Trzecia praca domowa z grafów rzadkich

Wojciech Przybyszewski

Zadanie 1.

Dla $p = 0$ teza jest trywialna, więc załóżmy $p \geq 1$. Pokażemy, że dla $c = \mathrm{wcol}_{p-1}(\mathcal{C})$ teza zadania zachodzi. Weźmy dowolny graf $G \in \mathcal{C}$ i niech $\sigma \in V(G) \to \{1, 2, \ldots, |V(G)|\}$ będzie takim liniowym porządkiem na wierzchołkach G, że $\mathrm{wcol}_{p-1}(G, \sigma) = c$. Niech $D = (V(G), \{(v_i, v_j) : v_j \in \mathrm{WReach}_{p-1}(v_i, \sigma)\})$. Oczywiście graf D jest grafem skierowanym na wierzchołkach grafu G – żeby skończyć zadanie wystarczy pokazać, że ma obie pożądane własności.

Po pierwsze, z definicji grafu D wynika, że z danego wierzchołka $v \in V(D)$ wychodzą skierowane krawędzie do i tylko do wierzchołków należących do zbioru $\mathrm{WReach}_{p-1}(v, \sigma)$. Zbiór ten ma moc $\leq c$, więc oczywiście stopień wyjściowy v jest nie większy niż c.

Po drugie, weźmy dowolne $A \subseteq V(G)$ takie, że $|A| \leq p$ oraz $G[A]$ jest spójny. Niech $a \in A$ będzie najmniejszym, względem porządku σ, wierzchołkiem z A. Ponieważ A jest spójny, to dla dowolnego $b \in A \setminus \{a\}$ mamy ścieżkę z b do a wykorzystującą tylko wierzchołki z A. Oczywiście, najkrótsza ścieżka z b do a wykorzystująca tylko wierzchołki z A nie powtarza żadnego wierzchołka z A, więc ma długość nie większą niż $p-1$. Korzystając z tej obserwacji oraz z faktu, że $\forall_{b \in A} b \geq_\sigma a$, mamy $a \in \mathrm{WReach}_{p-1}(b, \sigma)$. To daje nam, że w grafie D mamy dla każdego $b \in A \setminus \{a\}$ krawędź (b, a). To dowodzi, że graf D ma obie własności, czyli kończy (konstruktywny) dowód zadania 1.

Zadanie 2.

Ustalmy $\mathcal{C}$ i $d \in \mathbb{N}$ tak jak w treści zadania. Udowodnimy, że istnieje szukany schemat etykietowania w odległości d, omawiając działanie algorytmów kodowania i dekodowania. Weźmy dowolny graf $G \in \mathcal{C}$. Na mocy zadania 4. z ćwiczeń 5. wiemy, że istnieje liniowy algorytm obliczający liniowy porządek σ na wierzchołkach G, taki, że $\mathrm{wcol}_d(G, \sigma) \leq c$, gdzie $c = c(\mathcal{C}, d)$.

Zaczniemy od omówienia algorytmu kodowania, który zwróci etykietowanie wierzchołków G długości $O(\log n)$. Algorytm ten zaczyna od odliczenia wspomnianym już algorytmem porządku liniowego σ (czas liniowy). Następnie dla każdego wierzchołka $v \in V(G)$ oblicza jego zbiór $\mathrm{WReach}_d(v, \sigma)$, który jest mocy nie większej niż c (może być naiwnie zaimplementowane w czasie sześciennym). W kolejnym kroku dla każdego wierzchołka v i każdego wierzchołka $u \in \mathrm{WReach}_d(v, \sigma)$ oblicza długość najkrótszej ścieżki będącej świadkiem, że $u \in \mathrm{WReach}_d(v, \sigma)$ (naiwnie zaimplementowany czas kwadratowy (BFS dla każdej z cn par)) – zauważmy, że podobnie jak w zadaniu 1. te długości są mniejsze niż n. Po tych obliczeniach algorytm może już wyprodukować etykietkę dla danego wierzchołka v – składa się ona z binarnego zapisu $\sigma(v)$, następnie znaku $\#$, a potem $|\mathrm{WReach}_d(v, \sigma)|$ par oddzielonych znakami $\#$ – dla każdego $u \in \mathrm{WReach}_d(v, \sigma)$ zapisujemy $\sigma(u)$, znak $\#$ i długość najkrótszej ścieżki od v do u świadczącej o tym, że $u \in \mathrm{WReach}_d(v, \sigma)$. W treści zadania oczekiwaliśmy, że etykieta będzie składać się tylko ze znaków 0 i 1, natomiast w tym algorytmie używamy także znaku $\#$ – można łatwo to naprawić poprawiając wygenerowane etykiety przykładając do nich funkcję, która zamienia kolejne litery – $0 \mapsto 00, 1 \mapsto 11, \# \mapsto 10$. Dostajemy w ten sposób etykietowania nie dłuższe niż $2(\lceil \log n \rceil + 1 + c(\lceil \log n \rceil + 1 + \lceil \log n \rceil + 1)) = O(\log n)$. Ponieważ, jak już zauważyliśmy, działa on w czasie wielomianowym i zwraca etykiety odpowiedniej długości, to jest poprawnym algorytmem kodowania.

