Lemat 1

W dowolnym momencie działania algorytmu nie powstanie cykl. (zakładamy że $c(e_i) \neq c(e_j)$, gdy $i \neq j$)

Dowód Załóżmy nie wprost, że podczas działania algorytmu w jakiejś spójnej składowej powstał cykl C.
Oznacza to, że powstał on wskutek połączenia w wierzchołki $v_0, v_1, \ldots, v_n$. Niech $e_0, e_1, \ldots, e_n$ będą kolejnymi krawędziami tworzącymi te wierzchołki. Z działania algorytmu dostajemy że $c(e_0) < c(e_1) < c(e_2) < \ldots < c(e_n) < c(e_0)$ czyli sprzeczność.

Lemat 2

W każdym etapie działania algorytmu otrzymujemy dla każdego supernierzchołka ~~w~~ MST.

D-d:

Sytuacja po ~~wykonaniu~~ dodaniu krawędzi.

1) Załóżmy, że istnieje taki supernierzchołek V_i że $V_i \neq MST(V(V_i))$. Niech to MST to T. Istnieje zatem e_i t.że $e_i \in E(V_i) \wedge e_i \notin E(T)$. Dodajmy to e_i do T. Wygenerujemy w ten sposób cykl.

E_i jest incydentna do wierzchołka v w tym cyklu i istnieje inna krawędź e_i' również incydentna do v. Ale skoro $e_i \in E(V_i)$ to $C(e_i) < C(e_i')$. Zatem po usunięciu e_i' uzyskalibyśmy drzewo rozpinające mniejsze niż T. Sprzeczność.

Widzimy zatem, że ~~ostatnia~~ zachodzi sytuacja, w której każdy supernierzchołek jest MST. Rozważmy zatem k-tą iterację algorytmu, która jest pierwszą taką, że po scaleniu otrzymujemy drzewo spinające, które nie jest MST.

Rozważmy dwa zbiory krawędzi E_1 i E_2, odpowiednio przed i po wykonaniu k-tej iteracji. Niech V_i to ~~taki~~ ten supernierzchołek t.że $E(V_i) \neq E(T)$, gdzie T to ~~MST~~ MST na V_i. Zatem istnieje $e_i \in E(V_i) \wedge e_i \notin E(T)$.

Zauważmy, że ta e_i musiała zostać dodana w k-tym kroku (bo jest to pierwszy moment, gdy nie mamy MST oraz inaczej byłoby to sprzeczne z 1))

Zatem, gdy dodamy e_i do ~~E(T)~~ E(T) otrzymamy cykl. A skoro e_i jest najmniejszą krawędzią incydentną do pewnego supernierzchołka to ta inna incydentna e_i' musi być większa, zatem ~~po~~ zastępując ją e_i uzyskalibyśmy mniejsze drzewo co pozwala, że T to MST.

z6)

Lemat pomocniczy

Dla dowolnego cyklu C w grafie i krawędzi $e \in C$, jeśli $c(e) > c(e_j)$ dla $e_j \in C$ t.że $j \neq i$, to e nie należy do żadnego MST.

D-d:

Załóżmy nie wprost, że $e \in C$ i $c(e)$ jest max w C oraz e należy do jakiegoś MST_1. Po usunięciu e z MST_1 utworzą się dwa poddrzewa T_1 i T_2. Skoro $e \in C$ to istnieje $e' \in C$ t.że końce e' należą do T_1 i T_2.

Wiemy, że $c(e)$ jest max, więc jeśli połączymy T_1 i T_2 przez pomocy e' uzyskamy MST' t.że $MST' < MST_1$, czyli sprzeczność.

Idea algorytmu:

Usuniemy z grafu krawędź e. Niech wierzchołki, które one łączyła to u i v, a nowy graf to G'.
Teraz możemy sprawdzić bfsem dodrzucym po krawędziach o wadze $< c(e)$ czy istnieje droga z u do v.

