Złożoność obliczeniowa, homework 3

Jan Kwiatkowski

12.01.2024

Problem 1

Consider the length-preserving function which inputs a bitstring, and replaces all bits by 0, except the leftmost 1. Here are some examples:

$$0101101 \mapsto 0100000 \qquad 0000 \mapsto 0000 \qquad 0001101 \mapsto 0001000$$

Show that this function can be implemented by a circuit of constant depth and linear size (linear number of wires, not just gates).

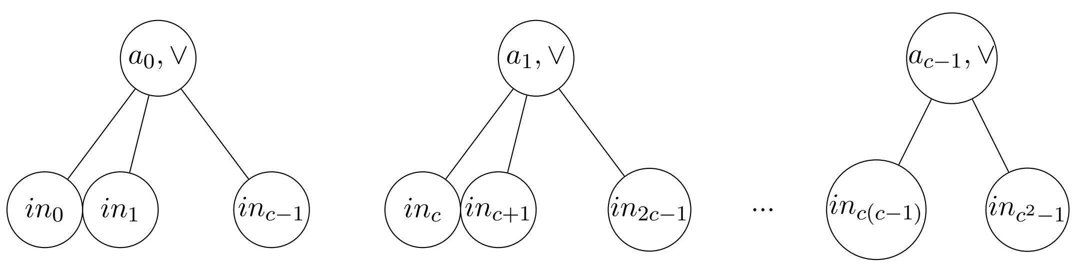
Solution

Niech n - długość inputu, in_i - i-ty bit inputu, out_i - i-ty bit output. Rozważmy na początku łatwiejszą wersję zadania, w której nie mamy ograniczenia na liniową wielkość. Moglibyśmy wtedy stworzyć następujący obwód:

$$out_i = in_i \wedge \forall_{j<i} \neg in_j$$

Innymi słowy dla danej pozycji i output to 1 jeśli dla każdego $j < i$ input to 0 oraz $in_i = 1$. Oczywiście taki obwód rozwiązuje problem z treści zadania. Dodatkowo ma on głębokość równą 2 (dla każdego bitu potrzebujemy go zanegować, następnie na zanegowanych bitach tworzymy koniunkcje). Niestety to rozwiązanie używa $O(n^2)$ pamięci. Niemniej nazwijmy je ALG (wykorzystamy je, ale w zmodyfikowanej wersji).

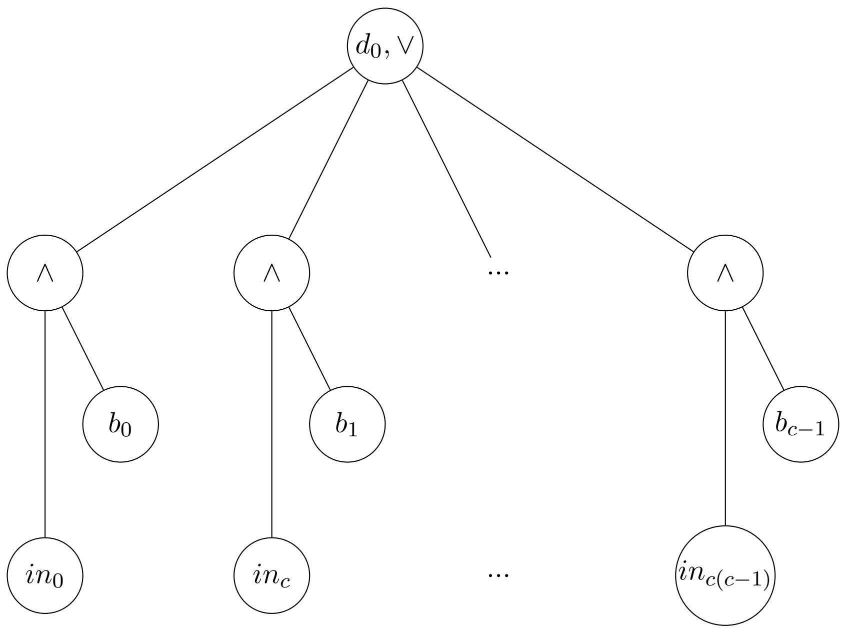
Niech $c = \sqrt{n}$. Podzielmy input na bloki o długości c (takich bloków będzie c). Dla każdego bloku i stwórzmy alternatywę a_i wszystkich jego bitów:



