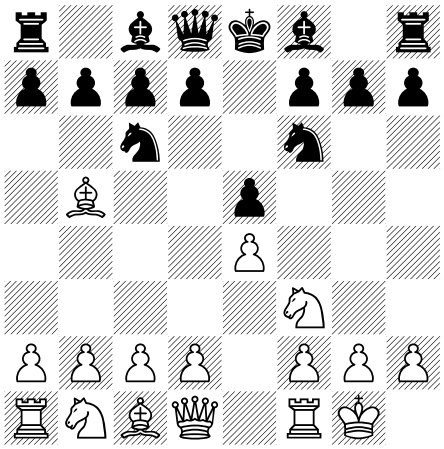
Zadání semestrální práce KIV/PT 2023/2024

Neznáte-li ještě Karlse Magnusena, je to mladý student FDULS a zapáleným hráčem šachu. Minulý semestr se seznámil se studentem Nakaru Hikamurem a přišel na to, že je velmi silný protivník a vhodný adept k tomu, aby se od něj něco z šachu přiučil. Již se naučil, že nemá akceptovat Englundův gambit, a že proti jeho variantě „Smažená játra“ hraje již jen Traxlerův protiútok, protože rád hraje ostře a uznává autora tohoto protiútoku, tohoto českého velikána šachu. Nakaru se nejprve tomuto zvláštnímu protiútoku smál, protože mu přišlo že Karls jen obětuje figury k ničemu, pak ale z dalšího průběhu byl tak vyděšený, že ho museli na konci partie skoro rozdýchávat.

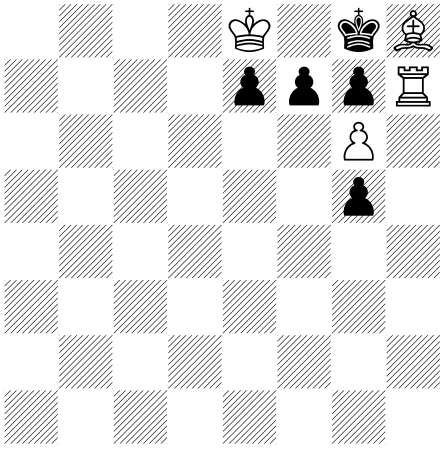
Jednou se chtěl Karls od Nakarua něco naučit a okoukat jeho šachovou přípravu a tak se připletl k jedné rozehrané partii Nakamury s Koprovem, a viděl následující pozici:



Jedná se o Berlínskou obranu, následovanou krátkou rošádou, tuto variantu ale Karls moc nestudoval. Přemýšlel ale, jak se do této pozice mohli oba hráči dostat. U této pozice na to přišel poměrně rychle:

1. ♙e4 ♙e5, 2. ♘f3, ♘c6, 3. ♗b5 ♘f6, 4. ♔♖O-O ...

Nedá mu ale spát jak je to s jinými pozicemi. Lze zjistit, jak se do dané pozice hráči dostali? Lze zjistit sled tahů, kterým se do dané pozice dostali? Lze zjistit sled který je nekratší možný? Všechny tyto otázky mu tzv. vrtají hlavou. Karls se o tomto problému bavil s Guditem Vujrathim jestli je to vždy možné, a protože on je znalec nevalidních pozic, připravil mu následující pozici a řekl, že Karls má bílé a on černé figury, pak řekl, že Karls je na tahu:



Této pozice (`4K1kB/4pppR/6P1/6p1/8/8/8/8 w - - 0 40`) nelze dosáhnout. Jaký zde byl tah Gudita, který do dané pozice vedl? Králem táhnout nemohl, protože neměl odkud, tři pěšáci na sedmé řadě jsou v počátečních pozicích (tzn. nemohly se od nikud pohnout) a pěšák na $g5$ nemohl přijít ze sloupce g. Nemohl ani přijít tím, že by vzal figuru ze sloupce f, protože předtím na pole $f6$ se nemohl dostat, protože nad ním je již dostatek pěšců. Nejvíce pravděpodobné je, že byl proveden tah $hxg6$, ale to také nelze provést, analýzu zde již nechám pro Vás. Dokážete Karlsovi s tímto složitým problémem backtrackování tahů pomoci?

