Aproksymacja najkrótszego nadsłowa

Mateusz Tokarz

June 27, 2020

Opis Problemu

Problem najkrótszego nadsłowa w wersji decyzyjnej jest problemem NP zupełnym. W związku z tym pojawiało się wiele podejść do aproksymacji optymalizacyjnej wersji problemu. Tarhio i Ukkonen oraz Turner napisali dwie niezależne prace naukowe, w których opisano algorytmy aproksymacyjne dla rozważanego problemu. Mimo, że autorzy twierdzili, że skonstruowane przez nich algorytmy powinny być 2-aproksymacyjne, to nie udało im się udowodnić żadnego ograniczenia.

W tym dokumencie przeanalizujemy algorytm opisany przez Minga Li, który jest pierwszą aproksymacyją, dla której udało się udowodnić współczynnik - $log(n)$. Autor pracy twierdzi, że współczynnik w rzeczywistości wynosi co najwyżej 2.

Algorytm

Zakładamy, że podane na wejściu słowa spełniają warunek, iż żadne nie jest podsłowem innego. Możemy tak zrobić, ponieważ wyeliminowanie tego typu sytuacji jest niezmiennicze ze względu na wynik i zajmuje wielomianowy czas.

Definicja. *Dla danych słów* s_1, s_2 *słowo* $m(s_1, s_2)$*, długości nie większej niż* $|s_1| + |s_2|$*, nazywamy scaleniem, jeżeli* s_1 *jest jego prefiksem, a* s_2 *sufiksem (lub* s_2 *prefiksem, a* s_1 *sufiksem).*

Lemat. *Dla danych słów s_1, s_2 istnieje co najwyżej $2min(|s_1|, |s_2|)$ scaleń.*

Dowód powyższego lematu jest trywialny.

Algorytm

1. Input to $S = \{s_1, \ldots, s_n\}$. Jeżeli $|S| = 1$, to zwróć s_1.
2. Zdefiniujmy $T = \emptyset$.
3. Niech s, s' będą słowami minimalizującymi
$$\min_{m-\text{scalenie}} \frac{|m(s, s')|}{v(m(s, s'))},$$
gdzie
$$v(m(s, s')) = \sum_{a \in A(m(s,s'))} |a|$$
dla $A(m(s, s'))$ będącego zbiorem podsłów $m(s, s')$ z S. Wrzucamy $m(s, s')$ do T dla m realizującego powyższe minimum, natomiast wszystkie słowa z $A(m(s, s'))$ wyrzucamy z S.
4. Jeżeli $|S| \leq 1$, to $S = T \cup S$ i przechodzimy do 1.
5. W przeciwnym przypadku przechodzimy do 3.

Złożoność jest oczywiście wielomianowa - przy każdym powrocie do punktu 1, moc zbioru S zmniejszyła się przynajmniej dwukrotnie, natomiast sumaryczna długość słów znajdujących się w nim może się tylko zmniejszyć.

Wyszukiwanie minimum i modyfikacja zbiorów w punktach 3 i 4 jest trywialnie realizowane w czasie wielomianowym.

Pozostaje ograniczyć współczynnik aproksymacji algorytmu.

Współczynnik aproksymacji

Twierdzenie. *Algorytm przedstawiony powyżej jest $log(n)$ aproksymacyjny.*

Dowód. Niech $s_1, \ldots, s_m$ będzie ułożeniem słów wejściowych w kolejności ich występowania w optymalnym rozwiązaniu - jeżeli słowo występuje kilka razy, to rozważamy najbardziej lewe wystąpienie. Dzielimy słowa na następujące

grupy:
Grupa G_1 zawiera $s_1, \ldots, s_i$, gdzie s_i to ostatnie słowo takie, że pierwsze wystąpienie s_1 nachodzi na pierwsze wystąpienie s_i w optymalnym rozwiązaniu. Grupa G_2 budowana jest w analogiczny sposób, z tym że zaczynamy konstrukcje od s_{i+1}. W ten sposób dzielimy całe wejście $s_1, \ldots, s_m$ na rozłączne grupy $G_1, \ldots, G_k$.

Dla każdej grupy G_i wyróżniamy elementy b_i, t_i - pierwsze i ostatnie słowa (we wcześniej wspomnianym porządku) budujące grupę. Jako że b_i nachodzi na t_i, to istnieje scalenie m_i takie, że wszystkie słowa z grupy G_i są podsłowami $m_i(b_i, t_i)$. Zauważmy, że

$$\sum_{i=1}^{k} |m_i(b_i, t_i)| \leq 2n,$$

gdzie n to długość optymalnego rozwiązania, ponieważ każda litera słowa optymalnego jest pokrywana przez maksymalnie dwie grupy.

Wprowadźmy teraz pojęcie kosztu słowa w kontekście analizowanego przez nas algorytmu. Załóżmy, że słowo s odpadało ze zbioru S na rzecz scalenia $m(s', s'')$.

$$cost(s) = \frac{|m(s', s'')|}{v(m(s', s''))} \cdot |s|.$$

Zauważmy, że koszt całego algorytmu (długość słowa wynikowego) jest równa nie więcej niż długość wszystkich słów, które powstały w trakcie pierwszej fazy opróżniania zbioru S - elementów $s_1, \ldots, s_m$. Niech $S_h = A(M_h(B_h, T_h))$ to zbiór elementów wyrzuconych h-tym kroku wspomnianej fazy.

$$COST \leq \sum_{h=1}^{r} |M_h(B_h, T_h)| = \sum_{i=h}^{r} \frac{|M_h(B_h, T_h)|}{v(M_h(B_h, T_h))} \cdot v(M_h(B_h, T_h)) =$$

$$\sum_{h=1}^{r} \sum_{s \in S_h} \frac{|M_h(B_h, T_h)|}{v(M_h(B_h, T_h))} \cdot |s| = \sum_{h=1}^{r} \sum_{s \in S_h} cost(s) = \sum_{s \in S} cost(s).$$

$\square$

Rozważmy teraz grupę G_j i koszt, jaki płacimy za wyrzucenie jej elementów. Grupę G_j dokładnie przed fazą h będziemy oznaczać G_j^h. Sumę długości słów w zbiorze A oznaczamy $||A||$.

$$Cost(G_j) = \sum_{h=1}^{k} \sum_{s \in (G_j^{h+1} \setminus G_j^h)} cost(s) = \sum_{h=1}^{k} \frac{|M_h(B_h,T_h))|}{v(M_h(B_h,T_h))} \cdot (||G_j^{h+1}|| - ||G_j^h||).$$

Jako że nasz algorytm wybierał scalenia $M_h(B_h, T_h)$ realizujące minimalne wartości, to

$$\sum_{h=1}^{k} \frac{|M_h(B_h,T_h))|}{v(M_h(B_h,T_h))} \cdot (||G_j^{h+1}|| - ||G_j^h||) \leq \sum_{h=1}^{k} \frac{|m_j(b_j,t_j))|}{v(m_j(b_j,t_j))} \cdot (||G_j^{h+1}|| - ||G_j^h||) =$$

$$\sum_{h=1}^{k} \frac{|m_j(b_j,t_j))|}{||G_j^h||} \cdot (||G_j^{h+1}|| - ||G_j^h||) \leq |m_j(b_j,t_j))| \cdot Harm(||G_j||).$$

Jako że n jest co najwyżej wielomianowo większe niż każde $||G_j||$, to dostajemy oszacowanie

$$Cost(G_j) \leq |m_j(b_j,t_j))| \cdot log(n),$$

co z poprzednim lematem skutkuje w

$$COST \leq 2nlog(n).$$