Algorytm BIRCH

balanced iterative reducing and clustering using hierarchies

Cechy

- Algorytm ma na celu wstępne pogrupowanie danych na dużo małych i bardzo spójnych skupień

Cechy

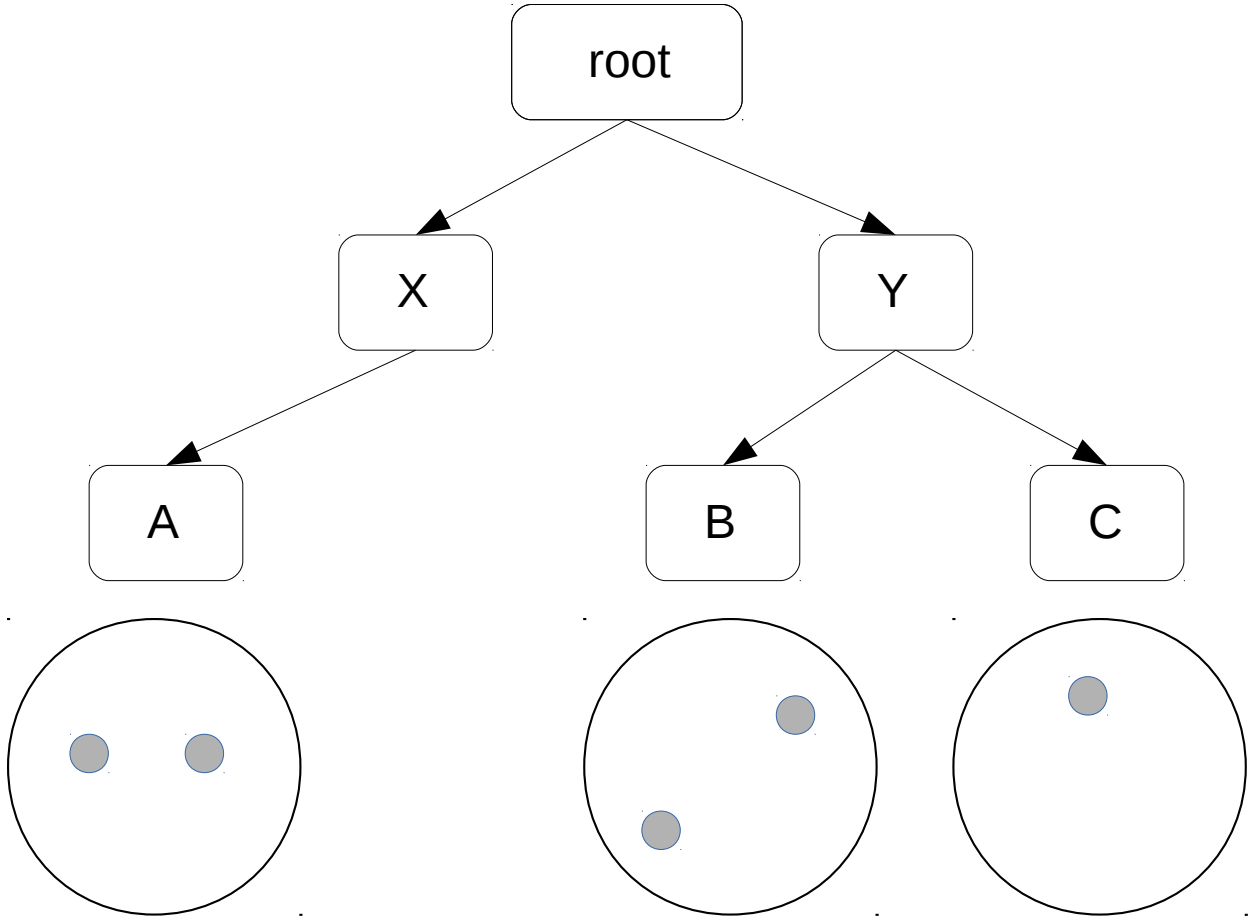
- Algorytm ma na celu wstępne pogrupowanie danych na dużo małych i bardzo jednorodnych skupień
- Następnie otrzymane klastry poddaje się grupowaniu (dowolnym algorytmem), gdzie nowymi obserwacjami są środki skupień

