ZLOŻONOŚĆ OBLICZENIOWA - WYK. 15

1. PCP, czyli Probabilistically Checkable Proofs

   - Idea: skoro **IP** to taka duża klasa, to może jakoś osłabmy protokoły interakcyjne i zobaczymy co wyjdzie. Na przykład, niech P nie może dostosowywać odpowiedzi na podstawie wcześniejszych pytań.
   - W dowodzie z poprzednich zajęć, V pytał P o wielomiany jednej zmiennej postaci

     $$h(x) = \sum_{v_2 \in \{0,1\}} \cdots \sum_{v_n \in \{0,1\}} g(a_1, \ldots, a_i, x, v_{i+2}, \ldots, v_n)$$

     dla $i = 0..n$ i różnych $a_i < p < 2^n$. Wszystkie te pytania są podobnej postaci, a jest ich z grubsza 2^{n^2}. P może obliczyć odpowiedzi na wszystkie możliwe pytania i przesłać je jako jeden wielki "dowód". V potem sprawdza tylko n miejsc w tym dowodzie.
   - $\mathbf{PCP}(r(n), q(n))$-verifier dla języka L to maszyna V, probabilistyczna, działająca w czasie wielomianowym, która:
     - na słowie w o długości n, z dostępem do słowa π (*dowodu* lub *świadka*), używa $r(n)$ bitów losowych i sprawdza $q(n)$ pozycji w słowie π (zakładamy że sprawdzenie zajmuje stały czas)
     - sprawdzane pozycje nie zależą od słowa π, tylko od w i od bitów losowych (V nie jest *adaptacyjny*),
     - dla $w \in L$ istnieje π takie że V akceptuje zawsze,
     - dla $w \notin L$, dla każdego π V akceptuje z prawdopodobieństwem najwyżej $\frac{1}{2}$.
   - *Fakt:* Możemy założyć, że dowód przedstawiany $\mathbf{PCP}(r(n), q(n))$-verifierowi ma długość najwyżej $q(n)2^{r(n)}$, bo tylko tyle różnych pozycji V jest w stanie sprawdzić. Na przykład, jeśli $r(n) = \mathcal{O}(\log n)$, to możemy się ograniczyć do dowodów wielomianowej długości.
   - Język L jest w klasie $\mathbf{PCP}(r(n), q(n))$ jeśli istnieją stałe c i d takie, że dla L istnieje $\mathbf{PCP}(c \cdot r(n), d \cdot q(n))$-verifier.
   - Ćwiczenie: liczbę $\frac{1}{2}$ można zastąpić dowolną liczbą z przedziału $(0, 1)$.
   - Przypadki trywialne: $\mathbf{PCP}(poly(n), 0) = \mathbf{BPP}$, $\mathbf{PCP}(0, poly(n)) = \mathbf{NP}$.

     Ćwiczenie: $\mathbf{PCP}(\log n, poly(n)) = \mathbf{NP}$.

     Ćwiczenie: Przeanalizować związek między $\mathbf{PCP}(poly(n), poly(n))$ a **IP**. Różnica jest taka, że w **PCP** prover nie widzi wcześniejszych pytań. Która inkluzja jest łatwa?

2. Twierdzenie PCP (Arora, Lund, Motwani, Safra, Sudan, Szegedy' 92): $\mathbf{PCP}(\log n, 1) = \mathbf{NP}$.

Zauważmy że mniej niż $\log n$ bitów już się nie da losować, bo wtedy nie można by zajrzeć w każde miejsce dowodu wielomianowej długości.

To znaczy, że dla każdego problemu w **NP** istnieje taki verifier, który

- dla słów wejściowych oczekuje dowodów wielomianowego rozmiaru,
- losując $\log n$ bitów sprawdza pewną stałą liczbę bitów w dowodzie,
- na tej podstawie na pewno akceptuje wszystkie poprawne słowa, a z dużym prawdopodobieństwem odrzuca niepoprawne.

