Dominik Wawszczak
numer indeksu: 440014
numer grupy: 1

Algorytmika
praca domowa 4

Zadanie 1

Skoro w każdej klauzuli każda zmienna występuje w co najwyżej jednym literale, to zbiór X jest zbiorem rozwiązującym wtedy i tylko wtedy, gdy każda klauzula zawiera co najwyżej dwie zmienne niebędące w X. Wynika z tego, że jeśli X nie jest zbiorem rozwiązującym, to istnieje klauzula, która zawiera przynajmniej 3 zmienne niebędące w X. Obserwacja ta prowadzi do następującego algorytmu:

Algorithm 1 SolvingSet

procedure $\textsc{SolvingSet}(k, X = \emptyset)$
    if every clause has at most two variables not in X **then**
        return true
    end if
    if $k \leqslant 0$ **then**
        return false
    end if
    $(a, b, c) \leftarrow$ three variables not in X from one clause
    return $\textsc{SolvingSet}(k-1, X \cup \{a\})$ **or** $\textsc{SolvingSet}(k-1, X \cup \{b\})$ **or** $\textsc{SolvingSet}(k-1, X \cup \{c\})$
end procedure

Algorytm ten wykorzystuje fakt, że jeśli aktualny zbiór X jest podzbiorem pewnego zbioru rozwiązującego, to przynajmniej jedna ze zmiennych a, b lub c musi być w tym zbiorze.

W drzewie rekurencji tego algorytmu, na i-tym poziomie głębokości jest co najwyżej 3^i wierzchołków, dla $i = 0, \ldots, k$. Łącznie daje to maksymalnie $\frac{3^{k+1}-1}{2}$ wywołań rekurencyjnych. W każdym wywołaniu musimy jedynie znaleźć odpowiednią klauzulę i wybrać z niej trzy zmienne, bądź stwierdzić, że taka klauzula nie istnieje, co oczywiście da się zrobić w czasie $|\varphi|^{O(1)}$. Ostateczna złożoność czasowa naszego algorytmu wynosi zatem $3^k \cdot |\varphi|^{O(1)}$, co kończy rozwiązanie pierwszej części zadania.

Przejdźmy teraz do rozwiązania drugiej części zadania. Bez straty ogólności możemy założyć, że dekompozycja drzewowa jest ładna. Skorzystamy z programowania dynamicznego. Niech v będzie dowolnym wierzchołkiem dekompozycji drzewowej. Dla każdej pary funkcji $f : \text{zmienne}(v) \to \{0,1\}$ oraz $g : \text{klauzule}(v) \to \{0,1,2\}$ definiujemy

$$\mathrm{dp}(v, f, g) \;=\; \min\left\{|X| \;:\; X \cap \text{zmienne}(v) = f^{-1}(1) \;\wedge \underset{l \in \text{klauzule}(v)}{\forall} |l \setminus X| = g(l) \;\wedge \underset{l \in \text{klauzule}(T_v)}{\forall} |l \setminus X| \leqslant 2\right\},$$

gdzie T_v jest poddrzewem wierzchołka v, a klauzule traktujemy jako zbiory zmiennych, które zawierają.

Tak zdefiniowaną tablicę można łatwo wypełnić, rozważając 5 przypadków (dodanie/usunięcie zmiennej/klauzuli oraz wierzchołek łączący dwóch synów). Na koniec wystarczy odczytać wartość z $\mathrm{dp}(\text{root}, \emptyset, \emptyset)$ i jeśli jest ona nie większa niż k, to odpowiedzią jest „Tak". W przeciwnym wypadku odpowiedzią jest „Nie".

Całkowita złożoność naszego algorytmu to $3^t \cdot |\varphi|^{O(1)}$, co kończy rozwiązanie zadania.

Zadanie 2