Cechy

- Algorytm ma na celu wstępne pogrupowanie danych na dużo małych i bardzo jednorodnych skupień
- Następnie otrzymane klastry poddaje się grupowaniu (dowolnym algorytmem), gdzie nowymi obserwacjami są środki skupień
- Algorytm jest bardzo wydajny – wymaga tylko jednego przejścia po danych

Mamy do pogrupowania obserwacje:

Algorytm polega na zbudowaniu drzewa wyznaczajcego klastry.



Parametry

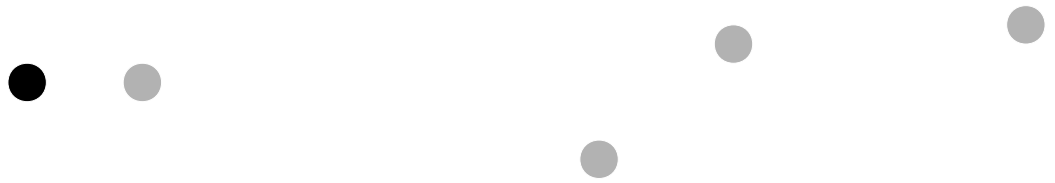
- T (*treshold*) – ograniczenie na promień wynikowych klastrów – będą one konstruowane tak, że promień nie przekroczy tej wartości
  - promień – średnia odległość punktów od środka (średniej arytmetycznej z obserwacji)

Parametry

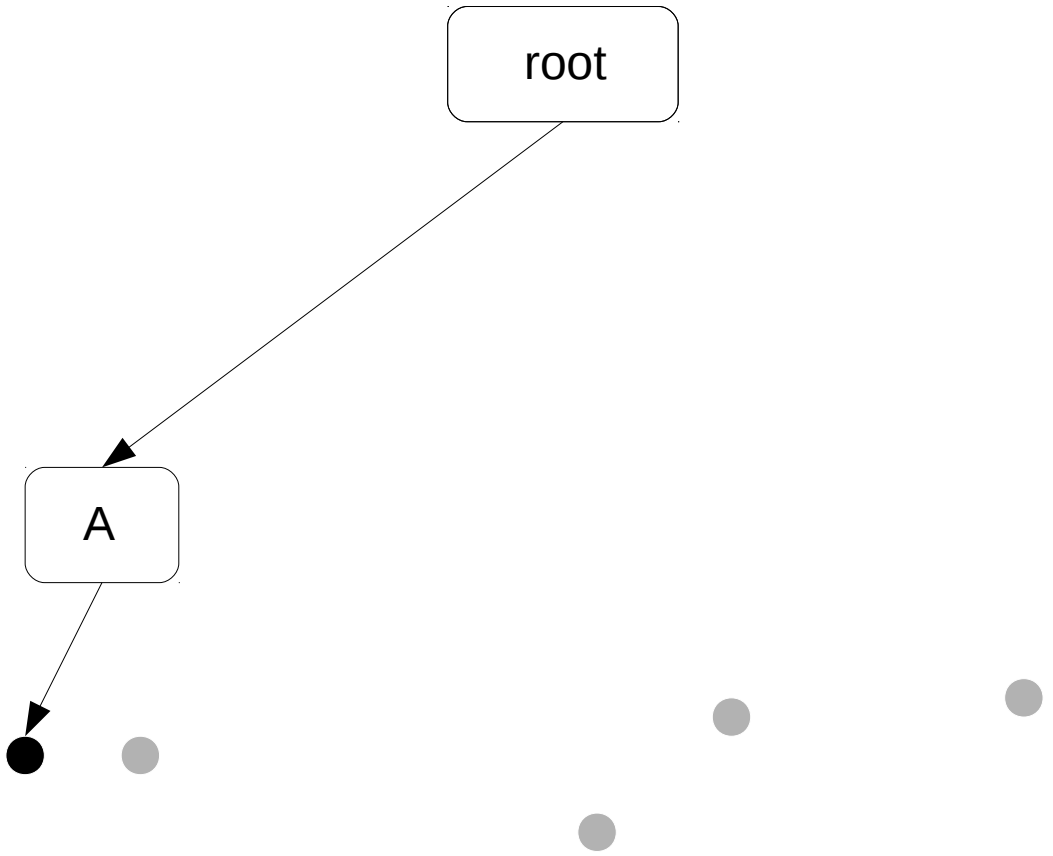
- T (*treshold*) – ograniczenie na promień wynikowych klastrów – będą one konstruowane tak, że promień nie przekroczy tej wartości
  - promień – średnia odległość punktów od środka (średniej arytmetycznej z obserwacji)
- B (*branching factor*) – maksymalna dopuszczalna liczba gałęzi wychodzących z jednego węzła

