# Semestrální práce

**(standartní zadání)**

# z předmětu KIV/PT 2022/2023

Vypracovali:

Lukáš Sobotka - A21B0276P - sobo@students.zcu.cz

Filip Černý -

**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 velboudy 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 jednodušše označit jako snahu o minimalizaci „ceny“ přepravy. Vytvořme pro Harpagona simulační program, který mu pomůže naplánovat přepravu, známe-li:

* počet skladů S
* specifikace jednotlivých skladů
* počet oáz 𝑂 a souřadnice každé oázy
* počet přímých cest v mapě 𝐶
* počet druhů velbloudů 𝐷,
* informace o každém druhu velblouda
* počet požadavků k obsloužení
* popis jednotlivých požadavků

**Základní Požadavky**

* 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).**

**Rozšiřující požadavky**

* 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 statistiky (v průběhu simulace ukládejte data do vhodných datových struktur, po jejím skončení je uložte ve vhodném formátu do vhodně zvolených souborů) **(10 bodů):**
* 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)**
* **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)**

**Analýza problému**

Už z prvního pohledu na zadání je zřejmé, že se jedná o hledání cest a o grafovou problematiku a. Při hledání cest je hlavní otázkou výběr algoritmu, hned jako první se nabízí Dijkstrův algoritmus, jelikož hrany našeho grafu jsou cesty s reálnou délkou, což znamená, že nemohou mít záporné ohodnocení, a tudíž nic nebrání jeho použití. Další algoritmus přicházející v úvahu je Floyd-Warshallův, který může být v jistých ohledech i lepším než právě ten Dijkstrův. Nejefektivnější implementace by zahrnovala oba tyto algoritmy, jelikož v tomto případě již předem víme počet vrcholů a cest, můžeme pro jiné vstupy programu zvolit vhodnější algoritmus podle hustoty grafu. Nakonec ale z důvodu snadnější implementace vybíráme samotný Dijkstrův, který budeme spouštět vždy v oáze, která si vyžádá zásilku a bude hledat nejpříhodnější sklad na vyslání velblouda.

Ovšem samotné vybrání cesty nestačí, jelikož v zadání úlohy se skrývají dosti záludné detaily. Nejprve bychom totiž měli zjistit, zda se do oázy vůbec dokážeme dostat, pokud bychom měli ideálního velblouda a poté jsme nuceni kontrolovat všechny již vytvořené velbloudy, zda by se některému z nich nepodařilo požadavek doručit. Teprve poté můžeme generovat vlastní velbloudy v tom dříve zmíněném příhodném skladě. Samotné velbloudy ještě musíme kontrolovat hned z několika hledisek. Nejprve zda dokáže překonat nejdelší hrany na vybrané cestě, pak zda dokáže unést požadované množství košů, a ještě zjistit požadavek vůbec stihne doručit během daného času.

**Návrh programu**

**Uživatelská dokumentace**

**Závěr**

Semestrální práce to byla velmi obtížná a naučná, i když na první pohled se zdála být jednoduchá. Každá sebemenší prkotina se ukázala ve finální práci být několikanásobně složitější, než jsme mohli čekat. Grafová část problematiky nebyla nijak složitá, ale podmínky, které musely být splněny pro obsloužení každého požadavku, ji značně zesložitily. Opravdu se zdálo, že všechno v zadání má svůj háček a máloco mohlo být splněno takříkajíc zadarmo.

Přínos jednotlivých členů

Lukáš Sobotka - Konkrétní části, kterými jsem nejvíce přispěl, budou nejspíše generátor vstupních dat, parser vstupu, zápis statistik do souboru a časová linka reprezentovaná prioritní frontu eventů. Jelikož parser vstupu byla úplně první potřebná část, tak jsem už od začátku měl velký vliv na kostru celé aplikace.