To jest dziwne twierdzenie. Weźmy na przykład 3-kolorowalność grafu, gdzie dowód to kolorowanie. Jeśli kolorowanie jest błędne w jednym miejscu, to ciężko to miejsce znaleźć. Twierdzenie PCP mówi, że kolorowanie można zapisać w taki sposób, że każdy błąd jest w nim widoczny w wielu miejscach, przy czym za błąd uznajemy także to, że nie jest zapisane w taki sposób.

Weźmy też taki problem: czy dane twierdzenie ϕ ma dowód długości $\leq n$, gdzie n jest podane unarnie? Normalnie, aby sprawdzić podany klasycznie dowód, trzeba przejrzeć go w całości, a błąd w każdym miejscu dyskwalifikuje go. Z twierdzenia PCP wynika, że istnieje taki format zapisu dowodów, w którym

- każdy błąd można z dużym prawdopodobieństwem wykryć strzelając na ślepo,
- z dużym prawdopodobieństwem można odrzucić dowody nie zapisane w tym formacie.

3. Udowodnimy łatwiejszą wersję twierdzenia PCP: $\mathbf{PCP}(poly(n), 1) = \mathbf{NP}$. Inkluzję z lewej w prawą już pokazaliśmy, została inkluzja z prawej w lewą. Pokażemy verifiera PCP dla pewnego problemu **NP**-zupełnego. Zauważmy, że ta słabsza wersja cały czas jest zaskakująca!

4. Idea jest taka: zamiast normalnego świadka dla naszego problemu **NP**-zupełnego, przedstawimy tego świadka zakodowanego w pewien (bardzo rozwlekły) sposób, z użyciem kodów korekcyjnych.

   Kod Walsha-Hadamarda to kodowanie ciągu binarnego $\mathbf{v} \in \{0,1\}^n$ jako funkcji liniowej z $\{0,1\}^n$ w $\{0,1\}$ zdefiniowanej tak:

$$\mathbf{x} \mapsto \mathbf{x} \odot \mathbf{v} = \sum_{i=1}^{n} x_i v_i \pmod 2$$

   Taką funkcję można zapisać jako ciąg długości 2^n.

   Podstawowa własność tego kodowania jest taka, że jeśli $\mathbf{x} \neq \mathbf{y}$ to kodowania $\mathbf{x}$ i $\mathbf{y}$ różnią się od siebie na dokładnie połowie pozycji.

5. Pierwsze zadanie jest takie: przetestować, czy dana funkcja $f : \{0,1\}^n \to \{0,1\}$ (przedstawiona jako ciąg długości 2^n) jest kodem W-H jakiegoś ciągu długości n. Kody W-H to dokładnie funkcje *liniowe*, tj. takie że

$$f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$

dla wszystkich wektorów $\mathbf{x}, \mathbf{y}$. Istotnie, dla funkcji liniowej f mamy:

$$f(\mathbf{x}) = f(\sum_{i=1}^{n} x_i \cdot \mathbf{b_i}) = \sum_{i=1}^{n} x_i \cdot f(\mathbf{b_i}) = \mathbf{x} \odot (f(\mathbf{b_1}), \ldots, f(\mathbf{b_n}))$$

gdzie $\mathbf{b_i}$ to i-ty jednostkowy wektor bazowy. Pytanie więc brzmi: jak sprawdzić, czy funkcja jest liniowa? Tak naprawdę będziemy mniej ambitni i spytamy: jak sprawdzić, czy dana funkcja niewiele się różni od liniowej?

Dla liczby $\rho \in (0,1)$, powiemy że funkcja $f : \{0,1\}^n \to \{0,1\}$ jest *ρ-bliska liniowej* jeżeli istnieje taka liniowa funkcja g że

$$\Pr_{\mathbf{x}\in\{0,1\}^n}[f(\mathbf{x}) = g(\mathbf{x})] \geq \rho$$

Pomoże nam takie twierdzenie o teście liniowości:

Twierdzenie [Blum-Luby-Rubinfeld'90]: Każda funkcja $f : \{0,1\}^n \to \{0,1\}$, która spełnia własność

$$\Pr_{\mathbf{x},\mathbf{y}\in\{0,1\}^n}[f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})] \geq \rho$$

dla pewnego $\rho > \frac{1}{2}$, jest ρ-bliska funkcji liniowej.