Przyjmijmy B = 2.

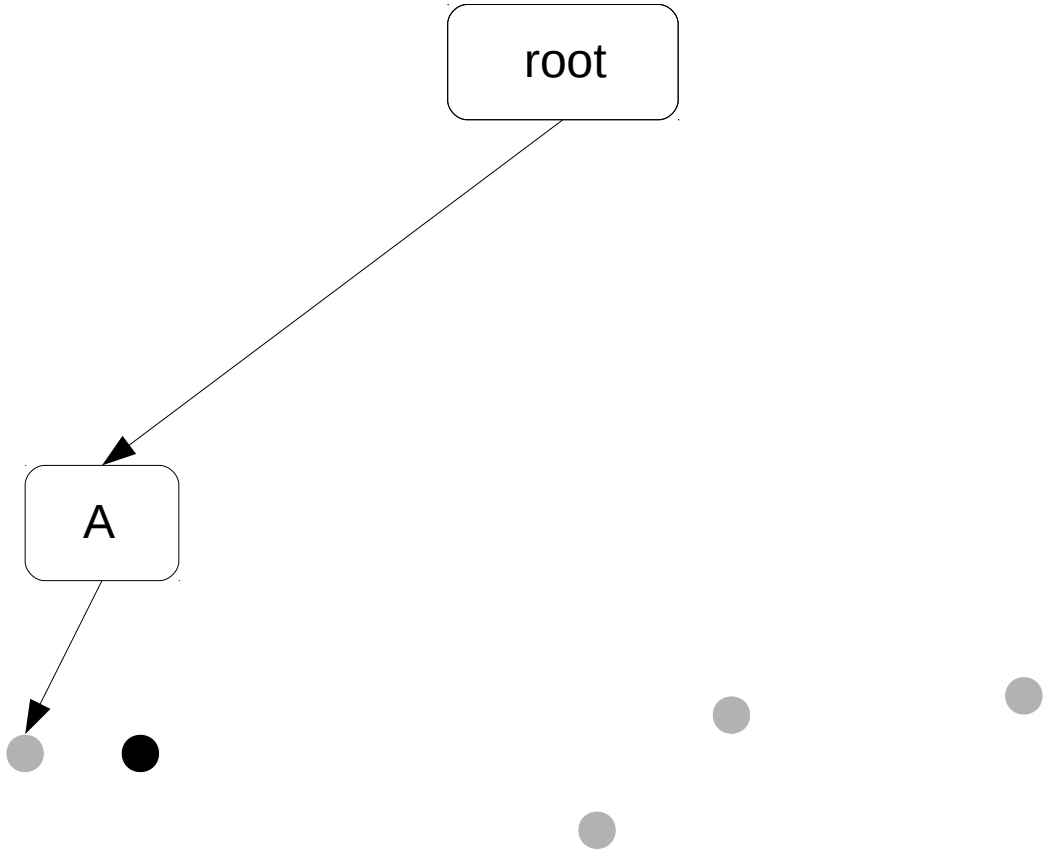
Algorytm przechodzi jednokrotnie po zbiorze danych. Zaczynamy od pierwszej obserwacji.