Formální popis zadání:

Cílem semestrální práce je určit nejkratší možný sled valdiních šachových tahů takový, který vede do dané šachové pozice. Pokud existuje více variant, stačí zjistit jednu z nich. Pokud je cílová pozice nevalidní nebo se do ní nelze dostat z počáteční pozice, pak ohlašte chybu.

V případě složitějších pozicích (například ke konci hry) je potřeba pamatovat na to, aby během celého sledu tahů nedošlo ani jednou k nevalidnímu tahu, např. král je v šachu a následující tah toto nereflektuje. Také tahy jako rošáda či braní mimochodem mohou výrazně skrátit počet tahů v daném sledu.

Na vstupu (standardním, nikoliv v souboru) očekávejte validní[1] jednořádkový popis pozice dle Forsyth-Edwardsovy notace (FEN) zakončený znakem konce řádky, pro případ na obrázku by zápis vypadal následovně:

```
r1bqkb1r/pppp1ppp/2n2n2/1B2p3/4P3/5N2/PPPP1PPP/RNBQ1RK1 b kq - 5 4
```

Výstup vašeho programu na standardní výstup bude validní PGN zápis, např:

```
[Event "KIV/PT 2022/2023"]
[Site "Pilsen, CZE"]
[Date "2023.09.18"]
[Round "?"]
[White "Nakaru Hikamura"]
[Black "Anatali Koprov"]
[Result "*"]

1.e4 e5 2.Nf3 Nc6 3.Bb5 Nf6 {Berilnska obrana} 4.O-O
```

Komentáře typu {`Berlinska obrana`} nejsou vyžadovány, ale lze je uvádět. Pokud pozice není validní, nebo se do ní nelze dostat z počátečního rozestavení kamenů, pak vygenerujete stále validní PGN avšak bez jakéhokoliv tahu a s dostatečně deskriptivním komentářem, například výstupem by mohlo být následující:

```
[Event "KIV/PT 2022/2023"]
[Site "Pilsen, CZE"]
[Date "2023.09.18"]
[Round "?"]
[White "Nakaru Hikamura"]
[Black "Anatali Koprov"]
[Result "*"]

{Pozice neni validni, protoze cerny ma dva krale}
```

Užitečné odkazy:

- PGN formát: `http://www.saremba.de/chessgml/standards/pgn/pgn-complete.htm`
- FEN formát: `https://www.chessprogramming.org/Forsyth-Edwards_Notation`

[1] Validní syntakticky, ne nutně sémanticky

Minimální požadavky:

- Funkcionalita uvedená v zadání výše je **nutnou podmínkou pro finální odevzdání práce**, tj. program při finálním odevzdání musí splňovat veškeré požadavky/funkcionalitu výše popsanou, **jinak bude práce ohodnocena 0 body**.
- V případě, že bude program poskytovat řešení výše popsaného problému, 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:

Při průběžné kontrole bude vyžadováno provedení načtení vstupních dat do vhodné datové struktury a při kontrole bude hodnocena následující funkcionalita a náležitosti:

- Načtení dat ze standardního vstupu do vlastní struktury **(max. 5b)**,
- Vhodnost a implementaci použité datové struktury **(max. 5b)**,
- Efektivita při načítání a ukládání **(max. 5b)**,
- Čitelnost zdrojových kódů **(max. 5b)**,
- Spolupráce ve dvojici a rovnoměrné rozložení práce **(max. 5b)**.

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

- Navržený algoritmus řešení problému **(max. 10b)**,
- Implementace algoritmu řešení **(max. 10b)**,
- Validitu kódu s použitím PMD validátoru (viz Courseware) **(max. 5b)**.
- Rozpoznání pěti nevalidních vstupů při kontrole **(max. 5b)**,
- Nalezení pěti nekratších sekvencí pro validní vstupy **(max. 5b)**,
- Čitelnost zdrojových kódů **(max. 5b)**,
- Spolupráce ve dvojici a rovnoměrné rozložení práce **(max. 5b)**,
- V rámci strukturované dokumentace **(celkově 20 bodů)** bude hodnocení následující:
  - Celková analýzu a popis možných variant řešení problému **(max. 5b)**,
  - Důkladný popis algoritmu pro hledání nejkratšího sledu tahů **(max. 5b)**,
  - Doprovodné ilustrace **(max. 2b)**,
  - Stylistická forma a gramatika **(max. 2b)**,
  - Zadání, vhodný úvod a závěr **(max. 2b)**,
  - Ústní prezentace výsledků **(max. 4b)**.