Dowód pominiemy (korzysta z analizy Fourierowskiej).

Dzięki testowi liniowości możemy odrzucić, z prawdopodobieństwem co najmniej $\frac{1}{2}$, funkcje które nie są ρ-bliskie liniowej, sprawdzając liniowość $\frac{1}{1-\rho}$ razy na losowych argumentach, dla ρ dowolnie bliskiego 1.

6. Drugie zadanie: załóżmy, że dla $\delta < \frac{1}{4}$ funkcja $f : \{0,1\}^n \to \{0,1\}$ jest $(1-\delta)$-bliska liniowej funkcji $\bar{f}$. Zauważmy, że funkcja $\bar{f}$ jest wtedy wyznaczona jednoznacznie, bo dwie funkcje liniowe różnią się na połowie argumentów. Załóżmy, że mamy argument $\mathbf{x} \in \{0,1\}$ i chcemy obliczyć $\bar{f}(\mathbf{x})$ mając dostęp do f. Chcemy, żeby to się udało dla każdego $\mathbf{x}$ z dużym prawdopodobieństwem, nawet jeśli akurat pechowo f różni się od $\bar{f}$ na $\mathbf{x}$.

   Robimy tak:

   - losujemy $\mathbf{y} \in \{0,1\}^n$,
   - zwracamy $f(\mathbf{y}) + f(\mathbf{x}+\mathbf{y})$.

   Ponieważ $\mathbf{y}$ i $\mathbf{x}+\mathbf{y}$ są losowe z jednostajnym rozkładem, to prawdopodobieństwo że f zgadza się z $\bar{f}$ na obu tych argumentach jest co najmniej $1-2\delta$. A jeśli tak jest, to w wyniku dostajemy

$$\bar{f}(\mathbf{y}) + \bar{f}(\mathbf{x}+\mathbf{y}) = \bar{f}(\mathbf{x}).$$

7. Pokażemy $(poly(n), 1)$-verifiera dla następującego problemu **NP**-zupełnego: czy dany układ równań kwadratowych taki jak:

$$\begin{aligned} x_1x_2 + x_3x_4 + x_2x_5 &= 1 \\ x_2x_3 + x_4x_5 &= 0 \\ x_1x_3 + x_3x_5 + x_3x_4 &= 1 \end{aligned}$$

ma rozwiązanie modulo 2? **NP**-zupełność tego problemu była zadaniem na kolokwium.

Układ m równań nad n zmiennymi można przedstawić jako macierz A o rozmiarach $m \times n^2$ oraz wektor $\mathbf{b}$ długości m. Pytamy wtedy o istnienie takiego wektora $\mathbf{v}$ długości n, że

$$A(\mathbf{v} \otimes \mathbf{v}) = \mathbf{b}$$

gdzie $\otimes$ to produkt tensorowy, czyli $\mathbf{v} \otimes \mathbf{v}$ to wektor długości n^2.

Verifier V, dla danego układu równań, oczekuje dowodu długości $2^n + 2^{n^2}$, który koduje funkcje $f : \{0,1\}^n \to \{0,1\}$ i $g : \{0,1\}^{n^2} \to \{0,1\}$. W poprawnym dowodzie te funkcje to kody W-H wartościowania spełniającego i jego kwadratu tensorowego.

Weryfikator sprawdza po pierwsze czy funkcje f i g są 0.999-bliskie liniowej. Jak to już sprawdzi to zakładamy że mamy dostęp do funkcji liniowych $\bar{f}$ i $\bar{g}$; dalszy ciąg weryfikacji zada tylko stałą liczbę pytań, więc z dużym prawdopodobieństwem nie trafimy w niewłaściwe wartości f i g.