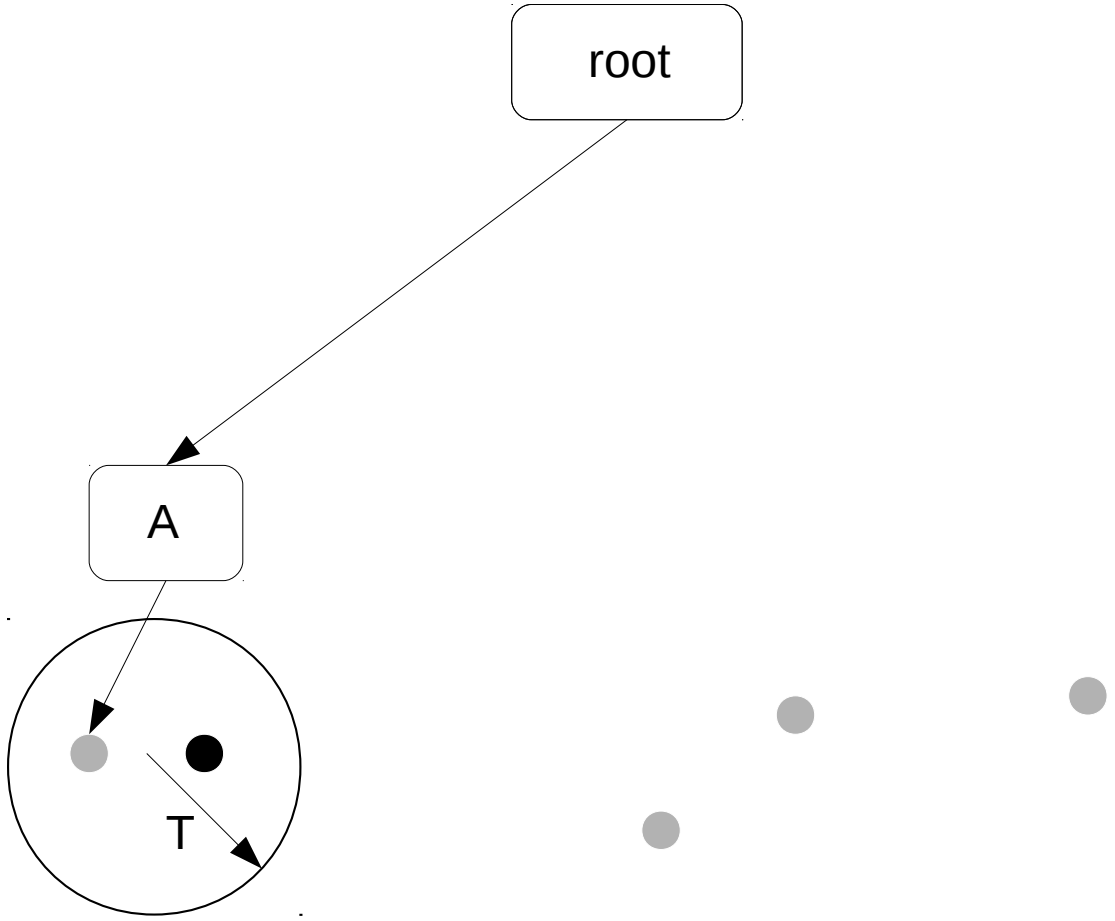
Inicjujemy korzeń drzewa oraz pierwszy klaster - A.