Alternativní zadání semestrální práce pro KIV/PT 2023/2024

Zásobovací společnost *Sto roků v šachtě žil s. r. o.*, se specializuje na přepravu uhlí ze svých skladů do různých domácností a firem. Její majitel Petr Beznoha je ale velký skrblík, a tak nejraději využívá jako přepravní prostředky zahradní kolečka, která nejsou náročná na údržbu a provoz jako například Avie, které používá konkurence. Přepravované uhlí je uskladněno speciálních pytlích, které jsou nakládány na kolečka, aby se zaměstnanci neumazali. Beznoha používá jak kolečka s železnou korbou, s pozinkovanou korbou, také kolečka s korbou plastovou a různé další alternativy. V poslední době se však panu Beznohovi moc nedaří a, částečně i vlivem toho že po cestě mu spousta kovových koleček orezla vlivem dešťů, ho přeprava stojí spoustu peněz navíc.

Rozhodl se tedy, že je čas dát prostor moderním technologiím, a proto si chce nechat vytvořit program, který mu pomůže rozplánovat přepravu uhlí k zákazníkům tak, aby nepřišel o nějaká další kolečka, dodržel závazky a neztratil klientelu, maximálně využil nosnosti koleček a zároveň kolečka zbytečně neopotřebovával zbytečně dlouhými cestami. Tyto požadavky můžeme jednodušše označit jako snahu o minimalizaci „ceny“ přepravy.

Formální popis zadání:

Vytvořme pro Beznoha simulační program, který mu pomůže naplánovat přepravu, známe-li:

- počet skladů S,
- každý sklad bude definován pomocí:
  - kartézských souřadnic každého skladu x_s a y_s,
  - počtu pytlů 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 pytlů uhlí,
  - doby t_n, která udává, jak dlouho trvá daný typ pytle na kolečko naložit/vyložit (každý sklad může používat jiný typ pytlů, se kterým může být různě obtížná manipulace),
- počet zákazníků Z,
- kartézské souřadnice každého zákazníka x_z a y_z,
- počet přímých cest v mapě C,
- seznam cest, přičemž každá cesta je definována indexy i a j, označujících místa (zákazník, sklad) v mapě, mezi kterými existuje přímé propojení a platí: $i, j \in \{1, \ldots, S, S+1, \ldots, S+Z\}$, tj. sklady jsou na indexech od 1 do S a zákazníci na indexech od $S+1$ do $S+Z$. Pozn. pokud existuje propojení z místa i do místa j, pak existuje i propojení z místa j do místa i,
- počet druhů koleček D,
- informace o každém druhu kolečka, kterými jsou:
  - slovní označení druhu kolečka, které bude uvedeno jako jeden řetězec neobsahující bílé znaky,
  - minimální $v_{\min}$ a maximální $v_{\max}$ rychlost, kterou se může daný druh kolečka pohybovat, přičemž kolečko se pohybuje konstantní rychlostí, která mu bude vygenerována v daném rozmezí pomocí rovnoměrného rozdělení,
  - minimální $d_{\min}$ a maximální $d_{\max}$ vzdálenost, kterou může daný druh kolečka překonat, dokud nebude potřebovat provést údržbu, přičemž pro každý druh kolečka je tato doba opět konstantní a je vygenerována v daném rozmezí pomocí normálního rozdělení se střední hodnotou $\mu = (d_{\min} + d_{\max})/2$ a směrodatnou odchylkou $\sigma = (d_{\max} - d_{\min})/4$,
  - doba t_d, udávající kolik času dané kolečko potřebuje pro provedení údržby,
  - počet pytlů k_d udávající maximální zatížení daného druhu kolečka,
  - hodnota p_d udávající procentuální zastoupení daného druhu kolečka ve vozovém parku firmy, přičemž platí: $\sum_{d=1}^{D} p_d = 100\%$,
- počet požadavků k obsloužení 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 zákazníka $z_p \in \{1, \ldots, Z\}$ kterému má být požadavek doručen,
- množství pytlů k_p, které zákazník požaduje,
- doby t_p udávající, za jak dlouho po příchodu požadavku musí být pytle doručeny (tj. nejpozději v čase $t_z + t_p$ musí být pytle vyloženy u zákazníka).