Po drugie, weryfikator sprawdza, czy g koduje $\mathbf{v}\otimes\mathbf{v}$, gdzie $\mathbf{v}$ jest wektorem kodowanym przez f. To robimy następująco:

- wybierz $\mathbf{x}, \mathbf{x}' \in \{0,1\}^n$ losowo,
- odrzuć jeśli $g(\mathbf{x} \otimes \mathbf{x}') \neq f(\mathbf{x})f(\mathbf{x}')$,
- powtórz 10 razy.

Jeśli kodowanie jest poprawne, to

$$f(\mathbf{x})f(\mathbf{x}') = \left(\sum_{i=1}^{n} x_i v_i\right)\left(\sum_{j=1}^{n} x'_i v_i\right) = \sum_{i,j=1}^{n} x_i x'_j v_i v_j = (\mathbf{x}\otimes\mathbf{x}')\odot(\mathbf{v}\otimes\mathbf{v}) = g(\mathbf{x}\otimes\mathbf{x}')$$

Czyli V nigdy nie odrzuci poprawnego dowodu. Z kolei jeśli dowód jest niepoprawny, to każdy test odrzuci z prawdopodobieństwem co najmniej $\frac{1}{4}$. Istotnie, załóżmy że g koduje pewne słowo $\mathbf{w}$ i przedstawmy je jako macierz W o rozmiarach $n \times n$. Słowo $\mathbf{v} \otimes \mathbf{v}$ też można przedstawić jako taką macierz V, gdzie $V_{ij} = v_i v_j$. Teraz:

$$g(\mathbf{x} \otimes \mathbf{x}') = \mathbf{w} \odot (\mathbf{x} \otimes \mathbf{x}') = \sum_{i,j=1}^{n} w_{ij} x_i x'_j = \mathbf{x} W \mathbf{x}'$$

$$f(\mathbf{x})f(\mathbf{x}') = (\mathbf{v}\odot\mathbf{x})(\mathbf{v}\odot\mathbf{x}') = \left(\sum_{i=1}^{n} v_i x_i\right)\left(\sum_{j=1}^{n} v_j x'_j\right) = \sum_{i,j=1}^{n} v_i v_j x_i x'_j = \mathbf{x}V\mathbf{x}'$$

Nasz test odrzuca jeśli $\mathbf{x}W\mathbf{x}' \neq \mathbf{x}V\mathbf{x}'$. Jeżeli $W \neq V$, to dla co najmniej połowy wektorów $\mathbf{x}$ mamy $\mathbf{x}W \neq \mathbf{x}V$. Z kolei wtedy dla co najmniej połowy wektorów $\mathbf{x}'$ mamy $\mathbf{x}W\mathbf{x}' \neq \mathbf{x}V\mathbf{x}'$, czyli odrzucimy z prawdopodobieństwem co najmniej $\frac{1}{4}$.

Skoro już wiemy że g i f kodują coś zgodnego, to możemy łatwo sprawdzić, czy dane równanie jest spełnione. Dla k-tego równania chcemy sprawdzić, czy

$$\sum_{i,j} A_{k,ij} v_i v_j = b_k$$

gdzie macierz A_k rozmiaru $n \times n$ koduje lewą stronę równania. Jeśli tę macierz potraktujemy jako wektor $\mathbf{a}$, to lewa strona to po prostu $\mathbf{a}\odot(\mathbf{v}\otimes\mathbf{v})$, czyli $g(\mathbf{a})$. Wystarczy więc obliczyć $\mathbf{a}$, sprawdzić $g(\mathbf{a})$ i porównać z b_k.

Tu jest jeszcze taki problem, że nie możemy sprawdzić każdego równania, bo możemy sprawdzić dowód tylko w stałej liczbie miejsc. Tak więc wybieramy losowo podzbiór równań i sumujemy je modulo 2, a potem sprawdzamy czy takie równanie jest spełnione. Jeżeli wartościowanie nie jest spełniające, to suma losowo wybranego podzbioru jest niespełniona z prawdopodobieństwem $\frac{1}{2}$.

Koniec dowodu.