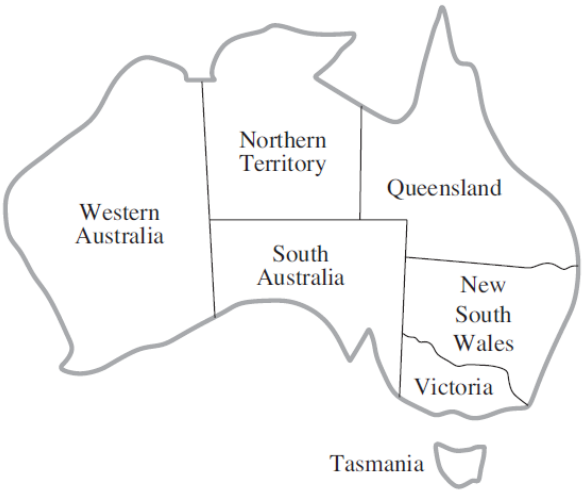
Úlohy na cvičení k **Umělé inteligenci (NI-UMI)**

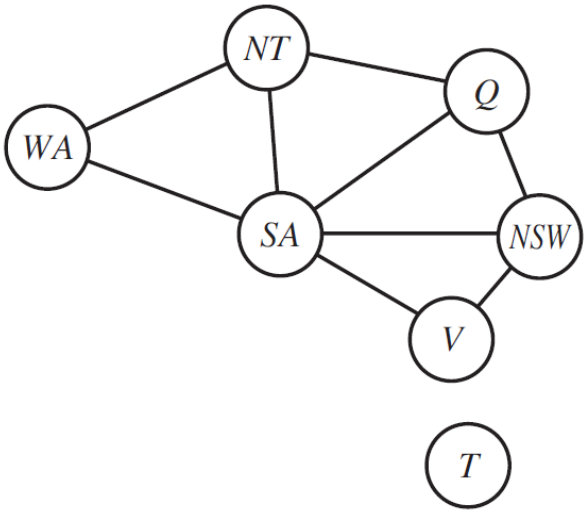
Téma: **Splňování omezení (cvičení 2)**

Uvádíme důležité úlohy. Na cvičení byly probírány další úlohy a celkově bylo řečeno mnohem víc, proto se navštěvování cvičení vřele doporučuje.

1. Procvičte si důležité algoritmy pro řešení problému CSP.

a) Rozmyslete si, jak fungují **backtracking** (BT), **backjumping** (BJ), backtracking s udržováním hranové konzistence (**MAC-BT**), případně dynamický backtracking (**DBT**), či závislostmi řízený backtracking (**DDBT**).
b) Simulujte BT, BJ, MAC-BT, případně DBT a DDBT na úloze barvení mapy Austrálie (viz. následující obrázek) třemi barvami (R,G,B). Snažte se o takovou simulaci (ovlivnit ji můžete pořadím proměnných k ohodnocení a pořadím hodnot), kde vyniknou rozdíly mezi jednotlivými algoritmy.
c) Vybrané algoritmy implementujte a otestujte jejich běh na úloze barvení mapy (opět lze použít Austrálii, nebo jiný graf).



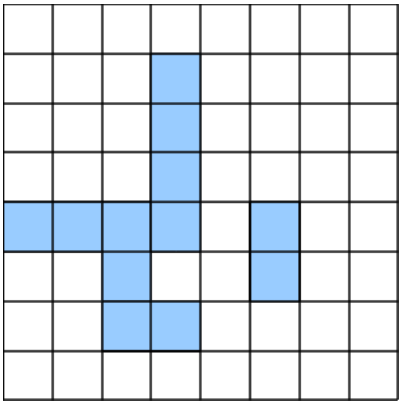


2. Vyzkoušejte si algoritmy BT, BJ, MAC-BT, případně DBT, DDBT na SUDOKU. Vstupní zadání SUDOKU můžete nalézt ve veřejně dostupných sbírkách, jako počáteční inspirace může posloužit následující obrázek:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

