Korespondenční Seminář z Programování

ZAČÁTEČNICKÁ KATEGORIE

36. ročník | **KSP-Z** | **Duben 2024**

Řešení čtvrté série začátečnické kategorie 36. ročníku KSP

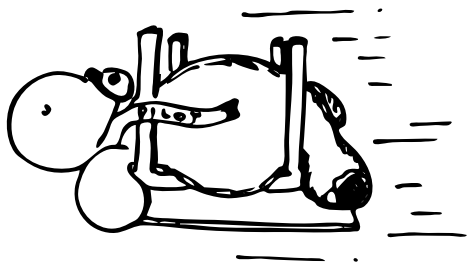
36-Z4-1 Sněhuláci

Nejprve si všimneme, že z libovolných tří navzájem různých koulí jde postavit sněhulák. Stačí je dát nad sebe v dobrém pořadí. V zadání máme slíbeno, že jsou koule různé, stačí tedy "rozřezat" posloupnost na vstupu do částí po třech a ty pak seřadit. Abychom ušetřili paměť, tak navíc můžeme sněhuláky vypisovat a třídit zároveň se čtením vstupu.

Rozdělení posloupnosti na trojice potrvá $\mathcal{O}(N)$. Mohlo by se zdát, že třídění potrvá více času, my však třídíme pouze posloupnosti o fixní délce 3 (na to ani nepotřebujeme žádný chytrý algoritmus, stačí to rozdělit na jeden ze 6 případů), každé třídění tedy potrvá konstantní čas. Algoritmus tedy celkově poběží v čase $\mathcal{O}(N)$. Program (Python 3):
`http://ksp.mff.cuni.cz/viz/36-Z4-1.py`

Úlohu připravili: Honza Černohorský, Adam Jahoda

36-Z4-2 Sáňkování



Na vstupu jsme získali pole výšek rovin a Sára se nachází na té první z nich. K vyřešení úlohy si stačí uvědomit, jaké situace při průjezdu na sáňkách mohou nastat – jedeme dolů, jedeme nahoru s dostatečnou rychlostí a jedeme nahoru, ale už nedojedeme. Celou úlohu tedy budeme simulovat a při výpočtu si budeme pamatovat rychlost, počet metrů, které musí Sára vyšlapat, a aktuální výšku.

Uděláme si cyklus, ve kterém pro každý prvek pole s výškami vypočítáme, která ze zmiňovaných situací nastala. Pokud jedeme dolů, přičteme k rychlosti rozdíl výšky aktuální a následující roviny. Jedeme-li směrem nahoru a máme dostatečnou rychlost, tedy výška aktuální roviny plus rychlost je aspoň tak velká jako výška následující roviny, tak odečteme od rychlosti rozdíl výšky vyšší (následující) a nižší (aktuální) roviny. Toto značí, že jsme použili rychlost na vyjetí kopce. V případě, že jedeme nahoru, ale nemáme dostatečnou rychlost na vyjetí kopce, tak se rychlost sníží na nulu a musíme stoupat rozdíl výšek dvou rovin pěšky. Na konci výpočtu budeme mít spočítaný počet metrů, co musí Sára vyšlapat, a tuto hodnotu vypíšeme.

Časová složitost je $\mathcal{O}(N)$, protože musíme projít postupně za sebou všechny hodnoty v poli, kterých je N. Program (Python 3):
`http://ksp.mff.cuni.cz/viz/36-Z4-2.py`

Úlohu připravili: Petr Budai, Honza Černohorský, Jirka Sejkora

36-Z4-3 Náledí

K řešení této úlohy použijeme prohledávání grafu. Začneme ze startovního políčka a postupně se zkusíme odrazit do všech 4 možných směrů. Políčko, na kterém po odrazu skončíme, označíme za navštívené a opět se zkusíme odrazit do všech 4 možných směrů (pakliže jsou daná cílová políčka nenavštívená).

Jak ale takovýto odraz vypadá? Respektive jak z naší pozice $[r, s]$, kde r je řádek a s je sloupec, zjistíme, na kterém políčku po odrazu do jednoho ze čtyř směrů skončíme?

Každý odraz si budeme v bludišti simulovat. Pro odraz `L` se budeme pohybovat po stejném řádku r doleva, čili budeme hodnotu s aktuálního sloupce postupně snižovat. Jakmile bychom se dostali na pozici v tabulce takovou, že se na levém přilehlém políčku nachází socha nebo okraj náměstí, nalezli jsme koncové políčko po odrazu doleva.

Analogicky budeme hledat políčka pro ostatní odrazy, jen pro odrazy `N` a `D` budeme měnit číslo řádku r a pro odrazy `R` a `D` budeme hodnotu zvyšovat, nikoliv snižovat.

Do cíle dorazíme tehdy, když někdy během našeho klouzání stoupneme na cílové políčko.

Nyní zbývá ještě vyřešit poslední problém – nalezenou cestu do cíle vypsat. K tomu nám stačí si v průběhu prohledávání ke každému políčku poznamenávat, ze kterého políčka jsme do něj přišli. Nakonec pak stačí tyto záznamy zpětně projít, zjistit, ze kterého směru jsme museli přijít, a vypsat směry v opačném pořadí.

V našem algoritmu se po sklouznutí zastavíme v $\mathcal{O}(WH)$ políčkách. Samotné nalezení koncových políček však trvá $\mathcal{O}(W + H)$ času (musíme projít pokaždé nejhůř jeden celý řádek a sloupec tabulky), výsledná časová složitost je tím pádem $\mathcal{O}(WH(W + H))$. Kromě mapy náměstí si musíme pamatovat, která políčka jsme již navštívili a odkud jsme do nich přišli, což nás tak vyjde na $\mathcal{O}(WH)$ paměti.

