Soutěž dětí a mládeže v programování

Obvodní kolo Prahy 1 a 2, rok 2023

Autoři: Adam Benda, Martin Horský, David Šertler; verze 1.0, 2023-02-22

Každá úloha má koeficient obtížnosti, kterým se násobí hodnocení vašeho řešení. Odevzdejte nám prosím řešení přes odevzdávátko (https://soutez.lnker.xyz/) jako komprimovanou složku zip. Odevzdat jednu úlohu můžete vícekrát - nejlépe po dosažení nějakého stupně funkčnosti. Hodnotíme verzi úlohy, kterou jste odevzdali jako poslední. Kromě splnění zadané funkčnosti se hodnotí přehlednost vašeho kódu (komentáře, popisující názvy proměnných) a efektivita vašeho řešení.

Při soutěži je zakázána komunikace (přítel na telefonu). Vyhledávač a Internetové zdroje používat můžete ALE: cílem je zjistit vaši schopnost algoritmizace. Zkopírovat si několik řádků kódu ze stackoverflow (uveďte zdroj) nebo použít knihovní funkci řešící dílčí část problému je v pořádku. Zkopírovat odněkud hotové řešení celé úlohy není přípustné. V případě pochybností konzultujte porotce.

Používání knihoven je povoleno, jelikož mnoho programovacích jazyků má relativně omezenou standardní knihovnu (např. grafické zobrazování v okýnkách). Nepoužívejte ale knihovny, které víceméně vyřeší úlohu za vás.

Současně nepoužívejte AI generátory stylu ChatGPT, GitHub Copilot a podobné. “Chytré našeptávače” nezakazujeme, ale ať vám to negeneruje víc jak pár řádek na myšlenku.

Každá úloha používá vstupní data pro definici instance problému. Tyto vstupy jsme připravili v několika formátech (hlavně text, JSON a XML). Pokud je to určeno v zadání úlohy, máte také na výběr, jestli budete vstup načítat ze souboru, standardního vstupu nebo v grafickém rozhraní. Tyto rozhodnutí jsou na vás, ale pokaždé nám to napište do přiloženého README.

Testovací vstupy k úlohám naleznete na https://soutez.lnker.xyz/res/.

Po souteži nám prosím vyplňte anketu na https://lnker.xyz/anketa.

Úloha 1: Find me a song (koef. 3)

Máte problém. Chcete poslouchat písničky v průběhu práce, víte kdy musíte skončit, ale nechcete skončit uprostřed písničky. Víte jaké písničky chcete poslouchat, takže víte jak jsou tyto písničky dlouhé. Mělo by tedy být možné vytvořit playlist, který obsahuje tyto písničky a jeho celková délka je přesně rovna určenému času, nebo jenom těsně kratší.

Na vstupu dostanete požadovanou délku playlistu a dostupné písničky a jejich délky. Váš program by měl z těchto dat určit, které z těchto písniček použije. Žádná z těchto písniček se

nesmí opakovat a celková délka vybraných písniček má být co možná nejbližší požadované délce, ale ne delší. Tedy rozdíl požadované délky a délky playlistu musí být nezáporný minimální.

Vstup máte k dispozici v několika formátech. Buďto textově nebo v JSON, YAML a XML.

Pro textový vstup: První řádka bude obsahovat požadovanou délku v sekundách (jedno číslo) a každá další řádka bude obsahovat délku písničky a její název oddělený mezerou (název písničky také může obsahovat mezery, ale důležitá je první mezera na řádce).

Konec vstupu může být buďto konec souboru nebo prázdný řádek a můžete jej načítat buďto ze standardního vstupu nebo ze souboru. To je na vás.

Vstupy v ostatních formátech obsahují stejné informace, ale strukturovaně. Konkrétní podoba je evidentní ze vzorových vstupů.

Výstup programu bude seznam vybraných písniček v playlistu. Na pořadí písniček nezáleží. Také je na vás, jestli budete vypisovat tyto písničky v nějakém grafickém okně, do standardního výstupu nebo do souboru. Pokud má výstup nějakou textovou podobu, tak ať je to buďto seznam písniček na samostatných řádcích nebo nějak strukturovaný JSON, YAML nebo XML.

Hodnocení:

- Program vygeneruje playlist s validní délkou - menší rovna požadované a složené z dostupných písniček - 10%
- Program vygeneruje playlist, který má prázdnou mezeru na konci kratší než nejkratší nepoužitá písnička - 20%
- Program vygeneruje optimální řešení, tedy prázdná mezera na konci není, nebo je nejmenší možná - 60%
- Přehlednost, efektivita řešení - 10%

Úloha 2: Čtverečkovaný papír - bludiště (koef. 2)

Představ si, že máš bludiště na čtverečkovaném papíře a potřebuješ v něm najít nejkratší cestu mezi dvěma body. V této úloze se zaměříme na tvorbu programu, který bude umět hledat nejkratší cestu v bludišti načteném ze souboru.

Bludiště bude definováno jako obdélníková mřížka, kde každý čtvereček bude buď průchozí (označený jako '.') nebo nepřístupný (zeď označená jako '#'). Startovní pozice bude označena jako 'S', cílová pozice bude označena jako 'C'.

Program by měl být schopen najít nejkratší cestu mezi startem a cílem. V bludišti se můžete pohybovat pouze nahoru, dolu, doprava a doleva. Po diagonálách se nepohybujete. Nejkratší cestu označte znakem 'X'. Bludiště načtěte ze souboru, jehož název bude zadaný jako první argument programu. Vstupní soubor reprezentující bludiště bude mít jeden z následujících

formátů: TXT, JSON, XML. Pro řešení úlohy stačí načítat vstup pouze v jednom z výše uvedených formátů.

