

Imię, nazwisko, nr indeksu

(kod)

(9 punktów) Wybierz 9 z poniższych pytań i wybierz odpowiedź tak/nie (bez uzasadnienia). Za prawidłowe odpowiedzi dajemy +1 punkt, za złe -1 punkt. Punkty policzymy za 9 najgorszych odpowiedzi, czyli nie opłaca się odpowiadać na więcej niż 9 pytań.

1. Czy całkowicie rozstrzygalny jest problem: dany automat ze stosem rozpoznający  $L$ , pytanie czy  $L = L^*$ .

**Odpowiedź.** Nie. Problem jest nierozstrzygalny, bo jest co najmniej tak samo trudny jak problem uniwersalności dla automatów ze stosem, który jest nierozstrzygalny. Przypomnijmy, że problem uniwersalności brzmi: "dany automat ze stosem rozpoznający  $L$ , pytanie czy  $L$  zawiera wszystkie słowa". Oto algorytm rozstrzygający problem uniwersalności, który korzysta z algorytmu dla problemu z zadania. Niech  $L$  język rozpoznawany przez automat ze stosem  $\mathcal{A}$ . Algorytm ma dwa kroki.

- (a) Sprawdzamy, czy  $L$  zawiera słowo puste i każde słowo jednoliterowe. Jeśli, nie to  $L$  nie jest zawiera wszystkich słów i algorytm odpowiada "nie". Ten krok można zrobić w czasie skończonym (nawet wielomianowym), bo problem "czy dany automat ze stosem  $\mathcal{A}$  akceptuje słowo  $w$ " jest rozstrzygalny w czasie wielomianowym.
- (b) Jeśli  $L$  zawiera wszystkie słowa jednoliterowe, to wówczas  $L^* = A^*$ , a więc problem uniwersalności ma tę samą odpowiedź co problem  $L = L^*$ , a więc możemy skorzystać z algorytmu dla problemu z zadania.

2. Czy całkowicie rozstrzygalny jest problem: dany automat skończony rozpoznający  $L$ , pytanie czy  $L^* = (LL)^*$ .

**Odpowiedź.** Tak. Problem jest całkowicie rozstrzygalny. Najpierw automat zmieniamy na wyrażenie regularne dla  $L$ , potem obliczamy wyrażenia regularne dla  $L^*$  oraz  $(LL)^*$ , następnie oba wyrażenia zmieniamy na automaty deterministyczne, minimalizujemy, i sprawdzamy czy w obu przypadkach wyszedł ten sam automat.

3. Jeśli  $L$  jest całkowicie rozstrzygalny, oraz  $K \subseteq L$ , to czy  $K$  też jest całkowicie rozstrzygalny?

**Odpowiedź.** Nie. Problem może być nierozstrzygalny. Na przykład zbiór wszystkich słów jest całkowicie rozstrzygalny, a ma nierozstrzygalne podzbiory.

4. Czy równoważne (pod względem akceptowanych języków) są deterministyczne automaty ze stosem i niedeterministyczne automaty ze stosem?

**Odpowiedź.** Nie. Nie są równoważne, bo języki rozpoznawane przez deterministyczne automaty ze stosem są zamknięte na dopełnienie, a niedeterministyczne nie. Oto dłuższe uzasadnienie. Dopełnienie języka  $a^n b^n c^n$  to język

$$\{a^i b^j c^k : \text{liczby } i, j, k \text{ nie są wszystkie równe}\} \cup (\{a, b, c\}^* - a^* b^* c^*),$$

który jest rozpoznawany przez niedeterministyczny automat ze stosem. Gdyby ten język był też rozpoznawany przez deterministyczny automat ze stosem, to i jego dopełnienie byłoby rozpoznawane, a wiemy że język  $a^n b^n c^n$  nie jest bezkontekstowy.

5. Czy klasa języków całkowicie rozstrzygalnych zamknięta jest na wszystkie z operacji: suma, dopełnienie, przecięcie, gwiazdka?

