek simulace, můžeme vybrat třeba ten, na kterém právě jsme.

U zrychlováků nechceme deaktivovat nějaký moc brzo, protože pak by mezi ním a koncem mohla rychlost klesnout na nulu. Začneme tedy od konce. Řekněme, že za posledním zrychlovákem je c zpomalováků. Jelikož na konci byla rychlost $1+b$, tak za posledním zrychlovákem musela být $1+b+c$. Když ho deaktivujeme, tak rychlost v daném místě klesne na $b+c$ a potom bude klesat až na $b>0$. Když toto zopakujeme b-krát, vlak dorazí do cíle s rychlostí jedna, aniž by zastavil uprostřed cesty.

S tímto výběrem deaktivací tedy vlak dorazí v pořádku do cíle. Jelikož víme, že s menším počtem deaktivací to nejde a větší není potřeba, máme optimální řešení.

Časová složitost je $\mathcal{O}(N)$ – vstup projdeme tam a zpátky a na každém prvku provádíme operaci v konstantním čase. Paměťová složitost je $\mathcal{O}(N)$ – musíme si pamatovat pozice zrychlováků, abychom je na konci mohli vypsat. Pokud nemusíme na začátku výstupu dávat jeho velikost a nezáleží na jeho pořadí, můžeme si pamatovat jenom ty zrychlováky, které mají šanci být součástí výstupu. V každém okamžiku je to tolik posledních zrychlováků, kolik je aktuální rychlost minus jedna. Paměť by pak odpovídala nejvyšší rychlosti, na kterou při simulaci narazíme.

Úlohu připravil: Jan Adámek



KSP pro vás připravují studenti Matematicko-fyzikální fakulty Univerzity Karlovy.
Realizace projektu byla podpořena Ministerstvem školství, mládeže a tělovýchovy.

Webové stránky: *https://ksp.mff.cuni.cz/*
E-mail: *ksp@mff.cuni.cz*
Organizátoři a kontakty: *https://ksp.mff.cuni.cz/kontakty/*