Teraz możemy zastosować ALG, ale na wierzchołkach $\{a_0, a_1, ..., a_{c-1}\}$ używając $O((\sqrt{n})^2) = O(n)$ krawędzi. W ten sposób znajdujemy blok, w którym znajduje się pierwsza jedynka. Mamy zatem c wierzchołków $b_0, b_1, ..., b_{c-1}$, w których dokładnie jeden z nich (b_s) zawiera jedynkę, a pozostałe zawierają zera.

Teraz będziemy chcieli przepisać wszystkie wierzchołki z inputu należące do bloku b_s a następnie ponownie zastosować ALG. W jaki sposób przepisać blok b_s? Stworzymy c nowych wierzchołków $d_0, ..., d_{c-1}$ oraz dodamy krawędzie

$$d_i = \bigvee_{j \mid j=ck+i,\, k\in\mathbb{Z}_{\geqslant 0}} \Big(b_k \wedge in_j\Big)$$



Opisując to słownie dla każdej reszty z dzielenia przez c tworzymy alternatywę po koniunkcjach $in_j \wedge b_k$, gdzie indeks j należy do bloku k. W ten sposób przepisaliśmy blok b_s do wierzchołków $d_0, ..., d_{c-1}$ używając $O(n)$ krawędzi.

Teraz możemy ponownie zastosować ALG tym razem na $d_0, ..., d_{c-1}$ znów używając $O(n)$ krawędzi. W ten sposób otrzymujemy wierzchołki $e_0, ..., e_{c-1}$. Pozostaje odpowiednio przedstawić output. Robimy to w następujący sposób

$$out_{ck+i} = b_k \wedge e_i \qquad k, i \in \mathbb{Z}_{\geqslant 0}$$

Innymi słowami output na i-tym bicie to 1 wtw. i należy do bloku b_s oraz jest pierwszą jedynką w tym bloku.

Stworzyliśmy zatem obwód znajdujący odpowiedź na zadany problem o liniowym rozmiarze oraz stałej głębokości równej 7. W powyższym rozwiązaniu założyłem, że $n = c^2$ dla pewnego całkowitego c. Jeśli tak nie jest to rozwiązanie wciąż działa, z tym że ostatni blok ma mniejszy rozmiar.

$\blacksquare$

Problem 2

Define *randomized* AC^1 to be the following circuit model. In the the n-th circuit, there are n inputs that represent the input bits, and also some other inputs that represent random bits. The circuits need to have the AC^1 constraints: the n-th circuit has size $poly(n)$ and depth $O(logn)$. We require that for every n and every input $w \in 2^n$, there is probability at least 0.6 of getting to choose the random inputs to get the correct answer.
Prove that this model defines the same languages as classical AC^1, without randomness.

Solution

Należy dowieść inkluzji w obie strony. Jedna z nich jest oczywista. Mając dany obwód $\mathcal{A}$ klasy AC^1 tworzymy na jego podstawie identyczny obwód klasy randomized AC^1, który dodatkowo ignoruje losowe bity, do których ma dostęp.