**Odpowiedź.** Tak. Zróbmy na przykład dopełnienie i gwiazdkę. Dla dopełnienia bierzemy maszynę deterministyczną i zamienamy stany akceptujące z odrzucającymi. Ponieważ nie ma niedeterminizmu ani pętlenia, konstrukcja daje właściwy wynik. Zróbmy teraz gwiazdkę. Dostajemy słowo  $w$ , pytanie czy  $w \in L^*$ , dla języka całkowicie rozstrzygalnego  $L$ . Powolny algorytm robi to tak: sprawdza wszystkie podziały  $w$  na concatenację kilku słów  $w = w_1 \cdots w_n$  (wykładniczo wiele sposobów) i dla każdego z nich sprawdza czy wszystkie słowa  $w_1, \dots, w_n$  należą do  $L$ . Bardziej algorytmicznie nastawiony czytelnik zauważy, że stosując algorytm dynamiczny można rozważać wielomianowo wiele podziałów.

6. Czy należy do PSPACE problem: dany automat niedeterministyczny, pytanie czy akceptuje wszystkie słowa?

**Odpowiedź.** Tak. Korzystamy z tego, że deterministyczny PSPACE jest równy niedeterministycznemu PSPACE. Ponieważ PSPACE jest zamknięty na dopełnienie (z determinizmu), wystarczy pokazać algorytm PSPACE, który sprawdza czy automat odrzuca pewne słowo. Dla problemu odrzucania słowa, możemy natomiast skorzystać z niedeterministycznego algorytmu PSPACE. Niech  $Q$  będą stanami danego automatu niedeterministycznego. Algorytm zgaduje kolejne litery  $a_1, a_2, \dots$  słowa, i dla każdego  $i$  oblicza zbiór stanów  $Q_i \subseteq Q$  osiągalnych po prefiksie  $a_1 \cdots a_i$ . To wszystko można zrobić w pamięci wielomianowej, bo obliczywszy  $Q_i$  można zapomnieć o  $Q_{i-1}$  i wszystkich poprzednich literach. Jak algorytm dojdzie do zbioru  $Q_i$  który nie zawiera żadnego stanu akceptującego, to akceptuje. Ten algorytm jest już poprawnym rozwiązaniem. Niektórych jednak może irytować fakt, że algorytm potrafi się pętlić. To można poprawić robiąc obserwację (patrz zadanie z gwiazdką spoza testu), że jeśli automat odrzuca słowo, to odrzuca słowo długości  $2^{|Q|}$ , a więc algorytm może przestać generować nowe litery po  $2^{|Q|}$  krokach.

7. Czy dla każdego  $n$  istnieje gramatyka bezkontekstowa rozmiaru  $O(n)$ , której najkrótsze generowane słowo jest długości co najmniej  $2^n$ ?

**Odpowiedź.** Tak. Gramatyka ma  $n$  nieterminali  $X_1, \dots, X_n$  i następujące reguły:

$$X_1 \rightarrow X_2 X_2 \quad X_2 \rightarrow X_3 X_3 \quad \cdots \quad X_{n-1} \rightarrow X_n X_n \quad X_n \rightarrow aa.$$

Gramatyka ta generuje dokładnie jedno słowo, jest nim  $a^{2^n}$ . To jest właściwie granica możliwości każdej gramatyki: najkrótsze generowane słowo jest zawsze długości wykładniczej ze względu na rozmiar gramatyki.

8. Czy dla każdego  $n$  istnieje gramatyka bezkontekstowa rozmiaru  $O(n)$ , której najkrótsze niegenerowane słowo jest długości co najmniej  $2^n$ ?

