Rozdział 1

Rozdział 2

Lista 2.

2.1. 0pkt

Przeczytaj notatkę o algorytmach zachłannych...

2.2. 1pkt

Danych jest n odcinków $I_j =< p_j, k_j >$, leżących na osi OX, $j = 1, \ldots, n$. Ułóż algorytm znajdujący zbiór $S \subseteq \{I_1, \ldots, I_n\}$, nieprzecinających się odcinków, o największej mocy.

Sortujemy odcinki rosnąco wg. końców k_j. Wybieramy pierwszy, gdyż najwcześniej się zakończy. Następnie pierwszy kolejny, który się zmieści po poprzednio wybranym.

Dowód: Niech nasz algorytm daje ciąg $S = \{s_1, s_2, \ldots, s_n\}$. Załóżmy, że istnieje lepszy, optymalny ciąg $O = \{o_1, o_2, \ldots, o_n\}$. Weźmy pierwszą pozycję i, na której rozwiązania się różnią, czyli taką, że: $s_i \neq o_i$, oraz $j < i \Rightarrow u_j = o_j$. Rozważmy dwa przypadki:

- Jeśli kończy się na tej samej pozycji to nie ma znaczenia, który z nich został wybrany. Zatem rozwiązanie $\{o_1, \ldots, o_i\}$ nie jest lepsze od $\{s_1, \ldots, s_i\}$. (sprzeczność)
- Jeżeli odcinek o_i kończy się wcześniej niż u_i to zostałby on wybrany przez nasz algorytm zgodnie z lokalny, zachłannym kryterium. (sprzeczność)

Rozumowanie powtarzamy indukcyjnie dla kolejnych pozycji na których rozwiązania się różnią. Algorytm zachłanny daje więc rozwiązanie optymalne.

2.3. 1pkt

Rozważ następującą wersję problemu wydawania reszty: dla danych liczb naturalnych a, b ($a \leqslant b$) chcemy przedstawić ułamek $\frac{a}{b}$ jako sumę różnych ułamków o licznikach równych 1. Udowodnij, że algorytm zachłanny zawsze daje rozwiązanie. Czy zawsze jest to rozwiązanie optymalne (tj. o najmniejszej liczbie składników)?

Wytwarzanie ułamka egipskiego mniejszego od $\frac{x}{y}$ o najwiekszym mianowniku:

$$\frac{x}{y} \rightarrow \frac{1}{\lceil \frac{y}{x} \rceil}$$

Licznik wyrażenia $\frac{x}{y} - \frac{1}{\lceil \frac{y}{x} \rceil} = \frac{x\lceil \frac{y}{x} \rceil - y}{x\lceil \frac{y}{x} \rceil}$ będzie malał, lecz zawsze będzie $\geqslant 0$:

Sprawdźmy, czy faktycznie $0 \leqslant x\lceil \frac{y}{x} \rceil - y < x$

$$x\lceil \frac{y}{x} \rceil - y < x \Leftrightarrow x\lceil \frac{y}{x} \rceil - x < y \Leftrightarrow \lceil \frac{y}{x} \rceil - 1 < \frac{y}{x}.$$

Co jest oczywiście prawdą. Dodatkowo: $x\lceil \frac{y}{x} \rceil - y \geqslant 0$. Więc ułamek zawsze się zmiejsza, i zatrzyma się na 0.
Kontrprzykład na optymalność. Nasz da jakieś gówno, optymalny da:

$$\frac{5}{121} = \frac{1}{33} + \frac{1}{121} + \frac{1}{363}$$

2.4. 2pkt

Udowodnij poprawność algorytmu Boruvki Solina.

2.4.1. Lemat 1

W każdym momencie działania algorytmu, oraz po jego zakończeniu w E' nie będzie cyklu.

Dowód: Załóżmy nie wprost, że podczas działania algorytmu w którymś etapie pojawiła się spójna składowa, w której jest cykl. Oznaczmy ją jako S. Rozważmy następujące sytuacje:

- S powstała przez połączenie dwóch superwierzchołków v_1 i v_2. Oznacza to, że do zbioru E' zostały dołączone krawędzie e_i i e_j. Ponieważ e_i została dołączona jako najlżejsza krawędź incydentna do v_1 więc $C(e_i) < C(e_j)$. Ale skoro e_j została dołączona jako najlżejsza krawędź incydentna do v_2 to musi zachodzić (Pamiętajmy, że w grafie nie ma krawędzi o takiej samej wadze) - sprzeczność.
- S powstała przez połączenie się trzech lub więcej superwierzchołków. Podzielmy powstały cykl C na następujące części: niech $v_1, .../v_l$ będą kolejnymi superwierzchołkami należącymi do C a $e_1, ..., e_l$ będą kolejnymi krawędziami należącymi do C, które zostały dodane w zakończonym właśnie etapie algorytmu. W C krawędzie e_i oraz superwierzchołki v_i występują na przemian. Z zasady działania algorytmu możemy stwierdzić, że aby powstał taki cykl, musi zachodzić $C(e_1) < C(e_2)... < C(e_l) < C(e_1)$ - Sprzeczność.