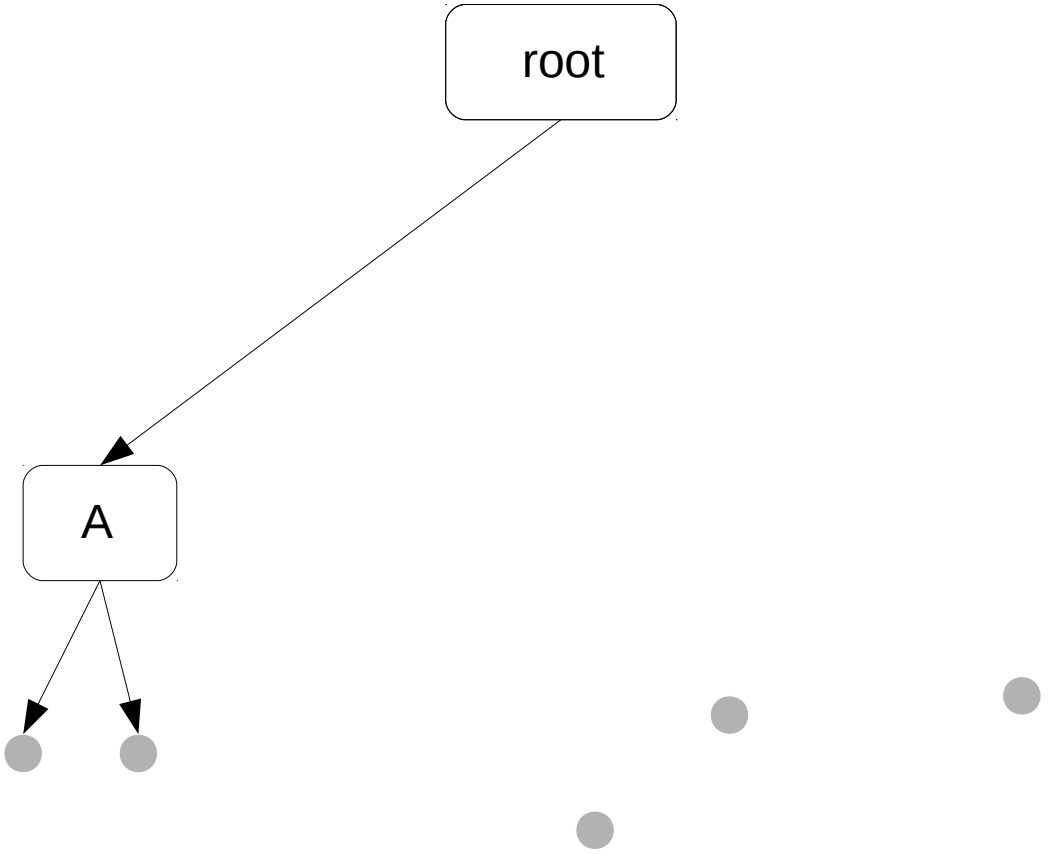
Rozpatrujemy drugi punkt. Idąc po drzewie od korzenia, szukamy najbliższego klastra.