**Odpowiedź.** Tak. Gramatyka ma  $n$  nieterminali  $X_1, \dots, X_n$  i następujące reguły:

$$X_1 \rightarrow X_2 X_2 \quad X_2 \rightarrow X_3 X_3 \quad \dots \quad X_{n-1} \rightarrow X_n X_n \quad X_n \rightarrow aa|a|\epsilon.$$

Gramatyka te generuje słowa postaci  $a^i$  dla  $i \leq 2^n$ . Wykraczając poza zadanie, można też zrobić gramatykę, gdzie najkrótsze niegenerowane słowo jest rozmiaru  $2^{2^n}$ , a nawet dłuższe niż  $f(n)$  dla dowolnej funkcji obliczalnej.

9. Rozważmy (skierowany) graf konfiguracji maszyny Turinga (wierzchołki to konfiguracje, krawędzie to jeden krok obliczeń). Konfiguracja osiągalna to taka, do której można dojść z konfiguracji postaci: słowo wejściowe na taśmie, głowica na początku taśmy w stanie początkowym. Maszynę Turinga nazwiemy odwracalną, jeśli do każdej osiągalnej konfiguracji wchodzi dokładnie jedna krawędź. Czy każdy język całkowicie rozstrzygalny można rozpoznawać maszyną deterministyczną odwracalną (maszyną totalną, czyli taką, która każdy bieg kończy akceptacją lub odrzuceniem w skończonym czasie)?

**Odpowiedź.** Tak. Niech  $L$  będzie językiem całkowicie rozstrzygalnym, rozpoznawanym przez maszynę deterministyczną i totalną  $M$ . Przeróbmy maszynę  $M$  tak, żeby z każdej konfiguracji można było odtworzyć słowo wejściowe. Na przykład maszyna może zostawić słowo wejściowe na taśmie i pisać tylko symbole spoza alfabetu wejściowego. To znaczy, jeśli maszyna dostaje na wejściu słowo  $w$ , to wszystkie kolejne zawartości taśmy w trakcie obliczenia są postaci  $wv$ , gdzie  $v$  korzysta tylko z tej części alfabetu roboczego, która nie jest alfabetem wejściowym. Nowa maszyna ma tę własność, że patrząc na jej konfigurację (zawartość taśmy, położenie głowicy i stan) możemy odtworzyć poprzednią konfigurację. Wystarczy bowiem przeczytać słowo wejściowe (które jest do odtworzenia z zawartości taśmy), wykonać od nowa całe obliczenie aż do bieżącej konfiguracji i zwrócić przedostatnią konfigurację. A więc maszyna jest odwracalna.

10. Rozważmy wariant wyrażeń regularnych, gdzie oprócz sumy, dostępna jest też część wspólna języków. Czy takie wyrażenia generują tylko języki regularne?

**Odpowiedź.** Tak, bo klasa języków regularnych jest zamknięta na przecięcie.

11. Rozważmy deterministyczne automaty z wieloma stosami, które akceptują przez stan akceptujący. Czy dla każdego automatu z 3 stosami istnieje równoważny automat z 2 stosami?

**Odpowiedź.** Tak. Oto dwa możliwe uzasadnienia. Pierwsze jest takie, że już z 2 stosami dostajemy model tak samo silny jak maszyny Turinga,

a maszyny Turinga umieją symulować 3 a nawet więcej stosów. Drugie uzasadnienie jest konstrukcją bezpośrednią. Załóżmy, że mamy zawartość 3 stosów w słowach  $w_1, w_2, w_3$ . Wówczas trzymamy na pierwszym stosie słowo  $w_1\#w_2\#w_3$ . Korzystając z drugiego stosu jako stosu pomocniczego, jesteśmy w stanie zasymulować dowolną operację stosową.

12. Załóżmy  $P \neq NP$ , bo inaczej zadanie jest oczywiste. Niech  $A, B$  skończone alfabetu. Rozważmy funkcję  $f : A \rightarrow B$ . Stosujemy ją do słów litera po literze (funkcja zachowuje długość słowa), oraz do języków słowo po słowie. Czy języki postaci  $f(L)$ , gdzie  $L$  jest w PTIME, mogą być NP-trudne?