- Jeśli istnieje, to e leży na cyklu i jest najcięższą krawędzią na nim. Zatem z lematu wiemy, że e nie należy do żadnego MST.
- Jeśli nie istnieje droga między u i v. To e nie leży na cyklu, lub nie jest maksymalną krawędzią na żadnym cyklu. Zatem e należy do jakiegoś MST.

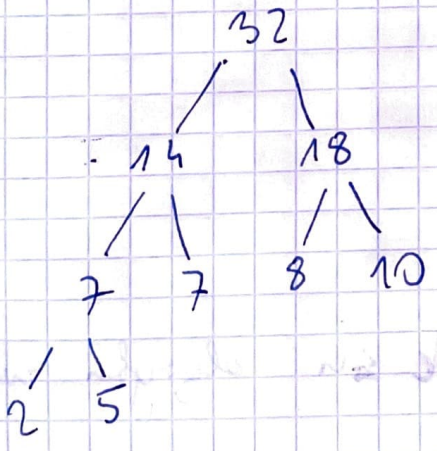
29)

Algorytm:

```
q := priority_queue                    (rosnąco)

for wi in W:
    q.push(wi)

while q.size() > 1:
    u, v = q.pop()
    q.push(Node(u.weight + v.weight, sons = (u, v)))

return q.pop()                         O(n log n)
```

Przykład:

$W = \{2, 5, 7, 8, 10\} \to \{7, 7, 8, 10\} \to \{8, 10, 14\} \to \{14, 18\} \to \{32\}$



D-d poprawności:

Lemat

Niech w_i i w_j będą najmniejszymi wagami w W. Istnieje optymalne rozwiązanie T, w którym w_i i w_j mają wspólnego ojca i są na najniższym poziomie drzewa.

D-d. lematu

W drzewie optymalnym T weźmy dwa wierzchołki u i v t. że mają wspólnego ojca i maksymalną głębokość.

Zamieniamy miejscami u z w_i i v z w_j. Mamy drzewo T'.

Pokażę, że $EL(T') \leq EL(T)$. Najpierw zauważmy, że w_i:

$$c(w_i) \leq c(u)$$

$$d_T(w_i) \leq d_T(u)$$

$$EL(T') = EL(T) - w_i \cdot d_T(w_i) - c(u) \cdot d_T(u) + w_i \cdot d_T(u) + c(u) d_T(w_i)$$
$$= EL(T) + w_i (d_T(u) - d_T(w_i)) + c(u)(d_T(w_i) - d_T(u)) =$$
$$= EL(T) + (d_T(u) - d_T(w_i))(w_i - c(u))$$

Wiemy, że

$$d_T(u) - d_T(w_i) \geq 0 \quad \wedge \quad w_i - c(u) \leq 0, \text{ więc}$$

$$EL(T') = EL(T) - r, \text{ gdzie } r \in \langle 0, \infty)$$

Analogicznie dla w_j □

D-d optymalności algorytmu:

Indukcja po liczbie liści:

Baza $n = 1$ ok

Krok: Załóżmy, że dla każdego $k \leq n$ algorytm zwraca poprawne rozwiązanie. Rozważmy n:

Niech w_i, w_j to najmniejsze wagi w W.

Niech $W' = W \setminus \{w_i, w_j\} \cup \{w_i + w_j\}$.

Niech P - rozwiązanie alg dla W, P' - rozw. alg. dla W',

T - drzewo optymalne dla W.

Z lematu wiemy, że w_i i w_j mają wspólnego ojca. Z T usuwamy w_i i w_j a wagę ojca ustalamy na $w_i + w_j$. Niech to drzewo to T'.

Mamy, że $EL(T') \geq EL(P')$ niech $D := d(v_i + v_j)$

$$EL(T) = EL(T') - D \cdot (v_i + v_j) + (D+1)(v_i + v_j) = EL(T') + v_i + v_j$$

$$EL(P) = EL(P') - D \cdot (v_i + v_j) + (D+1)(v_i + v_j) = EL(P') + v_i + v_j$$

$$\Downarrow$$

$$EL(P) \leq EL(T) \quad \square$$