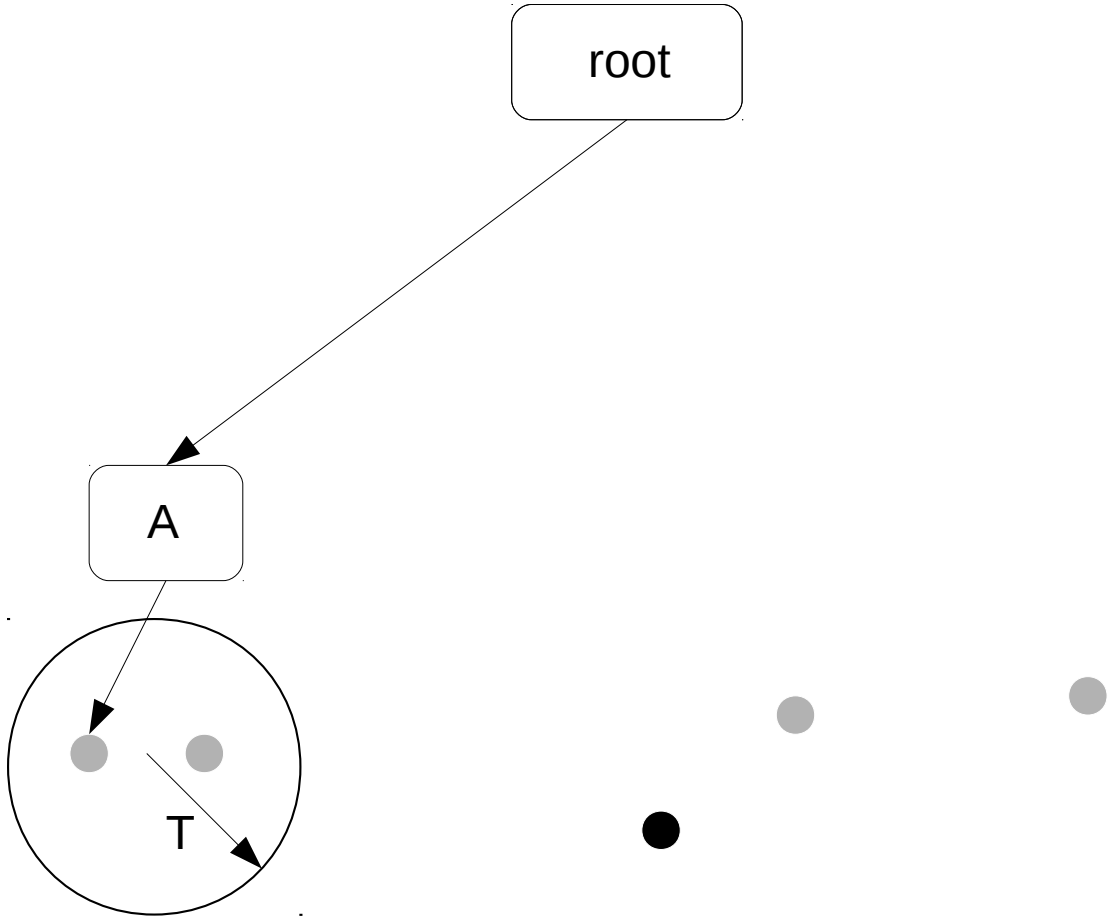
Jest to A (bo jest jedyny). Sprawdzamy czy promień klastra powstałego z połączenia A i nowej obserwacji nie przekroczy T.



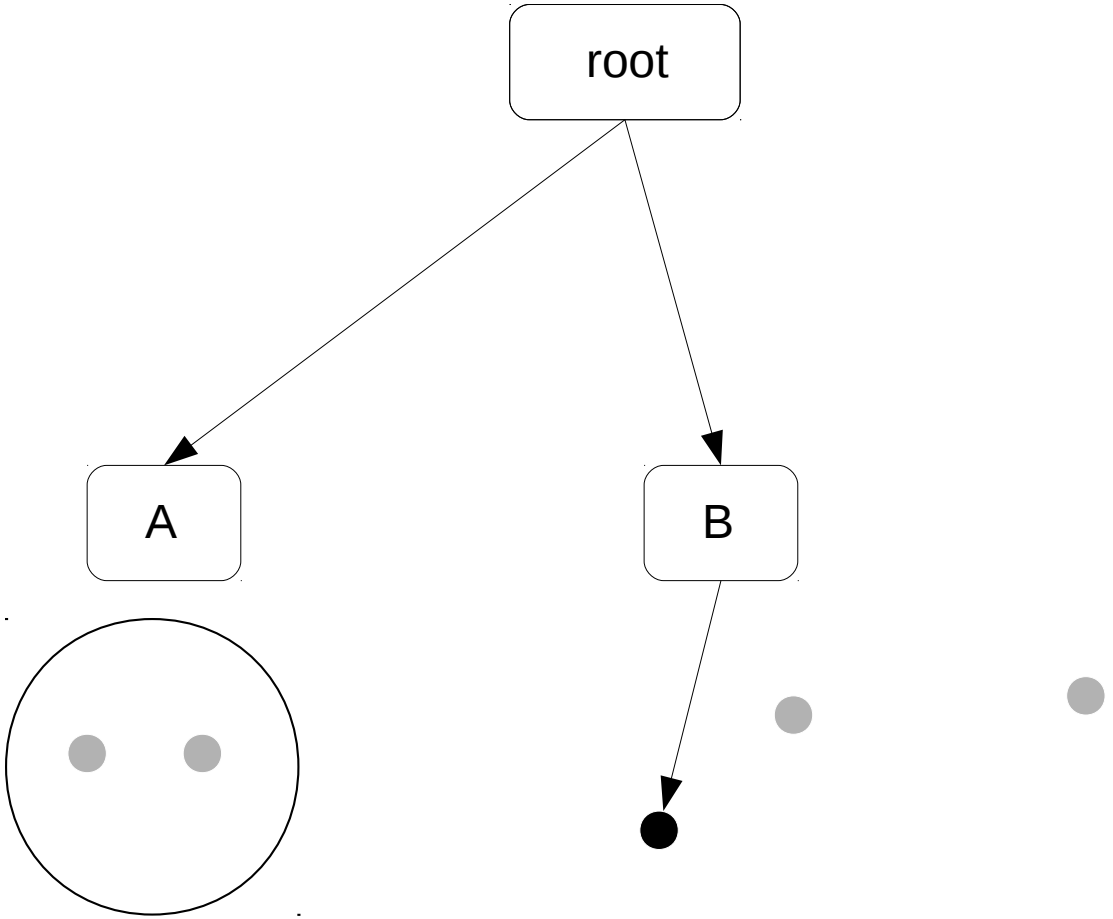
Nie przekroczy. Zatem dołączamy obserwację do A.