Za úspěšně ukončenou simulace se považuje moment, kdy jsou všechny požadavky obsloužené a všechna kolečka jsou vrácena 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>, Zakaznik: <z>, Pocet pytlu: <p>, Deadline: <t+t_p>
  ```

- Ve skladu se začíná připravovat kolečko:

  ```
  Cas: <t>, Kolecko: <k>, Sklad: <s>, Nalozeno pytlu: <p>, Odjezd v: <t+k*t_n>
  ```

- Zaměstnanec firmy dovezl kolečko k zákazníkovi, 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>, Kolecko: <k>, Zakaznik: <z>, Vylozeno pytlu: <p>, Vylozeno v: <t+k*t_n>,
  Casova rezerva: <t_r>
  ```

- Kolečko dojelo do skladu, kde ale na další cestu vyžaduje údržbu (nelze servisovat na cestě ani u zákazníka):

  ```
  Cas: <t>, Kolecko: <k>, Zakaznik: <z>, <druh> kolecko vyzaduje udrzbu,
  Pokracovani mozne v: <t+t_d>
  ```

- Kolečko projelo okolo zákazníka, ale nemá zde žádný zvláštní úkol:

  ```
  Cas: <t>, Kolecko: <k>, Zakaznik: <z>, Kuk na <druh> kolecko
  ```

- Kolečko dokončilo cestu a vrátilo se do skladu:

  ```
  Cas: <t>, Kolecko: <k>, Navrat do skladu: <s>
  ```

- Požadavek se nepodařilo (z jakéhokoli důvodu) obsloužit včas:

  ```
  Cas: <t>, Zakaznik <z> umrzl zimou, protoze jezdit s koleckem je hloupost, konec
  ```

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

```
...
Cas: 12, Pozadavek: 2, Zakaznik: 2, Pocet pytlu: 3, Deadline: 30
Cas: 12, Kolecko: 5, Sklad: 3, Nalozeno pytlu: 3, Odjezd v: 18
Cas: 21, Kolecko: 5, Zakaznik: 1, Pozinkovane kolecko vyzaduje udrzbu, Pokracovani mozne v: 22
Cas: 23, Kolecko: 5, Zakaznik: 10, Kuk na pozinkovane kolecko
Cas: 24, Kolecko: 5, Zakaznik: 2, Vylozeno pytlu: 3, Vylozeno 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 zákazníků, informacemi o cestách, kolečkách, 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 zva žujte časovou a paměťovou náročnost algoritmů pracujících s danými strukturami **(10 bodů)**.
- Proveďte základní simulaci jedné obslužné trasy včetně návratu kolečka do skladu. Vypište celkový počet doručených pytlů > 0 a celkový počet obsloužených požadavků > 0. Trasa kolečka 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 některého zákazníka 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 koleček - základní údaje o kolečkách (druh; id domovského skladu; rychlost; max. vzdálenost, kterou urazí před další údržbou), uskutečněné trasy (čas, kdy opustilo sklad; kudy jelo; kolik toho vezlo; kam a kdy doručovalo zboží; kde a kdy se zastavilo kvůli údržbě; kdy se vrátilo do svého domovského skladu), jak dlouho za celou dobu simulace nikam nejelo (tj. bylo ve skladu a čekalo na přiřazení požadavku) a celkovou vzdálenost, kterou ujelo,
  - přehled jednotlivých zákazníků - čas a velikost vzniklého požadavku; kdy musel být nejpozději doručen; kdy byl skutečně doručen; ze kterého skladu a kterým kolečkem byl obsloužen,
  - přehled jednotlivých skladů - časy, kdy došlo k doplnění skladu; kolik pytlů v té době ve skladu zbývalo a kolik jich je k dispozici po doplnění,
  - délka trvání celé simulace, celková doba, kdy kolečka stála ve skladech, celková ujetá vzdálenost, kolik koleček 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 Courseware v sekci 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)**.