ZADANIE 2.

Udowodnij, że algorytm Kruskala znajduje minimalne drzewa spinające poprzez przyrównanie tych drzew do drzew optymalnych

Po pierwsze, rozpiszmy co robi ten algorytm:

```
E' <- [] pusty zbior
C <- E

while |E'|<n+1
    e <- min(C)
    if (E' + e nie ma cyklu)
        E' <- E' + e
    C <- C - e
```

Niech T będzie drzewem minimalnym, a E' będzie wynikiem algorytmu. Chcę pokazać, że E' jest minimalnym spinning tree. To że jest wogóle tree, to widać, elo.

Co, jeśli istnieje $e \in E' \setminus T$? Wtedy $e \cup T$ będzie miało cykle, bo T zachaczało o wszystkie krawędzie, jara jara jara. Czyli w ten sposób tworzymy sobie pewien cykl. Super. To teraz w T musiałam mieć jakieś inne przejście w tym cyklu, niech to będzie f. No ale miara e była mniejsza, bo inaczej to był f próbowała dołączyć przed e do E' i by nie stwierdziło, że się zacykli. Czyli graf $E' \setminus e \cup f$ będzie miał troszkę większą sumę niż E'.

Robiąc tak indukcyjnie aż nam się wszystkie wierzchołki pokryją, dojdziemy do grafu pokrywającego się z T, ale o większej mocy niż E'. Czyli to nie mogło być tak, że takie usuwanie krawędzi faktycznie zwiększało sumę, tylko to wszystko musiało zostawać tak samo, więc suma z E' to to samo co te sumy w międzyczasie, a one z kolei równały się sumie T.

ZADANIE 3.

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 przecinków, o największej mocy.*

Mam listę P i K, odpowiednio początków i końców tych pyśków. Może jakoś od razu sobie założę, że mam dwójkę? Indeks i wartość początku/końca? wtedy mogę łatwo wiedzieć gdzie był czyj koniec?

Dobra, to teraz biorę pierwszego pyśka z końców i wkładam go sobie do S. W ten sposób gwarantuję sobie, że śmignie, bo nawet jeśli jakiś z dalszym końcem da mi ten sam wynik, to nie wiem startowo że tak będzie, więc po prostu wybieram to, co niszczy mi najmniej innych wyborów.

W następnej kolejności idę dalej przez K aż znajdę pierwszy koniec, który ma początek ostro większy niż to co jako pierwsze wybrałam. Tutaj jest ta sama historia. No i tak dalej aż do końca listy K.

ZADANIE 4.

Rozważ następująca wersję problemu wydawania reszty: dla danych liczb naturalnych $a, b, (a \leq b)$ *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?)*

Rozumiem, że algorytm zachłanny po prostu leci przez kolejne liczby naturalne i sprawdza, czy dodając je dostaję coś większego, czy się mieszczę? I jeszcze sprawdzam, czy jest sens iść dalej, to znaczy czy mam już dokładnie ten ułamek który chciałam.

Chyba daje to najlepsze, bo na pewno mogę każdy ten ułamek $\frac{1}{p}$ rozbić dalej, ale to wtedy zwiększam p niepotrzebnie i najpierw te mniejsze zaliczę w pierwszej kolejności. Jeżeli istniałaby reprezentacja bardziej optymalna, to miałaby jeden ułamek z mniejszym mianownikiem, ale my szliśmy od największych mianowników, czyli sprzeczność?

ZADANIE 5.

Ułóż algorytm, który dla danego n*-wierzchołkowego drzewa i liczby* k*, pokoloruje jak najwięcej wierzchołków tak, ba na każdej ścieżce prostej było nie więcej niż* k *pokolorowanych wierzchołków.*

Czy ja chcę to robić, idąc od liści i pałując te ścieżki od liści na siłę?