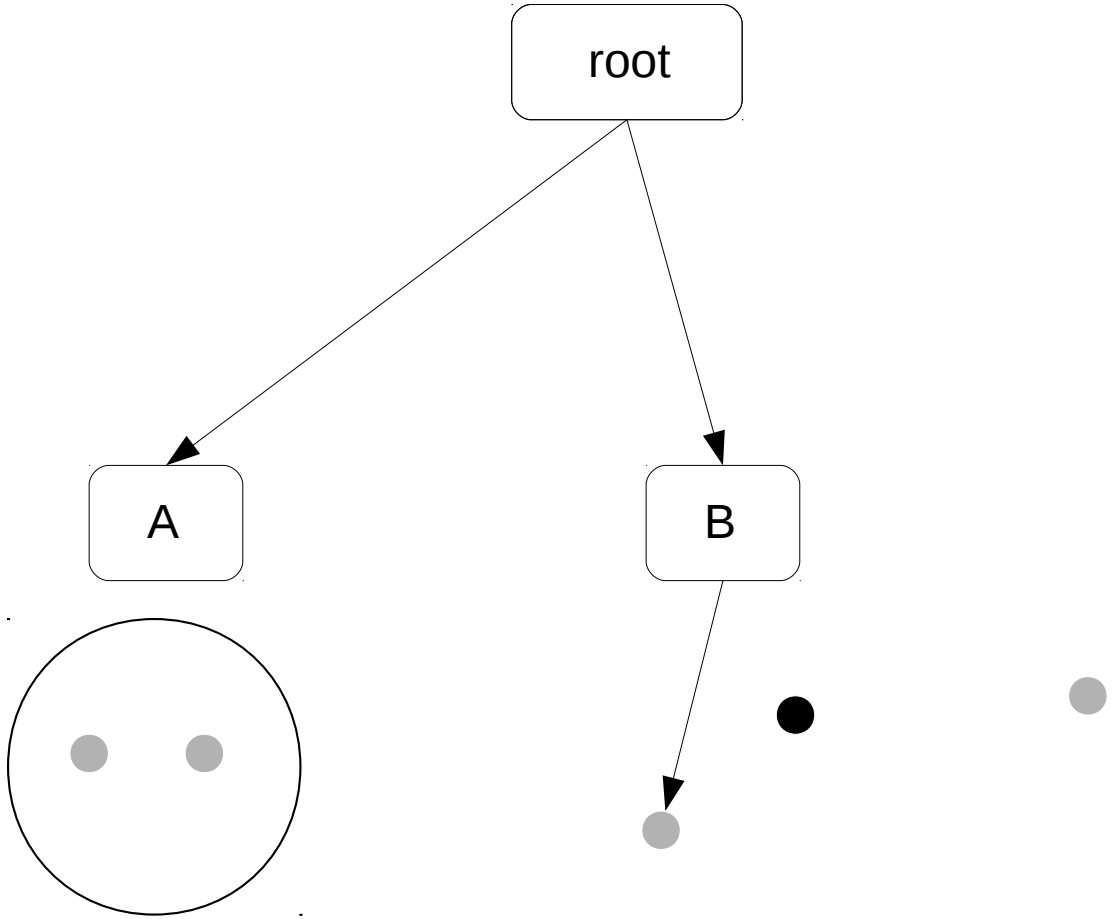
Rozpatrujemy kolejny punkt. Szukamy najbliższego klastra i sprawdzamy czy jest wystarczająco blisko (czy promień po dołączeniu nie przekroczy T).