Omówimy teraz algorytm dekodowania i wykażemy, że działa poprawnie. Załóżmy więc, że mamy dwie etykiety dla wierzchołków $u, v \in V(G)$ – od razu możemy z nich odczytać zbiory $\mathrm{WReach}_d(u, \sigma)$ i $\mathrm{WReach}_d(v, \sigma)$ oraz wspomniane już długości najkrótszych ścieżek (dla $v' \in \mathrm{WReach}_d(v, \sigma)$ oznaczmy ją przez $l_v(v')$). Następnie dekoder zwraca liczbę $\min\{l_u(w) + l_v(w) : w \in \mathrm{WReach}_d(u, \sigma) \cap \mathrm{WReach}_d(v, \sigma)\}$, o ile jest ona nie większa niż d – w przeciwnym razie zwraca $+\infty$. Algorytm dekodera jest oczywiście wielomianowy, pozostaje więc wykazać, że jest poprawny. Oczywiście dla dowolnych wierzchołków $u, v \in V(G)$, które są w odległości nie większej niż d, na najkrótszej ścieżce między nimi istnieje wierzchołek σ-najmniejszy – oznaczmy go w. Jeśli wspomnianą najkrótszą ścieżkę podzielimy na dwie części od u do w i od w do u to dostaniemy ścieżki będące świadkami, że odpowiednio $w \in \mathrm{WReach}_d(u, \sigma)$ oraz $w \in \mathrm{WReach}_d(v, \sigma)$. Ponadto, jako że rozważaliśmy najkrótsze ścieżki z u do v, to również otrzymaliśmy najkrótsze

ścieżki z u do w oraz z w do v – to wszystko implikuje, że dekoder nie zwróci liczby większej niż odległość z u do v, o ile jest ona nie większa niż d. Z drugiej strony, jeśli dekoder zwraca liczbę $m < +\infty$, to musi istnieć wierzchołek $w \in \mathrm{WReach}_d(u, \sigma) \cap \mathrm{WReach}_d(v, \sigma)$ taki, że mamy ścieżki z u do w i z w do v o łącznej długości nie większej niż d, co oznacza, że dekoder zwraca wartość nie mniejszą niż rzeczywista odległość z u do v. Tak więc dekoder musi zwrócić rzeczywistą odległość od u do v jeśli jest ona nie większa niż d, oraz $+\infty$ w przeciwnym wypadku. To dowodzi poprawności działania dekodera, a więc również kończy konstruktywny dowód istnienia schematu etykietowania w odległości d.

Zadanie 3.