2.4.2. Lemat 2

W każdym etapie działania algorytmu otrzymujemy dla każdego superwierzchołka minimalne drzewo rozpinające.

Dowód:

- Gdy zostanie zakończony etap 1:

  Załóżmy, że istnieje taki superwierzchołek v_i, który nie jest minimalnym drzewem rozpinającym poddrzewa złożonego z wierzchołków należących do v_i. Weźmy więc takie minimalne drzewo rozpinające T. Istnieje krawędź e_i taka, że $e_i \in E(v_i)$ oraz $e_i \notin E(T)$. Dodajmy e_i. W T powstał cykl. Ponieważ e_i jest incydenta do pewnego wierzchołka z tego cyklu, istnieje więc inna krawędź e_i' incydentna do tego samego wierzchołka. Jednak z tego, że $e_i \in E(v_i)$ wynika, że $C(e_i) < C(e_i')$. Jeśli usuniemy krawędź e_i' z T otrzymamy mniejsze drzewo rozpinające, co jest sprzeczne z założeniem o minimalności T.

- Gdy zostanie zakończony etap 2:

  Z poprawności etapu 1 wiemy, że istnieje takie wywołanie etapu 2, w którym każdy z superwierzchołków jest minimalnym drzewem rozpinającym. Jest to choćby pierwsze wywołanie. Załóżmy zatem, że dla pewnego wywołania tego etapu otrzymano superwierzchołki będące minimalnymi drzewami rozpinającymi, jednak scaliło przynajmniej dwa z nich w taki sposób, że dało się otrzymać mniejsze drzewo rozpinające. Niech etap k-ty będzie pierwszym takim etapem, w którym coś się popsuło. Niech E_1' będzie zbiorem krawędzi przed wywołaniem etapu k, a E_2' będzie zbiorem krawędzi po jego wywołaniu. Niech T będzie minimalnym drzewem rozpinającym takim, że $V(T) = V(v_i)$, ale że $E(T) \neq E(v_i)$. Istnieje więc krawędź $e_i \in E(v_i)$ oraz $e_i \notin E(T)$.

  Fakt: Krawędź dodana podczas k-tego wywołania. (Nie może należeć do E_1' gdyż inaczej superwierzchołek do którego by należała nie byłby minimalnym drzewem rozpinającym, co jest sprzeczne z dowodem dla pierwszego etapu i założeniem, że wywołanie k-te jest najmniejszym wywołaniem, które zwróciło nieoptymalne drzewa) Dodajmy krawędź e_i do $E(T)$. W T powstał cykl. Ponieważ e_i jest najmniejszą krawędzią incydentną do pewnego superwierzchołka z tego cyklu, istnieje więc inna krawędź incydenta do tego samego superwierzchołka. Jednak jej waga jest większa niż waga krawędzi e_i, zatem zastąpienie jej krawędzią e_i da nam mniejsze drzewo rozpinające co jest sprzeczne z założeniem o optymalności T.

2.5. 2pkt

Ułóż algorytm, który dla danego spójnego grafu G oraz krawędzi e sprawdza w czasie $O(n+m)$, czy krawędź e należy do jakiegoś minimalnego drzewa spinającego grafu G. Możesz założyć, że wszystkie wagi krawędzi są różne.

2.6.

System złożony z dwóch maszyn A i B wykonuje n zadań. Każde z zadań wykonywane jest na obydwu maszynach, przy czym wykonanie zadania na maszynie B można rozpocząć dopiero po zakończeniu wykonywania go na maszynie A. Dla każdego zadania określone są dwie liczby naturalne a_i i b_i określające czas wykonania i-tego zadania na maszynie A oraz B (odpowiednio). Ułóż algorytm ustawiający zadania w kolejności minimalizującej czas zakończenia wykonania ostatniego zadania praz maszynę B.