Teraz załóżmy, że mamy dany obwód $\mathcal{B}$ o n bitach wejściowych, klasy randomized AC^1 i na jego podstawie chcemy stworzyć obwód $\mathcal{A}$ klasy AC^1. Korzystając z faktu poznanego na ćwiczeniach zauważamy, że jesteśmy w stanie wybrać wielomianowo wiele ($N := p(n)$) zestawów bitów losowych tak, aby przeprowadzenie na nich testu większościowego zdeterminowało naszą odpowiedź. Jak skonstruować $\mathcal{A}$? Dla każdego ze zhardkodowanych zestawów losowych bitów tworzymy kopię $\mathcal{B}$, która ma do niego dostęp. W ten sposób otrzymujemy $p(n)$ odpowiedzi obwodu $\mathcal{B}$ i musimy na nich przeprowadzić test większościowy. Ponownie korzystamy z faktu z ćwiczeń - dodawanie dwóch liczb binarnych jest w klasie AC^0. Tworzymy więc strukturę drzewa binarnego, która będzie zliczać liczbę jedynek w poddrzewie (i zapisywać ją jako liczbę binarną). Opisane drzewo ma głębokość $\log N$, a każdy wierzchołek zapisuje swój wynik na $\log N$ bitach. W ten sposób w korzeniu znajdzie się liczba binarna opisująca ile spośród N wywołań obwodu $\mathcal{B}$ zakończyło się sukcesem.
Teraz musimy sprawdzić czy w korzeniu znajduje się liczba większa niż $N/2$. Zauważmy, że zapis binarny $N/2$ to zapis binarny N bez ostatniego bitu. Mamy zatem dwie liczby i chcemy je porównać. Możemy to zrobić w prosty sposób. Załóżmy, że porównujemy liczby a, b długości m bitów, oraz że x_i to i-ty bit liczby x (od najbardziej znaczącego). Zachodzi

$$a > b \qquad \Leftrightarrow \qquad \bigvee_{0 \leqslant i < m} \left(a_i = 1 \land b_i = 0 \land \bigwedge_{j<i} \neg(a_j = 0 \land b_j = 1) \right)$$

Oczywiście prawą stronę równoważności możemy bardzo łatwo zapisać jako obwód klasy AC^0. W ten sposób obwód $\mathcal{A}$ zwraca poszukiwaną wartość.

Przy konstrukcji $\mathcal{A}$ stworzyliśmy N (wielomianowo wiele) kopii obwodu $\mathcal{B}$ klasy randomized AC^1. Następnie stworzyliśmy kolejne $O(N \log N)$ wierzchołków oraz zwiększyliśmy głębokość $\mathcal{B}$ o $O(\log(N))$. Ponieważ $N = p(n)$, to z podstawowych własności wielomianów wynika, że obwód $\mathcal{A}$ jest klasy AC^1.

■

Problem 3

A triangle in an undirected graph is a clique of size three. Two triangles are called non-neighbouring if they share no vertices, and furthermore there is no edge that connects vertices from the two triangles. Let $k, d \in \{1, 2, ...\}$. Give an randomized algorithm which does the following:

- Input. An undirected graph G where every vertex has at most d neighbours.
- Question. Are there k triangles that are pairwise non-neighbouring?

The algorithm should give a correct answer with probability at least 0.6 for every input, and it should run in time

$$f(k, d) \cdot p(\text{number of vertices})$$

where f is some computable function and p is a polynomial whose degree does not depend on k. Hint: consider a graph that is obtained by removing every edge with probablity 0.5.

Solution

Naszym celem będzie stworzenie algorytmu RP. Niech

- G' - graf powstały z G poprzez pokolorowanie każdego wierzchołka na kolor biały lub czarny (kolory wybierane losowo, niezależnie).
- G'' - graf powstały poprzez usunięcie z G' każdej krawędzi (niezależnie) z prawdopobieństwem $\frac{1}{2}$.

Powiemy, że trójkąt $T = (v_1, v_2, v_3)$ jest *izolowany* w grafie H jeśli żaden z trzech wierzchołków, które go tworzą nie posiada sąsiadów poza tym trójkątem. Innymi słowami taki trójkąt jest izolowany wtw. $(v_i, y) \in E(H) \iff y \in T$.