Rozważany w treści zadania algorytm opiszemy w kolejnych fazach, aby ułatwić jego analizę. Na samym początku $\forall_{v \in V(G)}$ obliczamy $\mathrm{WReach}_r(v, \sigma)$, co będziemy w skrócie zapisywać $\mathrm{WReach}(v)$, jako że σ i r są ustalone na cały czas działania algorytmu. Następnie w pętli przeglądamy wszystkie wierzchołki A w kolejności od σ-największego do σ-najmniejszego utrzymując zbiór U (początkowo pusty). Jeśli w pętli przeglądamy wierzchołek v, to sprawdzamy, czy w U mamy jakiś wierzchołek u taki, że $v \in \mathrm{WReach}(u)$. Jeśli nie, to dodajemy v do U, jeśli tak, to wykonujemy kolejny obrót pętli. Oczywiście widać, że możemy tę pętlę zaimplementować tak, aby działała w czasie wielomianowym. Zauważmy, że każde dodanie wierzchołka v do U usuwa co najwyżej $c-1$ potencjalnych kandydatów do U – wierzchołki z $\mathrm{WReach}(v)$ inne niż v. To daje nam, że $|U| \geq \frac{|A|}{c}$. Co więcej, dla dowolnych $v_1, v_2 \in U$ mamy $v_1 \notin \mathrm{WReach}(v_2)$ oraz $v_2 \notin \mathrm{WReach}(v_1)$.

Gdyby udało nam się wskazać m wierzchołków $\{v_1, \ldots, v_m\} \subseteq U$ takich, że istnieje $S \subseteq G, |S| \leq c$ i $\forall_{i \neq j} \mathrm{WReach}(v_i) \cap \mathrm{WReach}(v_j) \subseteq S$, to moglibyśmy zwrócić $v_1, \ldots, v_m$ oraz $S \setminus \{v_1, \ldots, v_m\}$ jako wynik działania algorytmu. Istotnie, dla dowolnych $i \neq j$, gdyby w $G \setminus (S \setminus \{v_1, \ldots, v_m\})$ istniałaby ścieżka długości $\leq r$ między v_i oraz v_j to istniałby na niej wierzchołek σ-najmniejszy. Należy on oczywiście do $\mathrm{WReach}(v_i) \cap \mathrm{WReach}(v_j)$. Ponieważ $v_i, v_j \notin \mathrm{WReach}(v_i) \cap \mathrm{WReach}(v_j) \subseteq S$, to dostajemy sprzeczność. Możemy więc lekko zmienić nasz cel – mając już policzony zbiór U szukamy wspomnianych już zbiorów $\{v_1, \ldots, v_m\}$ oraz S. Jeśli nam się to uda, będziemy mieli poprawnie działający algorytm.

Taki algorytm możemy zrealizować za pomocą procedury rekurencyjnej **solve** o arności 5 – przyjmuje ona graf G, zbiór U, liczby m i c oraz tablicę WReach zdefiniowaną jak powyżej, a zwraca omówione zbiory $\{v_1, \ldots, v_m\}$ oraz S. Działa ona następująco – zaczyna od ustalenia pomocniczego zbioru W (początkowo pustego) i skopiowania zbioru U do zbioru U'. Następnie, pobiera losowy wierzchołek $v \in U'$, dodaje go do W i z U' usuwa wszystkie $u \in U$ takie, że $\mathrm{WReach}(v) \cap \mathrm{WReach}(u) \neq \emptyset$. Powtarza te instrukcje w pętli **while** dopóki U' nie jest pusty. Zauważmy, że jeśli otrzymany na koniec zbiór W będzie mocy przynajmniej m, to możemy zwrócić go jako zbiór B, a jako S zwrócić zbiór pusty. To, że spełniają one założenia omówione w drugim akapicie jest oczywiste. Pozostaje drugi przypadek – gdy $|W| < m$. Wtedy mamy taki wierzchołek, który przy dodawaniu go do W usunął z U' co najmniej $\frac{|U|}{|W|} \geq \frac{|U|}{m}$ wierzchołków. Oznaczmy go przez w, zaś $\{u \in U : \mathrm{WReach}(w) \cap \mathrm{WReach}(u) \neq \emptyset\}$ przez N_w. Każdemu $n \in N_w$ możemy przyporządkować jakiś $e \in \mathrm{WReach}(w)$ w taki sposób, że $e \in \mathrm{WReach}(w) \cap \mathrm{WReach}(n)$. Ponieważ $|\mathrm{WReach}(w)| \leq c$, to jakieś e będzie przyporządkowane co najmniej $\frac{|N_w|}{c} \geq \frac{|U|}{cm}$ razy. Te rozważania prowadzą nas do wniosku – jeśli $|W| < m$ to mamy jakiś $e \in V(G)$ taki, że $|\{v \in U : e \in \mathrm{WReach}(v)\}| \geq \frac{|U|}{cm}$. Możemy więc zapamiętać, że dodamy e do S

