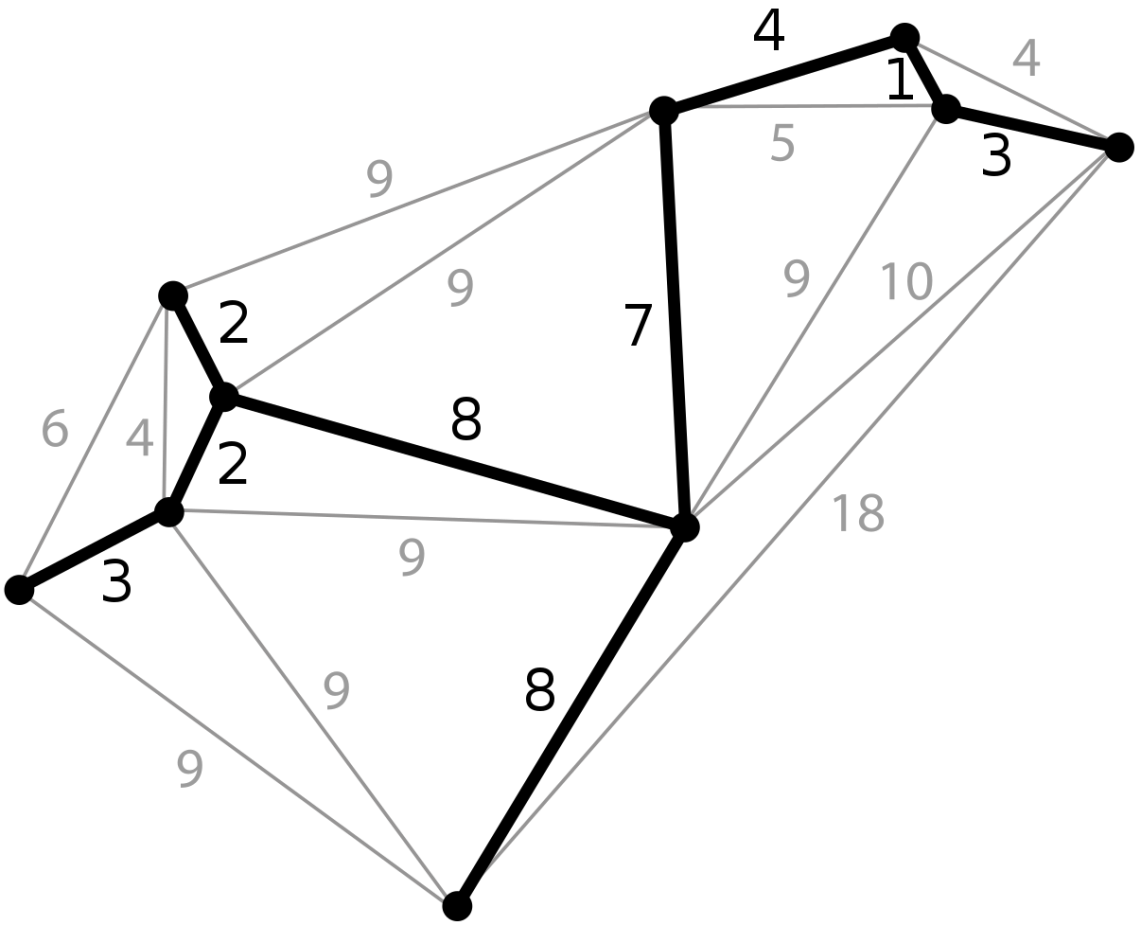
Minimalne drzewo rozpinające – minimum spanning tree (MST)

Rozważmy spójny (czyli taki, dla którego z każdego wierzchołka można dotrzeć do dowolnego innego) nieskierowany graf G o ważonych krawędziach. Minimalnym drzewem rozpinającym graf nazywamy taki podzbiór jego krawędzi, że nadal połączone są wszystkie wierzchołki grafu a suma wag krawędzi w minimalnym drzewie rozpinającym jest najmniejsza z możliwych. Poniżej przykład z Wikipedii: czarne grube krawędzie wchodzą w skład minimalnego drzewa rozpinającego, szare krawędzie nie wchodzą do tego drzewa.



Uwaga: może istnieć kilka minimalnych drzew rozpinających dany graf.

Do wyznaczenia MST można użyć np. algorytmu Prima albo algorytmu Kruskala. Najpierw zaimplementujemy algorytm Kruskala.

Algorytm Kruskala

1) Utwórzmy pusty zbiór X. Będzie on przechowywać krawędzie wchodzące w skład MST
2) Utwórzmy kolejkę priorytetową S, w której początkowo umieścimy wszystkie krawędzie wchodzące w skład grafu G. Priorytetem będzie waga krawędzi (im mniejsza tym wyższa pozycja w kolejce).

