Soutěž v programování 2023
Obvodní kolo Prahy 2
Zhodnocení úloh

Vážení účastníci,

od 47 řešitelů se nám sešlo 104 řešení úloh. 10 řešení úloh bylo ohodnoceno maximálním počtem bodů. Nejlepší řešitel obdržel 56,7 bodů z 90 možných.

Zadání tedy - záměrně - bylo velmi obtížné při daném časovém limitu. Můžete být hrdí na každý získaný bod - a na každou zvládnutou a odladěnou úlohu ještě víc.

Aby vám bylo přemýšlení nad úlohami co nejužitečnější - ať už v dalších soutěžích, studiu, nebo při budoucí profesionální dráze vývojářů - přinášíme krátké zasazení úloh do kontextu, a rozbor nejčastějších chyb.

Děkujeme za účast v soutěži, a doufáme, že jste si řešení úloh užili.

Adam Benda, Martin Horský, David Šertler - porotci

Zadání úloh, testovací vstupy i referenční řešení naleznete na webu programovacisoutez.cz

Úloha 1: Find me a song

O úloze

Cílem je dojít k optimálnímu řešení takzvaného Problému batohu - ale dost bodů dáváme i za nalezení řešení o něco horšího. Potíží totiž je, že při záludně zvolených vstupních datech může být optimální kombinace písniček jedna jediná z celé plejády možností, kterých je 2^N (kde N je celkový počet písní k výběru).

Máme-li tedy k dispozici 100 písní s různými délkami, možná hledáme jednu z 1 267 650 600 000 000 000 000 000 000 000 možností. To vám žádný počítač nespočítá. Říká se tomu exponenciální složitost.

Rozumným kompromisem je postupné přidávání písniček do playlistu, dokud máme k dispozici nějakou, která se vejde. Pokud postupujeme od nejdelších písní směrem ke kratším, jedná se o takzvaný hladový (greedy) algoritmus.Pokud jsou délky celočíselné a požadovaná doba je rozumně nízká (třeba do milionu sekund), je toto řešení možné.

Existuje ale ještě jiné řešení, které vrátí skutečně optimální playlist. Jedná se o použití dynamického programování, při kterém si vytváříme postupně dosažitelné součty “zdola”. Algoritmus, popsaný například zde, připomíná Eratosthenovo síto (pro prvočíselnost), nebo efektivní výpočet faktoriálu.

Nejčastější chyby

- Příliš dlouhé řešení - při formálně správném vygenerování všech kombinací (elegantní například v Pythonu `itertools.combinations`) pro velké testovací vstupy program prostě nikdy nedoběhne - přestože výsledky pro menší počty písniček (cca <20) jsou formálně v pořádku.
- Různé potíže vstupu a výstupu - většinou jsme v hodnocení příliš nezohledňovali
- Zmatení se v různých heuristikách - snaha o zlepšení hladového algoritmu (líbilo se nám třeba vygenerování všech dvojic písní, které zadám do playlistu na začátku a pak běh hladového algoritmu a výběr nejlepšího výsledku) sice mohla na testovacích datech dát o trochu lepší řešení, ale občas bohužel přidaná komplexita vedla k bugům.

Naše chyby, za které se moc omlouváme

- Vstupy poskytnuté při soutěži obsahovaly duplicitně nazvané písně s rozdílnými délkami
- Vstupy poskytnuté při soutěži byly perfektně řešitelné hladovým algoritmem, scházely tedy ty *zapeklité*.

Úloha 2: Čtverečkovaný papír - bludiště

O úloze

Tato úloha je implementací algoritmu pro nalezení nejkratší cesty v bludišti pomocí BFS (breadth-first search). Úloha obsahuje několik kroků:
- Načtení bludiště ze souboru.
- Nalezení startovního a cílového bodu v bludišti.
- Použití BFS pro nalezení nejkratší cesty v bludišti.
- Označení nejkratší cesty v bludišti.
- Výpis bludiště s označenou nejkratší cestou.

