

Standardní zadání semestrální práce pro KIV/PT 2022/2023

Zadání je určeno pro **dva** studenty. Práce zahrnuje dvě dílčí části - vytvoření funkčního programu diskrétní simulace a napsání strukturované dokumentace.

Zadání

Zásobovací společnost *Necháme to bloudovi s. r. o.* se specializuje na přepravu zboží do saharských oáz. Její majitel Harpagon Dromedár je však vyhlášená držgrešle, a tak nejraději využívá jako přepravní prostředky velbloudy, kteří nejsou nároční na údržbu a provoz. Přepravované zboží je uskladněno ve speciálních koších, které jsou přichycovány na velbloudí hrby. Pro svoji živnost Harpagon využívá jak velbloudy jednohrbé, známé též jako dromedáry, tak velbloudy dvouhrbé, kteří jsou někdy označováni jako drabaři. V poslední době se však starému Harpagonovi moc nedaří a spousta zvířat mu v poušti, částečně i vlivem změny klimatu, uhynula, což je pro něj citelná finanční rána, která mu dělá vrásky na čele. Rozhodl se tedy, že je čas dát prostor moderním technologiím, a proto si chce nechat vytvořit software, který mu pomůže rozplánovat přepravu všeho poptávaného zboží do oáz tak, aby nepřišel o nějaké další zvíře, dodržel závazky a neztratil klientelu, maximálně využil nosnosti zvířat a zároveň zvířata zbytečně neunavil delší cestou, než kterou opravdu musí jít. Tyhle požadavky můžeme jednoduše označit jako snahu o minimalizaci „cen“ přepravy. Vytvořme pro Harpagona simulační program, který mu pomůže naplánovat přepravu, známe-li:

- počet skladů S ,
- každý sklad bude definován pomocí:
 - kartézských souřadnic skladu x_s a y_s ,
 - počtu košů k_s , které jsou do skladu vždy po uplynutí doby t_s doplněny. Na začátku simulace předpokládejte, že došlo k doplnění skladů, tj. ve skladu je k_s košů,
 - doby t_n , která udává, jak dlouho trvá daný typ koše na velblouda naložit/vyložit (každý sklad může používat jiný typ koše, se kterým může být různě obtížná manipulace),
- počet oáz O ,
- kartézské souřadnice každé oázy x_o a y_o ,
- počet přímých cest v mapě C ,
- seznam cest, přičemž každá cesta je definována indexy i a j , označujících místa (oáza, sklad) v mapě, mezi kterými existuje přímé propojení, a platí: $i \in \{1, \dots, S, S+1, \dots, S+O\}, j \in \{1, \dots, S, S+1, \dots, S+O\}$, tj. sklady jsou na indexech od 1 do S a oázy na indexech od $S+1$ do $S+O$. Pozn. pokud existuje propojení z místa i do místa j , pak existuje i propojení z místa j do místa i ,
- počet druhů velbloudů D ,
- informace o každém druhu velblouda, kterými jsou:
 - slovní označení druhu velblouda, které bude uvedeno jako jeden řetězec neobsahující bílé znaky,
 - minimální v_{min} a maximální v_{max} rychlost, kterou se může daný druh velblouda pohybovat, přičemž jedinec se pohybuje konstantní rychlostí (obecně reálné číslo), která mu bude vygenerována v daném rozmezí pomocí rovnoměrného rozdělení,
 - minimální d_{min} a maximální d_{max} vzdálenost, kterou může daný druh velblouda překonat na jedno napojení, přičemž pro jedince je tato doba opět konstantní a jedná se o reálné číslo vygenerované v daném rozmezí pomocí normálního rozdělení se střední hodnotou $\mu = (d_{min} + d_{max})/2$ a směrodatnou odchylkou $\sigma = (d_{max} - d_{min})/4$, (pozn. velbloudi se mohou napít pouze v oázách a nebo ve skladech, jinde není voda k dispozici),
 - doba t_d , udávající kolik času daný druh velblouda potřebuje k tomu, aby se napil,
 - počet košů k_d udávající maximální zatížení daného druhu velblouda,
 - hodnota p_d udávající poměrné zastoupení daného druhu velblouda ve stádě, hodnota je zadána jako desetinné číslo z intervalu $< 0, 1 >$, přičemž platí $\sum_{d=1}^D p_d = 1.0$,

- počet požadavků k obslužení P ,
- každý požadavek bude popsán pomocí:
 - času příchodu požadavku t_z (pozn. požadavek přichází doopravdy až v čase t_z , tzn. nemůže se stát, že by jeho obsluha začala dříve, a že by v době příchodu požadavku byl náklad již na cestě),
 - indexu oázy $o_p \in \{1, \dots, O\}$, do které má být požadavek doručen,
 - množství košů k_p , které oáza požaduje,
 - doby t_p udávající, za jak dlouho po příchodu požadavku musí být koše doručeny (tj. nejpozději v čase $t_z + t_p$ musí být koše vyloženy v oáze).