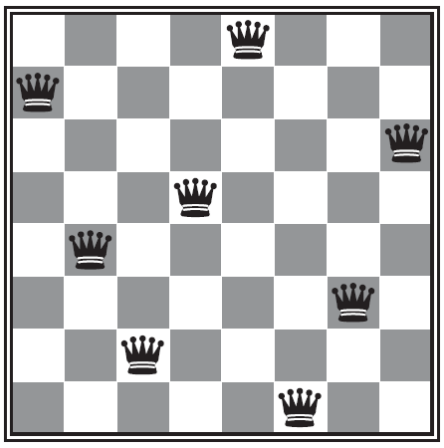
a) Formulujte problém a implementujte algoritmy s binárními podmínkami nerovnosti.
b) Formulujte problém a implementujte algoritmy s podmínkou **allDifferent**. Lze využít i knihovnu pro řešení CSP jako je například Gecode, z3, ILOG nebo řešiče CSP jako je Minion, MiniZinc atd.

3. Je dán konečný slovník slov S, jako například S = { ALFA, BETA, FIT, CVUT, ILOVEAI, IHATEAI, …}. Dále je dána čtvercová mřížka s překážkami, ve které budeme z dostupných slov ve slovníku sestavovat křížovku. Přitom chceme mřížku zcela vyplnit.



a) Formulujte úlohu jako CSP, jak budou vypadat proměnné, doména a podmínky. Uvažujte možnost, kdy každé slovo ze slovníku můžeme použít libovolněkrát a možnost, že každé slovo smí být použito nejvýše jednou.
b) Otestujte na úloze některý z řešících algoritmů.

4. N-královen formulujte jako CSP. Jak budou vypadat proměnné, podmínky a doména. Lze využít podmínku allDifferent? Pokud ano, implementujte řešící algoritmus založený na backtrackingu s udržováním zobecněné hranové konzistence, tedy **MGAC-BT**.



a) Vyskytují se v úloze nějaké symetrie (symetrická řešení). Pokud ano, navrhněte podmínky na rozbití těchto symetrií.
b) Rozbíjení symetrií zabudujte do řešícího algoritmu.

5. Formulujte jako CSP problém hledání Hamiltonovské kružnice v neorientovaném grafu G=(V,E). Opět můžete otestovat na úloze některý z řešících algoritmů.

6. Uvažujte ternární podmínku A+B=C s doménami proměnných D={1,2,…,10}. Pokuste se podmínku binarizovat, tj. nahradit binárními podmínkami s případným využitím pomocných proměnných (nápověda: můžete uvažovat, že prvky nové domény jsou dvojice, trojice, atd. prvků původní domény).

7. Náhodný binární CSP je zadán čtyřmi parametry: počet proměnných $n \in \mathbb{N}$, velikost domény $m \in \mathbb{N}$, pravděpodobnost přítomnosti náhodné podmínky mezi dvojicí proměnných $p_c \in \mathbb{R}_0^+$ (máme celkem $n(n-1)/2$ dvojic proměnných) a pravděpodobnost splnění uspořádané dvojice hodnot náhodnou podmínkou $p_d \in \mathbb{R}_0^+$ (celkem máme m^2 dvojic hodnot). Předpokládáme, že náhodné podmínky v náhodném CSP mohou být různé.

a) Implementujte generátor náhodných CSP podle zadaných parametrů (n, m, p_c, p_d)
b) Na náhodných CSP otestujte vybraný algoritmus (BT, BJ, MAC-BT, …)
c) Pro dané netriviální ale zase ne příliš velké hodnoty parametrů n a m nalezněte hodnoty p_c a p_d tak, aby výsledný náhodný CSP byl co nejtěžší, tj. vybraný algoritmus jej řešil co nejdéle (takové hodnoty p_c a p_d budou tvořit fázový přechod)