Zadanie 3

Danych jest n odcinków $I_j = < p_j, k_j >$, leżących na osi OX, $j = 1, \dots, n$. Ułóż algorytm znajdujący zbiór $S \subseteq A_1, \dots, A_n$ nieprzecinających się odcinków, o największej mocy.

Rozwiązanie: Proponowany algorytm:

uporządkuj zbiór $R = A_1, \dots, A_n$ względem k_i rosnąco
dopóki A jest niepusty, rób
    wybierz odcinek A_i o najmniejszym k_i z R
    jeśli $p_i \geqslant k_j$, gdzie A_j jest ostatnio dodanym odcinkiem w zbiorze S, dodaj A_i do S
    usuń A_k z R

Złożoność tego algorytmu to $O(n)$. Pokażmy, że znajduje on rozwiązanie poprawne.

Dowód. Weźmy pewne rozwiązanie optymalne A. Załóżmy, że A jest lepsze od rozwiązania B wygenerowanego przez powyższy algorytm. Weźmy dwa odcinki A_i, B_i, które się różnią w obu porządkowaniach. Mamy do rozważenia trzy sytuacje:

- $A_i.k < B_i.k$ (A_i kończy się **wcześniej** niż B_i). Ta sytuacja nie będzie mieć miejsca, bo powyżej zaproponowany algorytm bierze odcinki, które kończą się najwcześniej, jak to możliwe.
- $A_i.k = B_i.k$ (A_i kończy się **w tym samym momencie** co B_i). Wybór A_i lub B_i w tym kroku nie ma znaczenia, gdyż po końcu obydwu odcinków algorytmy mają taki sam zbiór odcinków do wyboru. Sprzeczne z założeniem.
- $A_i.k > B_i.k$ (A_i kończy się **później** niż B_i). Wtedy uporządkowanie A może być albo tak samo dobre (w sensie tak samo liczne) co B, lub gorsze (w sensie mniej liczne), gdyż kończy się później, i w najlepszym wypadku nie zmniejszy nam liczby możliwych do wybrania odcinków w kolejnym kroku, albo zablokuje nam dostęp do jakiegoś odcinka (odcinek skończy się później niż zacznie się jakiś inny). Sprzeczne z założeniem.

Zatem pokazaliśmy, że A nie jest lepsze niż B, co ciągnie za sobą optymalność rozwiązania B. □

Zadanie 4

Rozważ nastepującą 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 zachłany zawsze znajduje poprawne rozwiązanie. Czy to rozwiązanie zawsze jest optymalne (tzn. znajduje sumę o najmniejszej liczbie składników)?

Rozwiązanie: Na początku pokażmy pewną równość, na której bazuje algorytm zachłanny:

$$\frac{x}{y} = \frac{1}{\lceil y/x \rceil} + \frac{-y \mod x}{y\lceil y/x \rceil} = \frac{y}{y\lceil y/x \rceil} + \frac{-y \mod x}{y\lceil y/x \rceil} = \frac{y + (-y \mod x)}{y\lceil y/x \rceil} \overset{*}{=} \frac{y + x\lceil y/x \rceil - y}{y\lceil y/x \rceil} = \frac{x\lceil y/x \rceil}{y\lceil y/x \rceil} = \frac{x}{y}$$

Uzasadnienie przejścia (*):

$$(n = \left\lfloor \frac{n}{k} \right\rfloor \cdot k + (n \mod k)) \Rightarrow ((n \mod k) = n - \left\lfloor \frac{n}{k} \right\rfloor \cdot k) \Rightarrow ((-n \mod k) = -n - \left\lfloor \frac{-n}{k} \right\rfloor \cdot k = \left\lceil \frac{n}{k} \right\rceil \cdot k - n)$$

To, że algorytm zawsze znajdzie poprawne rozwiązanie, jest oczywiste. Znajduje natomiast skończone rozwiązanie, bo w każdym kroku zmniejszamy ułamek do rozszerzenia, ponieważ $\forall x, y(-y \mod x < x)$. Czy ten algorytm jest optymalny? Nie. Kontrprzykład (najpierw zachłannie, potem optymalnie):

$$\frac{9}{20} = \frac{1}{3} + \frac{1}{9} + \frac{1}{180} = \frac{1}{4} + \frac{1}{5}$$

Zadanie 5

Udowodnij poprawność algorytmu Borůvki (Sollina).

Rozwiązanie: Do przeprowadzenia dowodu przyda się następujące twierdzenie.

Lemat 1. ***Cut property.*** *Krawędź e należy do MST grafu G wtedy i tylko wtedy, gdy istnieje podział zbioru wierzchołków grafu G na A i B taki, że spośród wszystkich krawędzi mających jeden koniec w A, a drugi w B, e jest najlżejsza.*

Dowód. Pokażemy dwie implikacje.

- $\Rightarrow$: Weźmy dowolne e, MST, że $e \in MST$. Ściągnięcie krawędzi e z tego drzewa spowoduje rozspójnienie go. Podzielmy to rozspójnienie na dwie składowe: A i B. Załóżmy, że istnieje krawędź e' o mniejszej wadze niż e, która łączyłaby A i B. Gdyby taka krawędź istniała, to powstałoby nowe MST o mniejszej wadze. Sprzeczność z faktem, że $e \in MST$, co pokazuje, że e jest minimalną krawędzią łączącą A i B.