i wywołać rekurencyjnie procedurę **solve** z parametrami G, $\{v \in U : e \in \mathrm{WReach}(v)\}$, m, $c-1$, $\lambda v.\, \mathrm{WReach}(v) \setminus \{e\}$, a następnie zwrócić to co da ona, poprawiając zbiór S poprzez dodanie e. Zauważmy, że wszystkim wierzchołkom w nowym zbiorze U nowa funkcja WReach zwraca zbiory mocy nie większej niż $c-1$, więc przy rekurencyjnym schodzeniu nie dostaniemy zbioru S mocy większej niż $c-1$, więc to co zwrócimy będzie mocy nie większej niż c. Pozostaje tylko pytanie, czy nie natkniemy się nigdzie na problem – mianowicie, czy c nie spadnie nigdy poniżej 0. Oczywiście, gdy $c = 1$, to zbiór U, który **solve** dostanie jako argument ma tę własność, że $\forall_{v \in U}\, \mathrm{WReach}(v) = \{v\}$. To oznacza, że dostaniemy $W = U$. Pytanie jaka będzie moc U, kiedy c spadnie do 1. Na początku w A mieliśmy przynajmniej $4 \cdot (2cm)^{c+1}$ wierzchołków. Do pierwszego U włożyliśmy ich przynajmniej $4 \cdot (2cm)^c \cdot 2m \geq 4 \cdot (2cm)^c$. Potem w każdym kroku, jeśli schodzimy z rekurencją, dzielimy zbiór wierzchołków przez $cm \leq 2cm$, więc jeśli zejdziemy $c-1$ razy do $c = 1$, to będziemy mieli przynajmniej $4 \cdot (2cm)^1 = 8cm \geq m$ wierzchołków, czyli wszystko jest w porządku. Co więcej, widać że ten algorytm działa nawet przy założeniu $|A| \geq (cm)^c$. Te wszystkie rozważania kończą rozwiązanie zadania.

Dołączam również, w celu lepszego zobrazowania algorytmu, jego pseudokod:

Algorithm 1: Główna część algorytmu

Dane: graf G, porządek σ na $V(G), A \subseteq V(G), r \in \mathbb{N}, m \in \mathbb{N}$
Wynik: $B \subseteq A, S \subseteq V(G)$ jak w treści zadania
for $v \in V(G)$ **do**
  $\mathrm{WReach}(v) \leftarrow \mathrm{WReach}_d(v, \sigma)$;
end
$U \leftarrow \emptyset$;
for $v \in A$ **from** *σ-największy* **to** *σ-najmniejszy* **do**
  if $\exists u \in U \wedge v \in \mathrm{WReach}(u)$ **then**
    continue;
  end
  $U \leftarrow U \cup \{v\}$;
end
$B, S \leftarrow$ **solve**(G, U, m, c, WReach);
return $B, S \setminus B$

Algorithm 2: Procedura solve

Dane: graf G, $U \subseteq V(G)$, m, c i funkcja $\mathrm{WReach} : U \to \mathcal{P}(V(G))$ taka, że zwracane podzbiory są mocy nie większej niż c
Wynik: m wierzchołków z U i zbiór $S, |S| \leq c$ takie, że przecięcia WReach każdej pary z tych m wierzchołków zawierają się w S.
$W \leftarrow \emptyset$;
$U' \leftarrow U$;
while $U' \neq \emptyset$ **do**
  $v \leftarrow$ losowy wierzchołek z U';
  $N_v \leftarrow \{u \in U' : \mathrm{WReach}(u) \cap \mathrm{WReach}(v) \neq \emptyset\}$;
  $W \leftarrow W \cup \{v\}$;
  $U' \leftarrow U' \setminus N_v$;
end
if $|W| \geq m$ **then**
  return $W, \emptyset$;
end
$e \leftarrow$ taki element $\in V(G)$, że $|\{v \in U : e \in \mathrm{WReach}(v)\}|$ maksymalne;
$B, S \leftarrow$**solve**($G, \{v \in U : e \in \mathrm{WReach}(v)\}, m, c-1, \lambda v.\, \mathrm{WReach}(v) \setminus \{e\}$;
return $B, S \cup \{e\}$;