Za úspěšně ukončenou simulaci se považuje moment, kdy jsou všechny požadavky obsloužené a všichni velbloudi se vrátili do svých domovských skladů. V případě, že se některý požadavek nepodaří (z jakéhokoli důvodu) obsloužit včas, pak simulace skončila neúspěchem, o čemž bude program informovat příslušným výpisem (viz níže).

V rámci výpisů použijte jednu z následujících variant (formát je závazný, indexace od jedné):

- Příchod požadavku:
Cas: <t>, Pozadavek: <p>, Oaza: <o>, Pocet kosu: <k>, Deadline: <t+t_p>
- Ve skladu se začíná připravovat velbloud na cestu:
Cas: <t>, Velbloud: <v>, Sklad: <s>, Nalozeno kosu: <k>, Odchod v: <t+k · t_n>
- Velbloud došel do oázy, kde bude něco vykládat (pozn. výpis nikde v průběhu nebude odřádkován, časovou rezervou t_r je myšlen rozdíl mezi časem, kdy má být náklad nejpozději vyložen a časem, kdy k vyložení opravdu došlo):
Cas: <t>, Velbloud: <v>, Oaza: <o>, Vyloženo kosu: <k>, Vyloženo v: <t+k · t_n>, Casova rezerva: <t_r>
- Velbloud došel do oázy/skladu, kde se musí napít před další cestou:
Cas: <t>, Velbloud: <v>, Oaza: <o>, Ziznivý <druh>, Pokracovani mozne v: <t+t_d>
nebo:
Cas: <t>, Velbloud: <v>, Sklad: <s>, Ziznivý <druh>, Pokracovani mozne v: <t+t_d>
- Velbloud prochází oázou, ale nemá zde žádný zvláštní úkol:
Cas: <t>, Velbloud: <v>, Oaza: <o>, Kuk na velblouda
- Velbloud dokončil cestu a vrátil se do skladu:
Cas: <t>, Velbloud: <v>, Navrat do skladu: <s>
- Požadavek se nepodařilo (z jakéhokoli důvodu) obsloužit včas:
Cas: <t>, Oaza: <o>, Vsichni vymreli, Harpagon zkrachoval, Konec simulace

Výstup Vašeho programu bude do standardního výstupu a bude vypadat například následovně:

```
...
Cas: 12, Pozadavek: 2, Oaza: 2, Pocet kosu: 3, Deadline: 30
Cas: 12, Velbloud: 5, Sklad: 3, Nalozeno kosu: 3, Odchod v: 18
Cas: 21, Velbloud: 5, Oaza: 1, Ziznivý dromedar, Pokracovani mozne v: 22
Cas: 23, Velbloud: 5, Oaza: 10, Kuk na velblouda
Cas: 24, Velbloud: 5, Oaza: 2, Vyloženo kosu: 3, Vyloženo v: 30, Casova rezerva: 0
...
```

Čas ve výpisu bude zaokrouhlen dle pravidel zaokrouhlení na celé číslo.

Minimální požadavky:

- Funkcionalita uvedená v zadání výše je **nutnou podmínkou pro finální odevzdání práce**, tj. simulační program při finálním odevzdání musí splňovat veškeré požadavky/funkcionalitu výše popsanou, **jinak bude práce ohodnocena 0 body**.
- V případě, že bude program poskytovat výše popsanou přepravu, ale **řešení bude silně neefektivní**, bude uplatněna bodová **penalizace až 20 bodů**.
- **Finální odevzdání práce** v podobě **.ZIP souboru** (obsahujícím zdrojové kódy + přeložené soubory + dokumentace) bude nahráno **na portál**, a to **dva celé dny před stanoveným termínem** předvedení práce, tj.:
 - studenti, kteří mají **termín předvedení** stanoven na **pondělí**, nahrají finální verzi práce na portál **nejpozději v pátek ve 23:59**,
 - studenti, kteří mají **termín předvedení** stanoven na **čtvrtek**, nahrají finální verzi práce na portál **nejpozději v pondělí ve 23:59**.

Při nedodržení tohoto termínu bude uplatňována **bodová penalizace 30 bodů**, navíc studentům nemusí být umožněno předvedení práce ve smluveném termínu.