Program by měl vytisknout bludiště s nalezenou nejkratší cestou mezi startem a cílem. Pokud bude nemožné najít cestu mezi zadanými body, program by měl tuto skutečnost vypsat na standardní výstup.

Hodnocení:

- Program načte bludiště ze souboru a je schopen jej vypsat - 20%
- Program určí nejkratší řešení bludiště 10x10 a zda lze vůbec řešit - 30%
- Program určí nejkratší řešení libovolné velkého bludiště - 40%
- Přehlednost, efektivita řešení - 10%

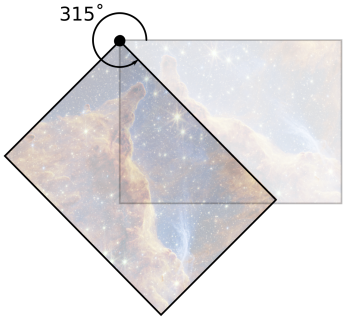
Úloha 3: Teleskop Jamese Webba (koef. 3)

Teleskop Jamese Webba byl uveden do provozu v roce 2022 a přináší fascinující fotografie nejvzdálenějších vesmírných objektů. V této úloze si vyzkoušíme, jak takové fotografie vznikají.

Díky tomu, že astronomické objekty se hýbou velice pomalu, používá se vícenásobná expozice. Pořídí se při ní mnoho fotografií s drobnými posuny. Tyto fotky se potom jako dlaždice skládají do úchvatných velkých obrazů.

Souřadnicový systém: 0,0 je levý horní bod cílového obrázku, osa x jde doprava, osa y dolů.

Otočení obrázku: úhel (ve stupních, celé číslo mezi 0 a 359). Dlaždice se otáčí ve svém levém horním rohu v kladném směru otáčení (proti směru hodinových ručiček). Dlaždici s úhlem 0 není třeba otáčet.



Vstup:

Složka s dlaždicemi - obrázkovými soubory (png) nezvanými

[nazevCelehoObrazku]_[sirkaCelehoObrazku]_[vyskaCelehoObrazku]_[x]_[y]_[sirkaKousku]_[vyskaKousku]_[uhelOtoceniStupne].png

a TXT/JSON/CSV/XML soubory se seznamem oněch obrázků. Ty strukturovaně opakují informace které naleznete v názvu, nepřinášejí žádné informace navíc. Úlohu je tedy možné plně zvládnout i bez načítání těchto doplňkových souborů.

x a *y* jsou celá čísla, určující souřadnice levého horního rohu dlaždice, v souřadnicovém systému celého obrázku.

Výstup:

- Sestavený obrázek uložte do souboru (png), nebo přímo zobrazte uživateli
- Vypište, zda bylo možné sestavit obrázek celý, a nebo nějaká část zbývá nepokrytá dlaždicemi. Vypište o jak velkou část obrázku se jedná (např. “21.90% obrázku nepokryto”)

Oblast kterou pokrývá více než jedna dlaždice může být pokryta libovolnou z nich - deformace způsobené případným otáčením nám pro zjednodušení nevadí.

Testovací sady:

- sada0 - elementární příklad užitečný pro ruční ladění
- sada1 - obrázky, kde není třeba otáčet (úhel otočení 0˚)
- sada2 - obrázky, pro které je třeba implementovat otáčení

Hodnocení:

- Program správně sestaví obrázky z dlaždic bez natočení - 30%
- Detekuje nepokryté místo v obrázcích z dlaždic bez natočení - 10%
- Program správně sestaví obrázky z dlaždic s natočením - 40%
- Detekuje nepokryté místo v obrázcích z dlaždic s natočením - 10%
- Přehlednost, efektivita řešení - 10%

Úloha 4: DTP - zalamování řádek (koef. 1)

Autoři lifestylového časopisu zásadně nepoužívají při svojí práci klávesu Enter. Panu sazeči to vždy přidělává tuze moc práce - pokud tedy nechce jejich plátek tisknout na toaletní papír. Pomozte mu a naprogramujte utilitku na zalamování řádků.

Vstupem programu je delší text, neobsahující znak pro nového řádku a číslo D určující maximální počet znaků na řádku. Program musí text na vhodném místě zalomit, a to ve dvou různých módech:

1. Zalomení nastane přesně po D znacích (tedy klidně uvnitř slova).
2. Zalomení nastane po posledním slově, které se ještě vejde do limitu D znaků na řádek. Mezeru následující po tomto slově nahraďte odřádkováním - následující řádek tedy

nezačíná mezerou. Pro jednoduchost jsou slova posloupnosti libovolných znaků neobsahující mezeru - interpunkční znaménka jsou součástí slov.

Řešení musí text měnit - přidávat odřádkování. Nestačí tedy text zobrazit v nějakém UI prvku, který by za vás zalomení vyřešil sám.
Jako odřádkování můžete použít “\n” (znak s byte hodnotou 0x0a) pokud máte rádi linux, “\r\n” (znaky s bajtovou hodnotou 0x0d 0x0a) pokud máte rádi windows, a “
”, pokud máte nejraději html.

Testovací vstupy naleznete v .txt souborech - každý soubor obsahuje jednu řádku. Soubory jsou kódovány v utf-8. Číslo D a mód v souborech nenaleznete, načtěte je od uživatele programu při běhu nebo jako parametr spuštění.

Hodnocení:

- Správná implementace módu 1 - 30%
- Správná implementace módu 2 - 60%
- Přehlednost, efektivita řešení - 10%