IZLO – Projekt 1: SAT solvery

Úvod

Váš strýček Pat Vás požádal o pomoc s řešením svého problému. Jako pošťák vstává brzy ráno a i se svým kocourem vyráží rozvážet poštu. Dokud byl ještě mladý a plný sil, příliš se netrápil s plánováním své trasy. Nyní, těsně před důchodem, si Vás jednoho dne zavolal s následující prosbou:

„Poslyš, ty rozumíš těm počítačům, nemohl/a bys mi nějak pomoct s plánováním trasy tak, abych projel každou ulicí ve městě, kde mám rozvést poštu právě jednou a nemusel se neustále někam vracet?"

Díky znalostem nabytým během svého studia jste se rozhodl/a problém řešit pomocí převodu na problém splnitelnosti ve výrokové logice s použitím SAT solveru.

Zadání

Instance problému se skládá z čísla K, $K > 0$, značícího počet křižovatek a seznamu S ulic délky U, $U > 0$. Čísla $0, 1, \ldots, K-1$ reprezentují křižovatky. Ulice je reprezentována dvojicí čísel (z, k), kde z je její začátek a k je její konec (platí tedy $0 \leq z < K$ a $0 \leq k < K$). Zadaný seznam ulic je tedy formy $S = (z_0, k_0), \ldots, (z_{U-1}, k_{U-1})$. Pro jednoduchost je ulice vždy průjezdná pouze jedním směrem, je tedy identifikována svou počáteční a koncovou křižovatkou; mezi dvěmi křižovatkami dále existuje v jednom směru nejvýše jedna ulice.
Řešením problému je cesta, která prochází přes každou zadanou ulici právě jednou (začínat může kdekoliv).

Vaším úkolem je vytvořit program, který pro instanci tohoto problému vygeneruje formuli, jejíž modely jsou právě řešeními dané instance. K dispozici již máte kostru (níže), která se stará o zpracování vstupu a generování formule ve formátu DIMACS (popis např. zde).

V rámci kostry již máte připraveno kódování problému do výrokových proměnných. Pro každou trojici čísel i, z, k, kde $0 \leq i < U$, $0 \leq z < K$, a $0 \leq k < K$, máme proměnnou $x_{i,z,k}$ s následující sémantikou:

$x_{i,z,k} = 1 \iff$ V i-tém kroce cesty je zvolena ulice začínající v křižovatce z a končící v křižovatce k.

Vaším úkolem je doplnit kód tří funkcí v souboru `code/add_conditions.c`, které se starají o generování následujících podmínek:

1. *V každém kroku cesty je zvolena alespoň jedna **existující** ulice.* Generování této formule je potřeba doplnit do funkce `at_least_one_valid_street_for_each_step(...)`.
2. *V každém kroku cesty je zvolena maximálně jedna ulice.* Generování této formule je potřeba doplnit do funkce `at_most_one_street_for_each_step(...)`.
3. *Ulice v rámci cesty na sebe navazují.* Přesněji, pokud ulice (z, k) je i-tým krokem cesty, kde $0 \leq i < U - 1$, potom ulice v kroce $i + 1$ začíná v k. Generování této formule doplňte do funkce `streets_connected(...)`.
4. *Žádná ulice není navštívená vícekrát než jednou.* Generování této podmínky je již implementováno ve funkci `streets_do_not_repeat(...)` jako ukázka. Vygenerovaná formule má podobu:

$$\bigwedge_{0 \leq i < U} \bigwedge_{0 < i_2 < U, i_2 \neq i} \bigwedge_{0 \leq z < K} \bigwedge_{0 < k < K} (\neg x_{i,z,k} \vee \neg x_{i_2,z,k}).$$

   Jinými slovy, pro každé dva různé kroky cesty $i \neq j$ a každou dvojici křižovatek (z, k), ulice daná (z, k) může být zvolena v nejvýše jednom z kroků i a j.

Výše zmíněné funkce jsou jediné části kódu, které mají být modifikovány.

Kostra

Kostra projektu je ke stažení zde.