3) Dopóki kolejka nie jest pusta (pętla while) robimy co następuje:
   a) Usuwamy z kolejki krawędź z najniższą wagą (najwyższy priorytet). Dla ustalenia uwagi nazwijmy ją „u".
   b) Jeśli dodanie „u" do zbioru X nie utworzy cykli w X to dodajemy „u" do X (jeśli utworzy to nic nie robimy i przechodzimy do punktu a).

Uwaga: przez *„utworzenie cykli w X"* rozumiem to, że jeśli w grafie złożonym z wierzchołków G połączonych krawędziami ze zbioru X powstanie jakikolwiek cykl.

Sprawdzenie czy w grafie są cykle tradycyjnymi metodami jest z reguły kosztowne. Aby tego uniknąć warto zastosować tzw. strukturę zbiorów rozłącznych do przechowywania wierzchołków grafu. Struktura ta pozwala bardzo szybko (czas O(1)) sprawdzić, czy w danym momencie (tzn. dla danej „zawartości" zbioru X) istnieje połączenie między dwoma wierzchołkami. Powiedzmy, że badamy, czy do X można dodać krawędź „u" (łączącą wierzchołki o indeksach „a" oraz „b") bez tworzenia cykli. Jeśli „a" i „b" nie są połączone to można, ale jeśli już są połączone, to dodanie „u" utworzyłoby cykl, więc jest zabronione.

Struktura zbiorów rozłącznych

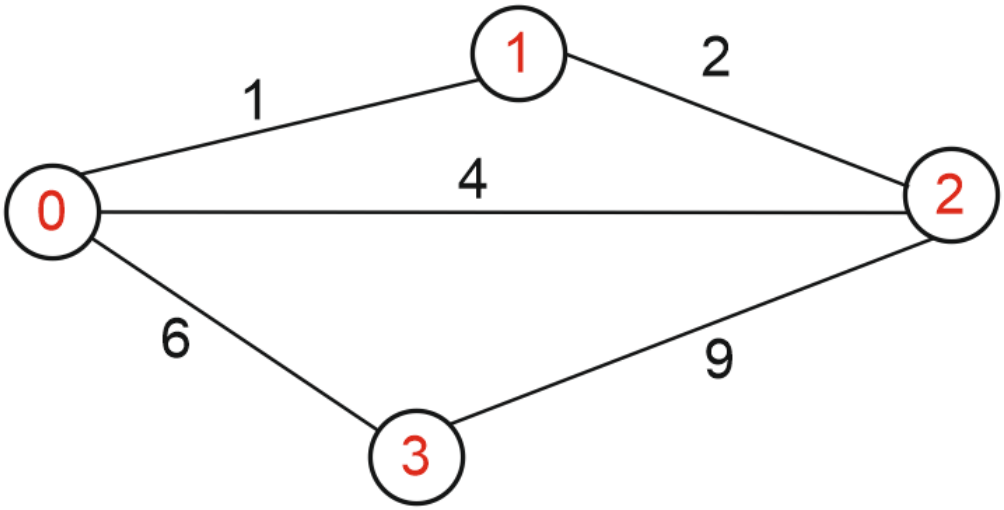
Tutaj opiszę chyba najprostszą implementację:

1) Mamy n wierzchołków grafu G (przy czym zakładam, że wierzchołki indeksujemy od 0, jeśli indeksowalibyśmy je od jedynki to w punkcie 3 trzeba by stosować Vec[a-1]==Vec[b-1]).
2) Utwórzmy wektor Vec, przechowujący liczby typu int, w którym początkowo przechowujemy liczby od 0 do n-1.
3) Aby sprawdzić czy dane wierzchołki (o indeksach „a" oraz" b") są połączone trzeba sprawdzić czy Vec[a]==Vec[b]. Jeśli tak, to wierzchołki są połączone i krawędzi nie można dodać do X. Jeśli nie są połączone (czyli krawędź dodać można), to trzeba uaktualnić strukturę zbiorów rozłącznych jak opisano w punkcie 4 poniżej.
4) Aby scalić wierzchołki o indeksach „a" oraz" b" (co trzeba uczynić podczas dodawania krawędzi „u" do zbioru X, przy czym indeksy „a" oraz „b" oznaczają wtedy końce tej krawędzi „u") należy wyznaczyć dwie zmienne min= minimum (Vec[a] oraz Vec[b]) oraz max= maksimum(Vec[a] oraz Vec[b]). Następnie należy przejść przez cały wektor Vec za pomocą pętli for (zmienna licznika pętli to i). Dla wszystkich elementów dla których Vec[i]==max trzeba dać Vec[i]==min.

