ZLOŻONOŚĆ OBLICZENIOWA - WYK. 12

1. *Fixed-parameter tractabillity.* Idea: czasem istotą trudności nie jest długość wejścia, ale jakiś jego parametr. Ustalając parametr, czasem możemy uzyskać algorytm wielomianowy. Czasem nawet wykładnik w tym wielomianie nie zależy od parametru. Takie problemy nazywamy FPT.

   Formalnie: *parametr* to funkcja ze słów wejściowych w liczby naturalne. Problem jest *fixed parameter tractable* względem parametru k jeśli ma złożoność $f(k) \cdot n^c$ (uwaga: k nie może być w wykładniku).

2. SAT jest FPT względem liczby zmiennych k. *Algorytm:* sprawdź wszystkie wartościowania; złożoność $2^k n$.

3. VERTEX-COVER jest FPT względem rozmiaru maksymalnego pokrycia.

   Algorytm: Dla każdej krawędzi decydujemy który wierzchołek wziąć do pokrycia i robimy przy tym pełny backtracking. Ten backtracking ma głębokość k, więc złożoność wychodzi $2^k n$.

   Uwaga: Pomysł ze sprawdzeniem każdego podzbioru k-elementowego nie działa, bo k wpada w wykładnik.

4. k-kolorowanie grafu nie jest FPT względem liczby kolorów (chyba że $\mathbf{P} = \mathbf{NP}$). Gdyby było, to 3-kolorowanie dałoby się rozwiązać w czasie $f(k) \cdot n^c$, czyli wielomianowym.

   Ale jest parametr, względem którego k-kolorowanie grafu jest FPT. Jest to *szerokość drzewiasta* (treewidth). Oto definicja: *dekompozycją drzewiastą* grafu (V, E) nazywamy drzewo T, w których wierzchołki są podzbiorami V, takie że:

   - dla każdego $v \in V$, wierzchołki z T do których należy v tworzą spójne (i niepuste) poddrzewo,
   - dla każdej krawędzi $(u, v) \in E$ istnieje wierzchołek X w T taki że $u, v \in X$.

   Dekompozycja nie jest jednoznaczna. *Szerokością drzewiastą* grafu nazywamy maksymalny rozmiar węzła w dekompozycji drzewiastej *minus jeden*. W ten sposób drzewa mają szerokość drzewiastą 1.

   Wiele problemów jest FPT względem szerokości drzewiastej. Dla ustalonej szerokości drzewiastej k, rozkład drzewiasty można obliczyć w czasie liniowym.

   Zastosowania: np. grafy przepływu kontroli w programach strukturalnych mają treewidth co najwyżej 6 (Thorup'98). To pomaga np. w optymalnej alokacji rejestrów podczas kompilacji. Programy z GOTO nie mają ograniczonej treewidth.

5. Problemy optymalizacyjne: znaleźć najmniejsze pokrycie wierzchołkowe, albo największą klikę. O takich problemach często się mówi że są np. **NP**-zupełne, ale co to właściwie znaczy? To nie są problemy decyzyjne.

   Problem optymalizacyjny można rozpatrywać w wersji decyzyjnej: dla danego grafu i liczby k, czy istnieje klika o rozmiarze k? A dla problemu szukania kliki (a nie tylko rozmiaru): dla danego grafu, liczby k i zbioru wierzchołków, czy istnieje klika o rozmiarze k zawierająca te wierzchołki? Wszystkie te problemy się do siebie łatwo redukują, i tak można rozpatrywać problemy optymalizacyjne.

6. *Aproksymacja:* dla ustalonego ϵ, znaleźć rozwiązanie takie że

   $$(\text{optimum–rozwiązanie})/\text{optimum} \leq \epsilon.$$

   To jest dla problemów maksymalizacyjnych; dla minimalizacyjnych chodzi o

   $$(\text{rozwiązanie–optimum})/\text{rozwiązanie} \leq \epsilon.$$

   Dla problemów maksymalizacyjnych, ϵ-algorytm zwraca rozwiązanie równe co najmniej $1-\epsilon$ razy optimum. Dla minimalizacyjnych, zwraca rozwiązanie równe co najwyżej optimum podzielić przez $1-\epsilon$.

   Kres górny wszystkich ϵ, dla jakich istnieje algorytm wielomianowy, nazywamy progiem aproksymacji. Jeżeli próg $= 1$ to w ogóle nie da się aproksymować. Jeżeli próg $= 0$, to mówimy że istnieje PTAS (*polynomial time approximation scheme*). Jeżeli wykładnik algorytmu nie zależy od ϵ, to mówimy że istnieje silny PTAS.

7. Weźmy taki algorytm dla VERTEX-COVER: po kolei brać wierzchołki o największym stopniu, dodawać do pokrycia i usuwać z grafu wraz z sąsiadami, aż graf zrobi się pusty. *Ćwiczenie:* Pokazać, że to aproksymuje minimalne pokrycie z dokładnością nie lepszą niż $\log n$.