Všimněme si ještě, že po nás zadání nevyžaduje nejkratší cestu, stačí nám jakákoliv. Nezáleží tedy, jestli náměstí prohledáváme do hloubky či do šířky.

Program (Python 3):
`http://ksp.mff.cuni.cz/viz/36-Z4-3.py`

Úlohu připravili: Honza Černohorský, Kačka Doubková, Jirka Sejkora

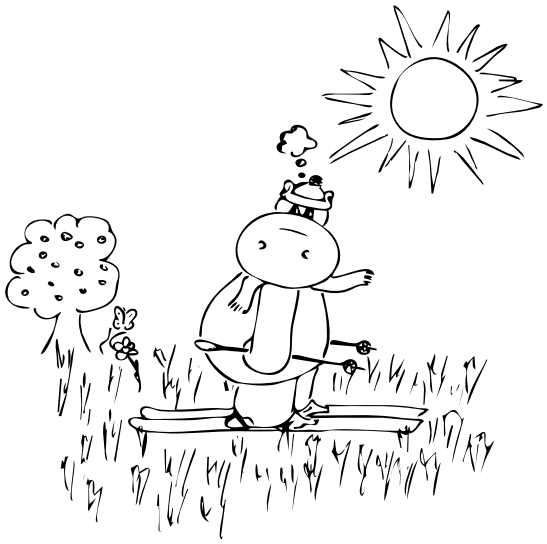
36-Z4-4 Chata

Celé Hroší hory tvoří strom. Pokud jste se s tímto grafovým termínem ještě nesetkali, můžete si přečíst víc v naší kuchařce.[1] O stromech platí jeden zajímavý fakt – mezi dvěma vrcholy vede vždy právě jedna cesta. A toho my využijeme. Náš algoritmus bude založen na prohledávání do hloubky (o něm si můžete více přečíst v té stejné kuchařce).

Křižovatku, na které se Zuzka zrovna nachází, označíme za kořen. Vybereme si libovolnou hranu směrem z kořene a vydáme se po ní. Vždy, když dorazíme k nějakému vrcholu, si budeme pamatovat, jak daleko od kořene jsme. Pokud jsme stejně daleko, jako je Zuzky chata, můžeme prohledávání ukončit. Zuzka už ví kudy domů.

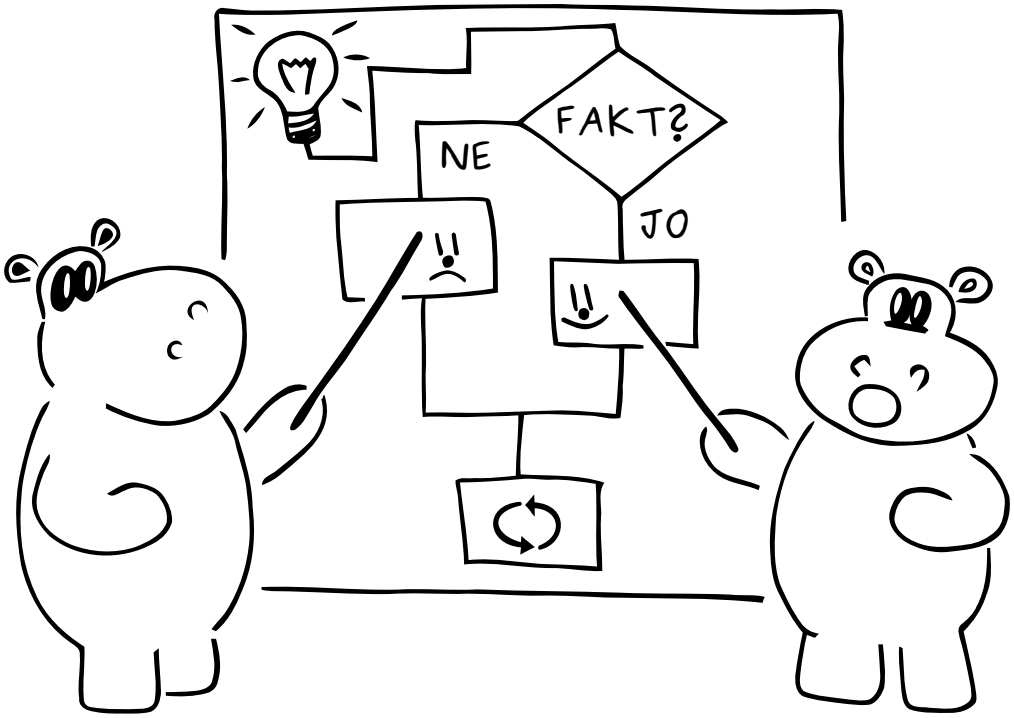
V nějaký moment zcela jistě dorazíme do vrcholu, z něhož už žádné neprozkoumané hrany kromě té, ze které jsme přišli, nevedou. Nemáme tedy jinou možnost, než se vydat zpátky. Tímto způsobem projdeme celý strom a u každého vrcholu spočteme, jak daleko je od kořene.

Pokud projdeme celý strom a na žádný vrchol ve vhodné vzdálenosti nenarazíme, můžeme si být jisti, že žádný takový vrchol neexistuje a Zuzce se zbláznily hodinky, neboť jsme prošli všechny vrcholy.



Jak rychle algoritmus běží? Každou hranu projdeme pouze dvakrát a hran je ve stromu o jedna méně než vrcholů, tedy $\mathcal{O}(N)$. Celý algoritmus tedy poběží v čase $\mathcal{O}(N)$.

Úlohu připravil: Adam Jahoda



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

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/*