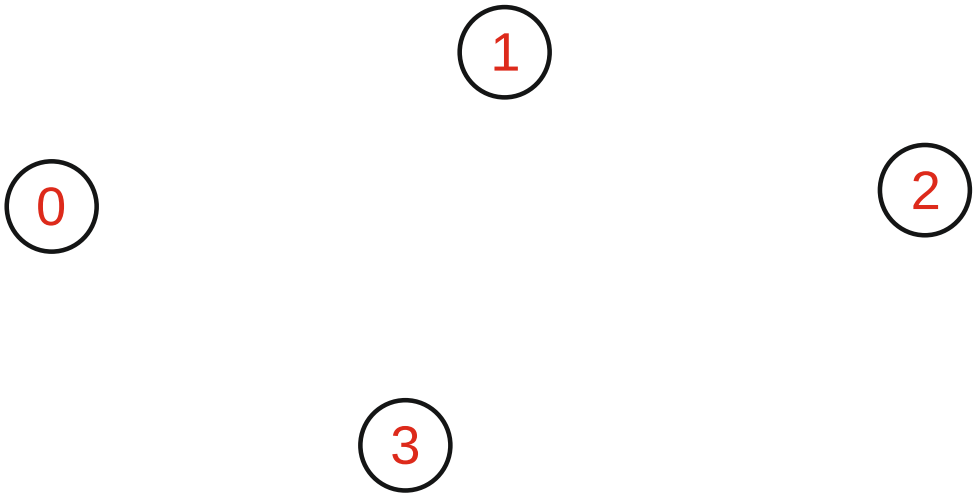
Złożoność obliczeniowa przy scalaniu wierzchołków to O(n) czyli naprawdę bardzo przyzwoicie. Na „upartego" można by to zaimplementować trochę lepiej, ale jest to kosztem zwiększenia skomplikowania struktury oraz wymagań pamięciowych więc to pominiemy.

Na następnej stronie pokażę prosty przykład tworzenia tej struktury (a także budowy MST) dla zadanego grafu.

Powiedzmy, że nasz graf (dla którego będziemy budować MST) wygląda tak:



Początkowo usuwamy wszystkie krawędzie zatem mamy 4 rozłączne wierzchołki

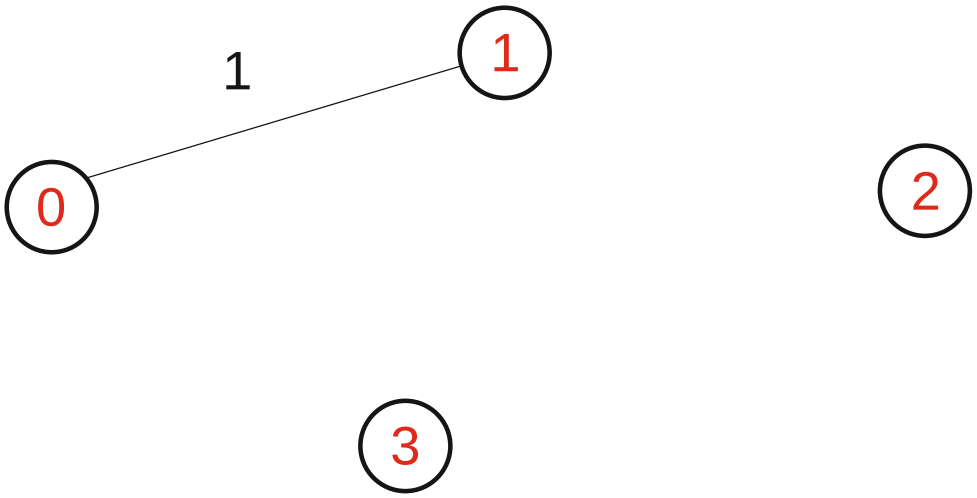


Tablica Vec wygląda tak (każdy wierzchołek stanowi osobną spójną składową grafu, oznaczoną numerem takim sam jak numer wierzchołka)

Indeks wierzchołka	0	1	2	3
Zawartość Vec	0	1	2	3

W naszym początkowym grafie najmniejszą wagę (1) miała krawędź łącząca wierzchołki 0 i 1. Zgodnie z tabelką możemy ją dodać (zawartość Vec[0] jest różna od Vec[1]).