   Rozwiązanie: Rozważa się tu grafy dwudzielne: po jednej stronie n wierzchołków, po drugiej $\lfloor \frac{n}{2} \rfloor + \lfloor \frac{n}{3} \rfloor + \cdots + \lfloor \frac{n}{n} \rfloor = n \cdot (\ln n + \mathcal{O}(1))$ wierzchołków z nimi połączonych.

8. **Zadanie:** Pokazać algorytm dla VERTEX-COVER z progiem $\frac{1}{2}$.

   Rozwiązanie: Z dowolnej krawędzi wybieramy oba wierzchołki i usuwamy je wraz z sąsiadami.

   Popularna hipoteza: nie ma lepszego algorytmu. (Wiadomo że nie ma przy założeniu *unique games conjecture*.)

9. Problem komiwojażera: próg aproksymacji równy 1, tj. w ogóle nie ma efektywnej aproksymacji, chyba że $\mathbf{P} = \mathbf{NP}$.

   Dowód: Załóżmy, że jest algorytm aproksymacyjny z progiem $\epsilon < 1$. Zrobimy dokładny algorytm rozwiązywania cyklu Hamiltona. Dla dowolnego grafu $G = (V, E)$, robimy instancję problemu komiwojażera: odległość między wierzchołkami i i j to:

- 1 jeśli jest krawędź z i do j,
- $\frac{|V|}{1-\epsilon}$ jeśli nie ma krawędzi.

Uruchommy na tym nasz przybliżony algorytm dla problemu komiwojażera. Są dwie możliwości:

- algorytm zwrócił podróż o koszcie $|V|$; wtedy jest cykl Hamiltona.
- algorytm zwrócił podróż o koszcie C ostro większym niż $\frac{|V|}{1-\epsilon}$. Wiemy że algorytm ma współczynnik aproksymacji ϵ, a problem jest minimalizacyjny, więc dla optimum O mamy:

  $$1 - \frac{O}{C} \leq \epsilon \qquad \text{więc} \qquad O > |V|$$

  czyli nie ma cyklu Hamiltona.

  To pokazuje nawet, że nie ma nie tylko stałej aproksymacji, ale nawet takiej, gdzie ϵ zbiega wykładniczo do zera (bo wtedy wciąż instancja komiwojażera jest wielomianowego rozmiaru).

10. Problem plecakowy: próg równy 0, istnieje silny PTAS. Dla każdego $\epsilon > 0$, istnieje algorytm ϵ-aproksymacyjny o złożoności $\mathcal{O}(n^3)$.

    Problem plecakowy: mamy danych n obiektów, każdy o wartości v_i i wadze w_i, oraz próg W. Znaleźć podzbiór tych obiektów taki, by maksymalizować sumę v_i, ale by suma w_i nie przekroczyła W.

    Rozwiązanie: Najpierw taki algorytm: niech $V = \max_i(v_i)$. Dla każdego $i = 1..n$ i $v = 0..nV$ obliczamy $W(i, v)$: minimalną wagę, jaką da się uzyskać wybierając spośród pierwszych i obiektów tak, by wartość była dokładnie V. To jest prosty algorytm dynamiczny, bo wzór rekurencyjny łatwo wyprowadzić:

    $$W(i+1, v) = \min(W(i, v), W(i, v - v_{i+1}) + w_{i+1})$$

    To daje koszt $\mathcal{O}(n^2V)$ (razy czynniki logarytmiczne), czyli za dużo. (Przy okazji: problem plecakowy jest FPT względem V.)

    Teraz sztuczka: dla pewnej liczby b, przekształćmy każdą wartość v_i na v'_i przez zastąpienie b najmniej ważnych bitów zerami. Teraz czas wyjdzie tylko $\mathcal{O}(n^2V2^{-b})$, bo b ostatnich bitów wartości możemy wszędzie zignorować.

    Niech S i S' będą optymalnymi wyborami dla zadania z v_i i v'_i. Mamy nierówności:

    $$\sum_{i \in S} v_i \geq \sum_{i \in S'} v_i \geq \sum_{i \in S'} v'_i \geq \sum_{i \in S} v'_i \geq \sum_{i \in S} v_i - n2^b$$

    Jeżeli więc potraktujemy S' jako przybliżone rozwiązanie oryginalnego problemu, to uzyskamy współczynnik aproksymacji co najwyżej $\epsilon = \frac{n2^b}{V}$, bo V to dolne ograniczenie na rozwiązanie oryginalnego problemu.

Dla ustalonego ϵ, weźmy więc $b = \log \frac{\epsilon V}{n}$. Wtedy uzyskamy właściwą aproksymację, a czas działania wyjdzie $\mathcal{O}(\frac{n^2 V}{2^b}) = \mathcal{O}\frac{n^3}{\epsilon}$, czyli wielomianowy.