Nasz algorytm RP jest prosty - zliczamy izolowane czarne trójkąty w G'' posiadające tylko białych sąsiadów w G'. Jeśli takich trójkątów jest co najmniej k to odpowiadamy "TAK", w przeciwnym wypadku odpowiadamy "NIE". Zauważmy, że jeżeli odpowiadamy "TAK", to rzeczywiście w G istnieje k niesąsiadujących trójkątów - to gwarantuje nam kolorowanie wierzchołków, czarne trójkąty mają tylko białych sąsiadów, więc dwa różne trójkąty nie mogą ze sobą sąsiadować.
Jeśli poprawna odpowiedź to "NIE", nasz algorytm na pewno (100%) odpowie "NIE". Pozostaje analiza algorytmu jeśli poprawna odpowiedź to "TAK".

Oszacujmy prawdopodobieństwo znalezienia przez nasz algorytm k szukanych trójkątów. Korzystając z reguły sumy zauważamy, że pesymistyczny przypadek jest wtedy, gdy w G znajduje się dokładnie k niesąsiadujących trójkątów.

$$p(\text{found}) \geqslant \left(\left(\frac{1}{2}\right)^d\right)^{3k} \cdot \left(\frac{1}{2}\right)^{3k} \cdot \left(\frac{1}{2}\right)^{3dk} = \left(\frac{1}{2}\right)^{6dk+3k} \geqslant \left(\frac{1}{2}\right)^{9dk}$$

Pierwszy czynnik iloczynu odpowiada prawdopodobieństwu, że dla każdego z k trójkątów dobrze usuwamy / zostawiamy w G'' krawędzie tworzących go wierzchołków. Drugi czynnik iloczynu opisuje pokolorowanie wierzchołków trójkątów na czarno. Ostatni czynnik iloczynu opisuje pokolorowanie wszystkich sąsiadów trójkątów na biało.

Podobnie jak na ćwiczeniach powtarzamy takie losowanie 2^{9dk} razy i jeśli chociaż raz algorytm znajdzie rozwiązanie, zwracamy "TAK". W ten sposób prawdopodobieństwo znalezienia odpowiedzi to

$$1 - \left(1 - \left(\frac{1}{2}\right)^{9dk}\right)^{2^{9dk}} \approx 1 - \frac{1}{e} > 0.6$$

Zatem nasz randomizowany algorytm zwraca poprawną odpowiedź z prawdopodobieństwem co najmniej 0.6 dla każdego inputu. Dodatowo zliczanie izolowanych czarnych trójkątów z białymi sąsiadami łatwo zrealizować w czasie $O(n+m)$ (a więc również w $O(n^2)$), gdzie $n = |V(G)|, m = |E(G)|$. Zatem ostatecznie algorytm działa w czasie

$$O(2^{9dk} \cdot n^2) = f(k,d) \cdot poly(n)$$

■

Problem 4

Show that the following problem is fixed parameter tractable (FPT).

- Input. An undirected graph G.
- Parameter $k \in \{1, 2, ...\}$.
- Question. Can one remove k vertices so that in the resulting graph, every connected component has size at most 10.

Solution

Załóżmy, że w G istnieje wierzchołek v o stopniu większym lub równym $k + 10$. Taki wierzchołek musi zostać usunięty - w przeciwnym razie możemy usunąć co najwyżej k z jego sąsiadów, a więc pozostanie on w spójnej o $\geqslant 11$ wierzchołkach. Oczywiście jeśli musielibyśmy usunąć więcej niż k wierzchołków, to możemy odpowiedzieć "NIE".

Teraz mamy do czynienia z instancją problemu, w której stopień każdego z wierzchołków jest ograniczony przez $k + 10$. Zauważmy, że w G może istnieć maksymalnie k spójnych składowych o rozmiarze większym niż 10. W przeciwnym wypadku z zasady szufladkowej Dirichleta wynika, że jedna z nich nie miałaby żadnych usuniętych wierzchołków (i możemy od razu zwrócić "NIE"). Nie chcemy również usuwać wierzchołków ze spójnych składowych o rozmiarze $\leqslant 10$.