BFS prohledává graf postupně od startovního bodu a hledá cestu k cílovému bodu. V každé iteraci se vybírají sousedi aktuálního uzlu a pokud některý z nich není navštívený, přidá se do fronty a označí se jako navštívený. Pro každý uzel se ukládá informace o jeho předchůdci, což umožní vytvoření nejkratší cesty.

Tato úloha vyžaduje nalezení nejkratší cesty v bludišti, což je klasický příklad problému grafu, kde vrcholy představují body v bludišti a hrany představují přechody mezi těmito body. Nejkratší cesta v grafu může být nalezena pomocí různých algoritmů, nejenom pomocí BFS. Níže jsou uvedeny některé další algoritmy, které mohou být použity k řešení této úlohy:
- Dijkstrův algoritmus

- A* algoritmus
- Bellman-Fordův algoritmus
- Floyd-Warshallův algoritmus

Nejčastější chyby

- Nesprávné načtení bludiště ze souboru
- Nesprávné nalezení startovního a cílového bodu v bludišti
- Chybný algoritmus pro hledání nejkratší cesty v bludišti (v tomto případě je použit algoritmus BFS)
- Chybné označení nejkratší cesty v bludišti
- Nesprávný výstup, například chybějící hláška, když neexistuje cesta mezi startovním a cílovým bodem, nebo nevypsání bludiště s označenou nejkratší cestou - přidaná komplexita vedla k bugům.

Úloha 3: Teleskop Jamese Webba

O úloze

Těžiště úlohy tkví v soustavě souřadné a práci s vaší oblíbenou knihovnou pro funkci s obrázky. První část (bez otáčení) šlo zvládnout opravdu na pár řádků.

Správné umístění natočených dlaždic již vyžadovalo poměrně složitou transformaci - po otočení dlaždice se totiž posune souřadnice jejího horního levého rohu z (0,0) někam jinam - a to různě dle kvadrantu otáčení (0˚-90° = I. kvadrant, 90°-180° = II. kvadrant, 180°-270° = III. kvadrant, 270°-360° = IV. kvadrant). Užitečné je si situaci rozkreslit na papír, dříve než začneme s programováním.

Pokročilejší postup pak spočívá ve správném použití transformačních matic. Elementární grafické transformace se totiž dají skládat, čehož se používá při programování 2d i 3d her.

Pokrytí dlaždicemi je nejsnazší počítat po sestavení obrázku - prostě spočtením vyplněných pixelů. Pokud obrázek tvoříte s průhledností (RGBA), prostě počítáte poměr průhledných bodů vůči všem (šířka*výška). Zapeklité může být, pokud obrázky kladete na podkladovou barvu - například na bílou. Jak poznáte bílou barvu pozadí od bílé barvy na dlaždici? Lze použít techniku, kdy pro tuto část úlohy vytváříte další obrázek - masku - s podkladovou černou barvou a nekladete dlaždice, ale pouze jejich bílé obrysy. Pak spočítáte bílé body. Pokud používáte rotaci, nezapomeňte, že vlivem antialiasingu vám některé body vyjdou částečně průhledné (nebo “částečně bílé” na masce) a je třeba se rozhodnout, jak je chcete do výsledného poměru započítat. Tento jev je ale mimo rozsah úlohy, a tak hodnocení nijak neovlivnil.

Úloha 4: DTP - zalamování řádek

Nejčastější chyby

- Spolknutá slova
- Nedodržení délky řádky v módu 2 - například nezapočítání mezery do délky, nebo špatně zvládnutý cyklus na slova
- Nekonečný cyklus - zvlášť při výskytu delšího slova, než je délka řádky
- Problémy s utf-8 - česká písmenka s háčky a čárkami jsou totiž vícebajtová. Většina jazyků a rozhraní dnes naštěstí s utf-8 pracuje bez problémů, ale například v C++ je správné použití dost složité.
- Spolknutá poslední řádka - pokud v rámci cyklu dojde k `break` před výpisem řádky, je třeba řádku ještě dodatečně vypsat