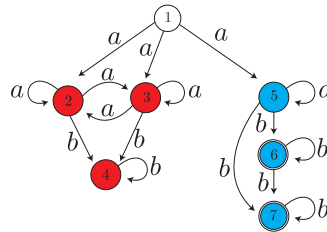
**Odpowiedź.** Tak. Rozważmy język  $L$  słów postaci

$$b_1 \cdots b_n \varphi$$

gdzie  $b_1, \dots, b_n$  są w  $\{0, 1\}$ , oraz  $\varphi$  jest zapisem formuły o  $n$  zmiennych, prawdziwej przy wartościowaniu  $b_1 \dots b_n$ . Załóżmy, że formułę  $\varphi$  zapisujemy za pomocą alfabetu  $A$ , który nie zawiera liter  $\{0, 1\}$ . Język  $L$  jest w PTIME, bo to jest po prostu ewaluacja formuł. Z drugiej strony, po zastosowaniu funkcji  $f$ , która jest identycznością na  $A$ , a która obie litery  $\{0, 1\}$  zastępuje symbolem  $?$ , dostajemy problem SAT, który jest NP-zupełny.

13. Rozważmy model niedeterministycznego automatu, gdzie kryterium akceptacji to “co najmniej połowa biegów akceptuje, spośród tych które do dochodzą do końca słowa”. Czy taki automat rozpoznaje wyłącznie (choć niekoniecznie wszystkie) języki bezkontekstowe?

**Odpowiedź.** Nie. Rozważmy automat, który wygląda tak:



Będzie nas głównie interesowało zachowanie automatu na słowach postaci  $a^+b^+$ . Pokażemy, że z nowym kryterium akceptacji, automat akceptuje język  $L$  o własności:

$$L \cap a^+b^+ = \{a^n b^m : 2^n \leq m + 1, m \geq 1, n \geq 1\}.$$

Kluczowa własność automatu jest taka, że w czerwonej części jest niedeterminizm wykładniczy, a w niebieskiej niedeterminizm liniowy (patrz niżej co to znaczy). Przy każdej literze  $a$ , automat rozdwaja się na dwie kopie w czerwonej części (bo ze stanu 2 są przejścia do stanów  $\{2, 3\}$ , a ze stanu

3 są też przejścia do stanów  $\{2, 3\}$ ). A więc przy  $n$  literach  $a$ , automat stworzy  $2^n$  biegów, które kończą się w stanie 4. Z kolei w niebieskiej części, automat robi tylko jeden krok niedeterministycznie: raz może wybrać pomiędzy stanem 6 a 7. A więc przy  $m$  literach  $b$ , automat stworzy  $m + 1$  biegów, które kończą się w stanach  $\{6, 7\}$ . Na końcu słowa z  $a^+b^+$ , każdy bieg skończy w jednym ze stanów  $\{4, 6, 7\}$ . Aby automat zaakceptował, musi być więcej biegów w stanach  $\{6, 7\}$  niż biegów w stanie 4.

Język  $L$  nie może być bezkontekstowy, bo i wtedy przecięcie  $L$  z regularnym językiem byłoby bezkontekstowe, a wiadomo, że  $\{a^n b^m : 2^n \geq m + 1\}$  nie jest bezkontekstowy.

14. Czy każdy język generowany przez wyrażenie regularne jest w klasie PTIME?

**Odpowiedź.** Tak. Skoro wyrażenie regularne można przerobić na automat deterministyczny, to języki te można rozpoznawać nawet w czasie liniowym.

15. Załóżmy  $P \neq NP$ , bo inaczej zadanie jest oczywiste. Czy istnieje NP-trudny język rozpoznawany przez niedeterministyczny automat ze stosom?

**Odpowiedź.** Nie. Algorytm CYK, który był na wykładzie, pokazuje, że każdy język bezkontekstowym można rozpoznawać w czasie wielomianowym, konkretnie  $O(n^3)$ .