Vytvoření funkčního programu:

- Seznamte se se strukturou vstupních dat (polohou skladů a oáz, informacemi o cestách, velbloudech, požadavcích ...) a načtěte je do svého programu. Formát souborů je popsán přímo v záhlaví vstupního souboru `tutorial.txt` (**5 bodů**).
- Navrhněte a implementujte vhodné datové struktury pro reprezentaci vstupních dat, důsledně zvažujte časovou a paměťovou náročnost algoritmů pracujících s danými strukturami (**10 bodů**).
- Proveďte základní simulaci jedné obslužné trasy včetně návratu velblouda do skladu. Vypište celkový počet doručených košů > 0 a celkový počet obsloužených požadavků > 0 . Trasa velblouda musí být smysluplná. (**10 bodů**).

Výše popsaná část bude váš minimální výstup při kontrolním cvičení cca v polovině semestru.

- Vytvořte prostředí pro snadnou obsluhu programu (menu, ošetření vstupů včetně kontroly vstupních dat) - nemusí být grafické, během simulace umožněte manuální zadání nového požadavku na zásobování některé oázy či odstranění některého existujícího (**5 bodů**).
- Umožněte sledování (za běhu simulace) aktuálního stavu přepravy. Program bude možné pozastavit, vypsat stav přepravy, krokovat vpřed a nechat doběhnout do konce, podobně jako je tomu v debuggeru (**5 bodů**).
- Proveďte celkovou simulaci a vygenerujte do souborů následující statistiky (v průběhu simulace ukládejte data do vhodných datových struktur, po jejím skončení je uložte ve vhodném formátu do vhodně zvolených souborů) (**10 bodů**):
 - přehled jednotlivých velbloudů - základní údaje o velbloudovi (druh; id domovského skladu; rychlost; max. vzdálenost, kterou urazí na jedno napojení), uskutečněné trasy (čas, kdy opustil sklad; kudy šel; kolik toho vezl; kam a kdy doručoval zboží; kde a kdy se zastavil na napojení; kdy se vrátil do svého domovského skladu), jak dlouho za celou dobu simulace odpočíval (tj. byl ve skladu a čekal na přiřazení požadavku) a celkovou vzdálenost, kterou ušel,
 - přehled jednotlivých oáz - čas a velikost vzniklého požadavku; kdy musel být nejpozději doručen; kdy byl skutečně doručen; ze kterého skladu a kterým velbloudem byl obsloužen,
 - přehled jednotlivých skladů - časy, kdy došlo k doplnění skladu; kolik košů v té době ve skladu zbývalo a kolik jich je k dispozici po doplnění,
 - délka trvání celé simulace, celková doba odpočinku všech použitých velbloudů, celková ušlá vzdálenost, kolik velbloudů od jednotlivých druhů bylo použito. Nemá-li úloha řešení, vypište, kdy a kde došlo k problému.

- Vytvořte generátor vlastních dat. Generátor bude generovat vstupní data pomocí rovnoměrného rozdělení, přičemž volte vhodně rozsah hodnot pro jednotlivé veličiny. U seznamu cest se vyhněte duplikátům. Data budou generována do souboru (nebudou přímo použita programem) o stejném formátu jako již dodané vstupní soubory. Při odevzdání přiložte jeden dataset s řešitelnou úlohou a jeden dataset, kdy nebude možné obsloužit všechny požadavky včas. **(5 bodů)**.
- Vytvořte dokumentační komentáře ve zdrojovém textu programu a vygenerujte programovou dokumentaci (Javadoc) **(10 bodů)**.
- Vytvořte kvalitní dále rozšiřitelný kód - pro kontrolu použijte softwarový nástroj PMD (více na <http://www.kiv.zcu.cz/~herout/pruzkumy/pmd/pmd.html>), soubor s pravidly `pmdrules.xml` najdete na portálu v podmenu Samostatná práce **(10 bodů)**
 - mínus 1 bod za vážnější chybu, při 6 a více chybách nutno opravit,
 - mínus 2 body za 10 a více drobných chyb.
- **V rámci strukturované dokumentace (celkově 20 bodů):**
 - připojte zadání **(1 bod)**,
 - popište analýzu problému **(5 bodů)**,
 - popište návrh programu (např. jednoduchý UML diagram) **(5 bodů)**,
 - vytvořte uživatelskou dokumentaci **(5 bodů)**,
 - zhodnoťte celou práci a vytvořte závěr **(2 body)**,
 - uveďte přínos jednotlivých členů týmu (včetně detailnějšího rozboru, za které části byli jednotliví členové zodpovědní) k výslednému produktu **(2 body)**.