Formát vstupu

Instance problému je popsána v následujícím textovém formátu. Na první řádce souboru se nachází hlavička obsahující přesně dvě přirozená čísla udávající postupně celkový počet křižovatek a celkový počet ulic. Následuje seznam ulic, kde se na každém řádku nachází jedna ulice reprezentovaná jako dvojice čísel (v rozsahu vymezeném hlavičkou):

```
<pocet_krizovatek> <pocet_ulic>

<zacatek_ulice_1> <konec_ulice_1>
<zacatek_ulice_2> <konec_ulice_2>
...
```

Konkrétní příklad vstupu je následující soubor:

```
4 4

0 1
1 2
2 1
2 3
```

Pro tento vstup neexistuje řešení neboť ulici $(1, 2)$ je potřeba projít dvakrát.

Generování formulí

Jelikož formát DIMACS pracuje s proměnnými indexovanými přirozenými čísly, proměnná $x_{i,z,k}$ je převedena na proměnnou s indexem n, kde $n = i \cdot K^2 + z \cdot K + k + 1$. S těmito proměnnými však nebudete pracovat přímo, ale pomocí funkcí popsaných v následujícím odstavci. Nicméně způsob reprezentace proměnných se může hodit, pokud si budete chtít projít vygenerovaný DIMACS soubor (ve vygenerovaném souboru jsou na řádcích začínajících symbolem `c` komentáře, kde je kopie vstupního problému a popis mapování čísel proměnných na kroky a ulice).

Všechny formule ze zadání je potřeba vygenerovat v konjunktivní normální formě (CNF), která je standardním vstupem SAT solverů. Pro reprezentaci formule jsou již v kostře projektu vytvořeny potřebné struktury. Pro manipulaci s těmito strukturami jsou k dispozici následující funkce:

- `Clause *create_new_clause(CNF *f)` – Vytvoří novou prázdnou (disjunktivní) klauzuli ve formuli `f`. **POZOR**: prázdná klauzule je ekvivalentní 0 (`false`).
- `void add_literal_to_clause(Clause *c, bool is_positive, unsigned i, unsigned k, unsigned z)` – Vloží do klauzule `c` literál $x_{i,k,z}$ (pokud `is_positive = true`), nebo $\neg x_{i,k,z}$ (pokud `is_positive = false`).

Testování

Referenčním SAT solverem je MiniSat[1]. Na linuxových systémech založených na Debianu lze tento nástroj nainstalovat jednoduše příkazem `apt-get install minisat`, na Windows lze použít stejný postup ve WSL[2], na macOS lze použít `brew install minisat`. Pro vyzkoušení lze také využít online rozhraní (pak ale nelze využít automatické testy popsané níže).

K ověření základní funkčnosti vašeho řešení je možné použít dva přiložené skripty. Tyto skripty vyžadují `python3` a nainstalovaný MiniSat dostupný v `PATH`. Po přeložení vašeho řešení je možné využít skript `run.sh`, který spustí vaše

řešení nad vstupním souborem a následně spustí SAT solver nad vygenerovanou formulí. V případě, že vygenerovaná formule je splnitelná, skript vypíše model jako lidsky čitelné řešení problému (jednotlivé kroky cesty) a zkontroluje jestli model splňuje podmínky specifikované zadáním. V případě, že ne, je vypsána první porušená podmínka. Příkazem `make test` je možné ověřit základní funkcionalitu vašeho řešení na jednoduchých vstupech. Tyto vstupy lze nalézt ve složkách `tests/sat` a `tests/unsat`.

Další pokyny a doporučení

- Potřebné formule jdou vytvořit téměř přímo v CNF, nemělo by být potřeba aplikovat distributivní zákony pro úpravu do CNF. Před samotným programováním si zkuste formule nejprve napsat na papír.
- Generování formulí není nutné optimalizovat. Vaše řešení nebude testováno na větších vstupech než jsou ty, které máte k dispozici.
- Odevzdáváte pouze soubor `add_conditions.c` do IS VUT.
- Své řešení vypracujte samostatně. Odevzdané projekty budou kontrolovány proti plagiátorství, za nějž se považuje i sdílení vlastního řešení s ostatními studenty.
- Před odevzdáním si zkontrolujte zda váš soubor jde opravdu přeložit s přiloženým Makefile.
- Případné dotazy směřujte do fóra „Fórum k látce a projektům".

---

1. http://minisat.se/↩
2. https://ubuntu.com/wsl↩