- $\Leftarrow$: Na mocy kontrapozycji można alternatywnie pokazać, że jeśli krawędź e z grafu G nie należy do żadnego MST, to dla każdego podziału na A i B nie jest ona najlżejszą łączącą A i B. Zauważmy, że dodanie e do tego MST skutkuje powstaniem cyklu. Załóżmy nie wprost, że istnieje podział A i B taki, że e jest najlżejszą łączącą je krawędzią. Przynajmniej jedna krawędź w cyklu inna niż e, i z tego samego cyklu, musi przechodzić między A i B. Usunięcie tej krawędzi skutkuje powstaniem nowego MST, którego waga będzie mniejsza. Sprzeczność z faktem, że $e \notin MST$.

□

Korzystając z *Cut property*, udowodnienie poprawności algorytmu Borůvki (Sollina) jest łatwe.

Dowód. W dowolnym kroku algorytmu, gdzie dla jakiegoś (super)wierzchołka wybierana jest krawędź, można potraktować tą sytuację jak podział wierzchołków na A i B, gdzie A jest wierzchołkiem (zbiorem wierzchołków, które składają się na superwierzchołek), dla którego wybiera się krawędź, a B - wszystkimi innymi. Algorytm zawsze wybiera najlżejszą krawędź, a z *Cut property* wiemy, że należy ona do MST. Zatem wszystkie krawędzie wybierane przez algorytm budują MST.

Należy jeszcze pokazać, że ten algorytm nie stworzy żadnego cyklu. Załóżmy nie wprost, że pewien superwierzchołek zawiera w sobie cykl. Oznaczamy jego wierzchołki przez $v_1, v_2, \ldots, v_n$, krawędzie przez $e_1, e_2, \ldots, e_n$, gdzie $e_k = \{v_k, v_{k+1}\}$. Zgodnie z algorytmem, dla v_1 wybrana została krawędź e_1, dla v_2 - e_2, i tak dalej. Przez $w(x)$ oznaczamy wagę wierzchołka x. W takim razie, z zasady działania algorytmu, wnioskujemy, że:

$$w(e_1) > w(e_2) > \cdots > w(e_n) > w(e_1)$$

co jest oczywiście nieprawdą dla jakichkolwiek wag (pamiętamy, że wagi na krawędziach są różne). Sprzeczność, że w superwierzchołku powstaje cykl. □

Zadanie 6

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 rozpinającego grafu G. Możesz założyć, że wszystkie wagi krawędzi są różne.

Rozwiązanie: Udowodnimy sobie najpierw pewną właściwość, która przyda się w konstrukcji algorytmu.

Lemat 2. ***Cycle property.*** *Krawędź e nie jest maksymalna na żadnym cyklu w grafie G wtedy i tylko wtedy, gdy e należy do przynajmniej jednego z minimalnych drzew rozpinających grafu G.*

Dowód. Pokażemy dwie implikacje.

- $\Rightarrow$: Załóżmy, że e nie jest maksymalna na żadnym cyklu z G, oraz że nie należy do żadnego MST. Są dwie możliwości:
  - e nie należy do żadnego cyklu - wtedy trywialnie pokazujemy, że $e \in MST$. Sprzeczność.
  - e należy do jakiegoś cyklu (lub cykli) - weźmy dowolne $MST \subset G$ i dołóżmy do niego e. Powstanie nam cykl, bo $e \notin MST$. Zauważmy, że skoro e nie jest maksymalne na tym cyklu, to istnieje pewne e', które ma maksymalną wagę. Ściągnięcie e' daje nam drzewo rozpinające o mniejszej wadze niż MST. Zatem e należy do minimalnego drzewa rozpinającego. Sprzeczność.
- $\Leftarrow$: Na mocy kontrapozycji alternatywnie pokazujemy, że jeśli krawędź e jest maksymalna na pewnym cyklu w grafie G, to e nie należy do żadnego MST. Załóżmy nie wprost, że e należy do MST. Ale wtedy możemy zdjąć e z cyklu i dodać inną krawędź z cyklu o mniejszej wadze, która nie jest w MST, a wtedy otrzymamy drzewo rozpinające o mniejszej wadze. Sprzeczność, że e należy do MST.

□

Wiedząc powyższe, konstruujemy algorytm:

$w \leftarrow$ waga krawędzi e
opcjonalnie: wyciągnij krawędź $e = \{u, v\}$ z G
przechodź z u do następnych wierzchołków algorytmem DFS/BFS z dodatkowym założeniem:
    $e' \leftarrow$ krawędź łącząca wierzchołek, do którego algorytm chce przejść z wierzchołkiem wyjściowym
    $w' \leftarrow$ waga krawędzi e'
    jeśli $w' > w$, przejdź normalnie do następnego wierzchołka
    wpp. zignoruj tą krawędź
jeśli odwiedzono wierzchołek v, zwróć **NIE**
wpp. zwróć **TAK**

Dowód poprawności. Usunięcie krawędzi $e = \{u, v\}$ może rozspójnić ten graf w przypadku, gdy e jest mostem tego grafu. Gdy e jest mostem, to oczywiście należy do MST. Gdy jednak usunięcie e nie rozspójnia grafu, to znaczy, że e nie jest maksymalna na żadnym cyklu, co nam nie pozwala na dotarcie z u do v, gdyż waga nie jest wystarczająco duża na przejście przez cały cykl. Wtedy z *Cycle property* dostajemy, że $e \in MST$.