Nasz aktualny graf wygląda więc tak:

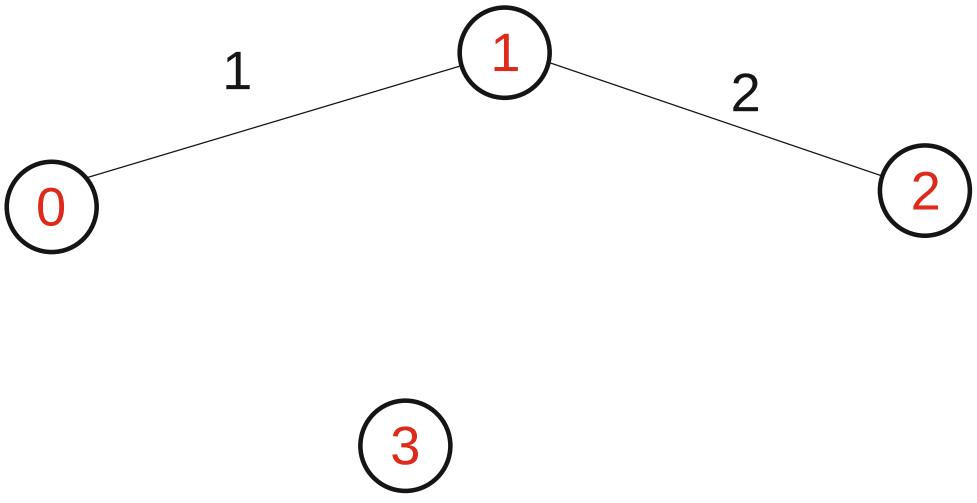


Teraz ten graf ma 3 spójne składowe. Musimy uaktualnić Vec (zmieniając zawartość Vec[1] na 0).

Indeks wierzchołka	0	1	2	3
Zawartość Vec	0	0	2	3

Teraz próbujemy dodać do MST krawędź o wadze 2 (łączącą 1 z 2)

Można to zrobić bo Vec[1] jest różne od Vec[2]. Po tym dostaniemy graf:

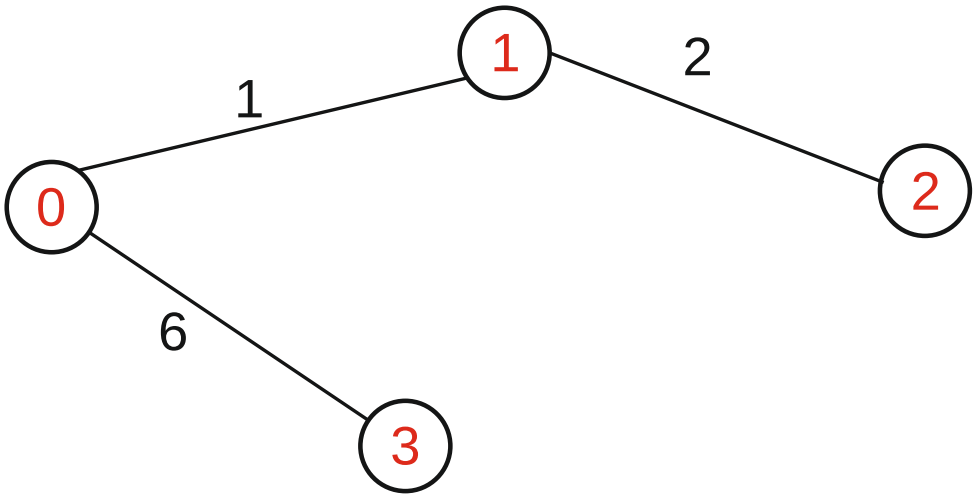


Ma on już tylko 2 spójne składowe. Uaktualniamy Vec, tabelka wygląda tak:

Indeks wierzchołka	0	1	2	3
Zawartość Vec	0	0	0	3

Próbujemy dodać kolejną krawędź z najniższą wagą jest to krawędź łącząca wierzchołki 0 i 2 o wadze 4. Ale – zgodnie z zawartością wektora Vec , oraz powyższej tabelki, wierzchołki te są już połączone! Dodanie tej krawędzi wytworzy cykl, więc nie dodajemy jej do MST.

Sprawdzamy kolejną krawędź łączącą 0 i 3 o wadze 6. Ta krawędź łączy wierzchołki z różnych spójnych składowych, więc ją można dodać:



Po uaktualnieniu wektora Vec mamy taką tabelkę

Indeks wierzchołka	0	1	2	3
Zawartość Vec	0	0	0	0

Próba dodania do MST krawędzi łączących 2 i 3 (o wadze 9) nie powiedzie się, bo 2 i 3 są już połączone. W początkowym grafie nie ma już żadnych krawędzi. Zatem graf z góry tej strony stanowi MST grafu początkowego.