Zatem pozostaje sytuacja, w której każdy wierzchołek ma maksymalnie $k+10$ sąsiadów, a graf składa się z maksymalnie k spójnych składowych. Spróbujmy oszacować rozmiar takiej spójnej S. Zauważmy, że usunięcie wierzchołka v powoduje powstanie nie więcej niż $\deg(v)$ nowych spójnych. W naszym przypadku $\deg(v) \leqslant k + 10$. Załóżmy, że S składała się z więcej niż $k + 10k \cdot (k + 10)$ wierzchołków. Usunęliśmy k wierzchołków, a powstały graf ma maksymalnie $k \cdot (k + 10)$ spójnych składowych. Z zasady szufladkowej Dirichleta istnieje spójna o rozmiarze większym niż 10. Z tego wynika, że jeżeli istnieje spójna składowa o rozmiarze większym niż $k + 10k \cdot (k + 10)$, to możemy od razu zwrócić "NIE".

Pozostaje sytuacja, w której mamy niewiele niewielkich spójnych. W tym wypadku używamy brutalnego algorytmu, który sprawdza każdy możliwy podzbiór wierzchołków do usunięcia, a następnie sprawdza czy powstały graf ma spójne składowe o rozmiarze $\leqslant 10$.

Mogła również wystąpić sytuacja, w której nasz budżet (k) był za duży i w trakcie działania algorytmu nie wykorzystaliśmy go do końca (np. gdy G nie zawiera żadnych krawędzi). Możemy wtedy usunąć dowolne wierzchołki, aby wyzerować budżet (chyba, że $k > n$, wtedy oczywiście odpowiedź to "NIE").

Niech $n = |V(G)|$. W czasie n^2 możemy łatwo sprowadzić wejściową instancję problemu do równoważnej instancji z niewielką liczbą niewielkich spójnych. Zatem sumaryczny czas działania algorytmu to

$$O\left(n^2 + 2^{k\cdot(k+10k\cdot(k+10))} \cdot k \cdot (k + 10k \cdot (k + 10))\right) = O(n^2 + 2^{k^3} \cdot k^3)$$

A więc zadany problem jest FPT.

■

Problem 5

Consider the following problem:

- Input. A number $t \in \{0, 1, ...\}$ and two size n subsets of $\{0, 1\}^d$.
- Question. Can one find a vector in the first subset that is at Hamming distance at most t from some vector in the second subset?

Assuming the orthogonal vectors conjecture, show that there is no $\varepsilon > 0$ such that this problem can be solved in time

$$poly(d, t) \cdot n^{2-\varepsilon}.$$

Solution

Zacznijmy od wprowadzenia kilku oznaczeń. Dla $h \in \mathbb{Z}_+$, $x, y \in \{0, 1\}^h$ niech:

- $\mathrm{Ham}(x, y)$ - odległość Hamminga między x a y
- 11_{xy} - liczba takich indeksów $i \in \{0, 1, ..., d-1\}$, że $x_i = 1$ oraz $y_i = 1$
- $\widetilde{x}$ to wektor powstały przez odwrócenie wszystkich bitów x (czyli $\widetilde{x}_i = 1 - x_i$)
- 1_x to liczba jedynek w wektorze
- $\mathbb{0}^h$ to wektor złożony z samych zer

Naszym ogólnym celem jest zapisać problem *orthogonal vectors* (OV) w terminach problemu z treści zadania (HAMV). Zauważmy, że dwa wektory x, y są prostopadłe wtw. $11_{xy} = 0$, a więc to wartość 11_{xy} będziemy chcieli przedstawić w terminach odległości Hamminga. Przedstawię kilka prostych obserwacji dla $x, y \in \{0, 1\}^d$:

1. $\widetilde{x}$ powstaje przez odwrócenie wszystkich bitów x, więc $\mathrm{Ham}(\widetilde{x}, y) = d - \mathrm{Ham}(x, y)$
2. $\mathrm{Ham}(x, \mathbb{0}^d) = \mathrm{Ham}(\mathbb{0}^d, x) = 1_x$
3. $11_{xy} = \Big(1_x + 1_y - \mathrm{Ham}(x, y)\Big) \cdot \frac{1}{2}$ - liczba indeksów, na których zarówno x jak i y mają jedynki to łączna liczba jedynek w x, y minus odległość Hamminga między x a y (przemnożona przez $\frac{1}{2}$, aby policzyć indeksy, a nie łączną liczbę jedynek)

Teraz możemy obliczyć 11_{xy}:

$$\begin{aligned}
& 11_{xy} = \Big(1_x + 1_y - \mathrm{Ham}(x, y)\Big) \cdot \frac{1}{2} \\
\Leftrightarrow \quad & 2 \cdot 11_{xy} = \mathrm{Ham}(x, \mathbb{0}^d) + \mathrm{Ham}(\mathbb{0}^d, y) - (d - \mathrm{Ham}(\widetilde{x}, y)) \\
\Leftrightarrow \quad & 2 \cdot 11_{xy} = \mathrm{Ham}(x, \mathbb{0}^d) + \mathrm{Ham}(\mathbb{0}^d, y) + \mathrm{Ham}(\widetilde{x}, y) - d \\
\Leftrightarrow \quad & 2 \cdot 11_{xy} + d = \mathrm{Ham}(x, \mathbb{0}^d) + \mathrm{Ham}(\mathbb{0}^d, y) + \mathrm{Ham}(\widetilde{x}, y)
\end{aligned}$$

Możemy już udowodnić tezę zadania. Niech zbiory (X, Y), n oraz d stanowią instancję problemu OV. Niech $\cdot$ oznacza konkatenację wektorów. Zauważmy, że

$$\mathrm{Ham}(x, \mathbb{0}^d) + \mathrm{Ham}(\mathbb{0}^d, y) + \mathrm{Ham}(\widetilde{x}, y) = \mathrm{Ham}(x \cdot \mathbb{0}^d \cdot \widetilde{x},\ \mathbb{0}^d \cdot y \cdot y)$$

Zatem dokładnie w taki sposób modyfikujemy X oraz, Y:

- $X' := \forall_{x \in X}\ x \mapsto x \cdot \mathbb{0}^d \cdot \widetilde{x}$
- $Y' := \forall_{y \in Y}\ y \mapsto \mathbb{0}^d \cdot y \cdot y$

Załóżmy, że udało nam się rozwiązać problem HAMV dla zbiorów X', Y' oraz $t := d$, a rozwiązaniem jest para wektorów (x', y'). To oznacza, że

$$
\begin{aligned}
& 2 \cdot 11_{x'y'} + d \leqslant t \\
\Leftrightarrow \quad & 2 \cdot 11_{x'y'} \leqslant 0 \\
\Leftrightarrow \quad & 11_{x'y'} = 0
\end{aligned}
$$

Zatem odpowiadająca parze (x', y') para wektorów w OV jest rozwiązaniem OV. Analogicznie pokazujemy, że jeśli w algorytmie dla HAMV szukana para nie istnieje, to taka para nie istnieje również dla oryginalnej instancji problemu OV.

Zatem potrafiąc rozwiązać HAMV w czasie

$$poly(3d) \cdot n^{2-\varepsilon} = poly'(d) \cdot n^{2-\varepsilon}$$

potrafilibyśmy rozwiązać OV w takim samym czasie. Otrzymaliśmy zatem sprzeczność z *orthogonal vector conjecture.*

■
