Algorytm Kruskala - pseudokod

```
Kruskal(G,w)
// G=(V,E) - graf nieskierowany
// V - zbiór wierzchołków
// E - zbiór krawędzi
// w - funkcja wag dla krawędzi
// znajduje drzewo spinające graf G posiadające minimalną wagę

  dla każdego wierzchołka v
    Makeset(v)
  sortuj zbiór krawędzi E niemalejąco według wag krawędzi
  dla każdej krawędzi (u,v), niemalejąco według wag krawędzi
    ru = FindSet(u)
    rv = FindSet(v)
    if ru != rv
      wypisz krawędź (u,v) // nowa krawędź drzewa spinającego graf G
      Union(ru,rv)
```

Algorytm Kruskala - pseudokod dla nieco konkretniejszej, tablicowej reprezentacji grafu

Wierzchołki grafu są numerowane kolejnymi liczbami $0, 1, 2, \ldots,$ `ilW` $- 1$, gdzie `ilW` to ilość wierzchołków. Tworzymy rodzinę zbiorów zawierających te wierzchołki: `V` będzie tablicą wskaźników do węzłów reprezentujących elementy tych zbiorów, czyli `V[i]` to węzeł reprezentujący wierzchołek o numerze `i` .
Krawędzie grafu będą przechowywane w tablicy `E` której elementami są struktury posiadające pola `u,v,waga` , gdzie `u,v` to numery wierzchołków będących końcami krawędzi, a `waga` to waga krawędzi.

```
Kruskal(ilW, ilK, E)
// ilW - ilość wierzchołków; wierzchołki: 0,1,2,...,ilW-1
// ilK - ilość krawędzi
// E - tablica krawędzi
// V - tablica wskaźników do elementów zbiorów
  for i=0 to ilW-1
    V[i] = Makeset(i)
  sortuj tablcę E niemalejąco według wag krawędzi
  for i=0 to ilK-1
    ru = FindSet(V[E[i].u])
    rv = FindSet(V[E[i].v])
    if ru != rv
      wypisz krawędź E[i] //  nowa krawędź drzewa spinającego graf G
      Union(ru,rv)
```

Stwierdzenie Jeżeli w algorytmie Kruskala zastosujemy drzewiastą reprezentację zbiorów z rangą i kompresją ścieżek to algorytm ten ma złożoność pesymistyczną $\Theta(e \lg e)$, gdzie e to ilość krawędzi w grafie.

Dowód Przypomnijmy algorytm

```
Kruskal(G,w)
  dla każdego wierzchołka v
    Makeset(v)
  sortuj zbiór krawędzi E niemalejąco według wag krawędzi
  dla każdej krawędzi (u,v), niemalejąco według wag krawędzi
    ru = FindSet(u)
    rv = FindSet(v)
    if ru != rv
      wypisz krawędź (u,v) // nowa krawędź drzewa spinającego graf G
      Union(ru,rv)
```

Niech e oznacza ilość krawędzi w grafie a v ilosć wierzchołków.

Sortowanie krawędzi: $\Theta(e \lg e)$

Wypisywanie krawędzi, porównania `ru != rv` : $O(e)$

Pozostałe operacje to operacje na rodzinie zbiorów rozłącznych:
v operacji `Makeset` , $v + 3e$ wszystkich operacji:

$$O((v + 3e) \cdot \alpha((v + 3e, v)) = O(e)$$

Zatem czas pesymistyczny to $\Theta(e \lg e)$ (sortowanie krawędzi jest najbardziej czasochłonnym fragmentem algorytmu).

Stwierdzenie Algorytmie Kruskala znajduje drzewo spinające o minimalnej wadze.

Dowód Niech A oznacza zbiór dotychczas wypisanych krawędzi w trakcie działania pętli po krawędziach grafu. Pokazujemy, że niezmiennikiem pętli po krawędziach grafu jest: *istnieje drzewo spinające T o minimalnej wadze, którego krawędzie to wszystkie krawędzie ze zbioru A i pewne krawędzie dotąd nie zbadane.*

Gdy w pętli po krawędziach grafu bierzemy kolejną krawędź `(u,v)` to są trzy możliwości:

(1) `FindSet(u) == FindSet(v)`

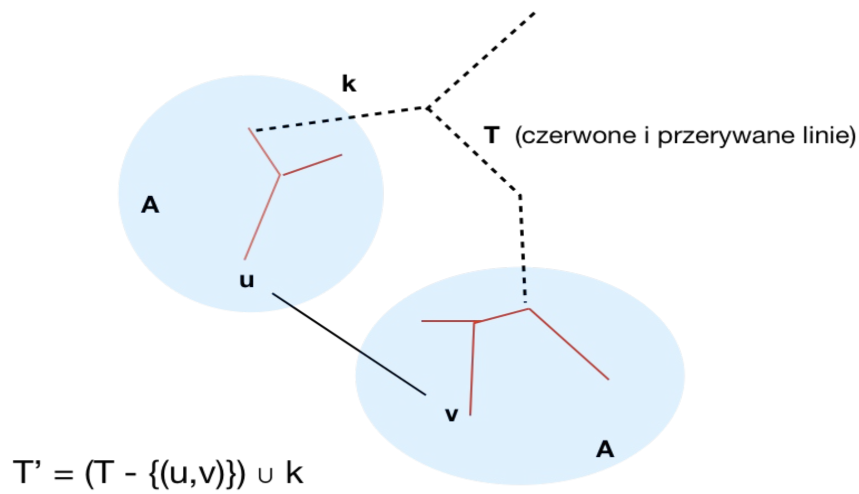
Takiej krawędzi nie wypisujemy, czyli nie dokładamy do zbioru A, ale też nie może być ona krawędzią drzewa spinającego bo łaczy wierzchołki między którymi już jest połączenie przez krawędzie z A, więc nie może należeć do drzewa T. Niezmiennik jest zachowany.

(2) `FindSet(u) != FindSet(v)` i `(u,v)` należy do T

Taką krawędź wypisujemy, czyli dokładamy do zbioru A a ponieważ należy ona do T to niezmiennik jest zachowany.

(3) `FindSet(u) != FindSet(v)` i `(u,v)` nie należy do T

Taką krawędź wypisujemy, czyli dokładamy do zbioru A. Sprawdzimy, że wówczas istnieje inne drzewo spinające T' o minimalnej wadze, które zawiera wszystkie krawędzie ze zbioru A i pewne krawędzie dotąd nie zbadane czyli dla którego zachowany jest niezmiennik.



`(u,v)` są połączone ścieżką w drzewie T więc musi być krawędź k na ścieżce w drzewie T, która wychodzi ze zbioru, do którego należy u. T' uzyskujemy usuwając z T krawędź k a dołączając krawędź pro(u,v). Waga `(u,v)` jest nie większa niż waga k, więc waga T' jest nie większa niż waga T. Zatem T' też jest drzewem o minimalnej wadze.

Uwaga Algorytm Kruskala to algorytm realizujący strategię zachłanną. Zachłanny wybór to wybieranie krawędzi w kolejności niemalejących wag, czyli zawsze bierzemy do sprawdzenia krawędź o najmniejszej wadze spośród krawędzi, które jeszcze mamy do sprawdzenia.