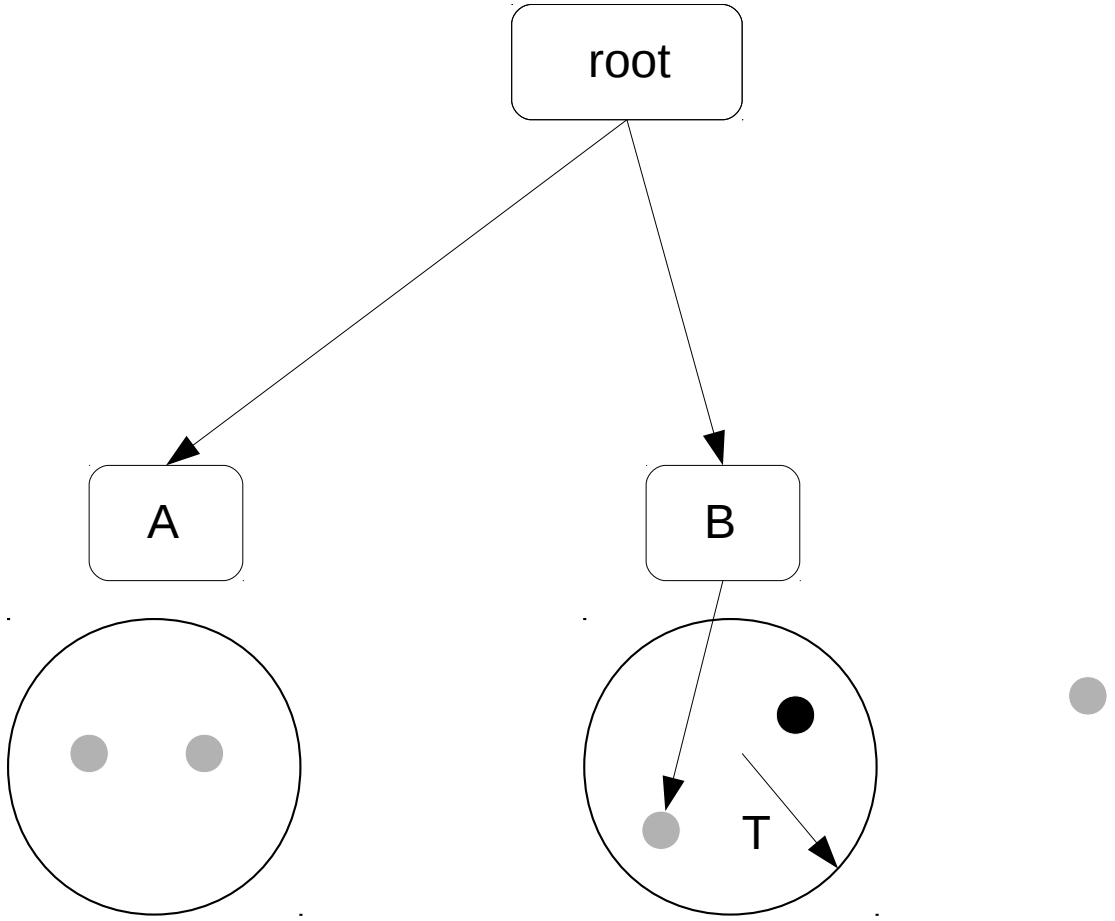
Nie jest – zatem tworzymy nowy klaster.



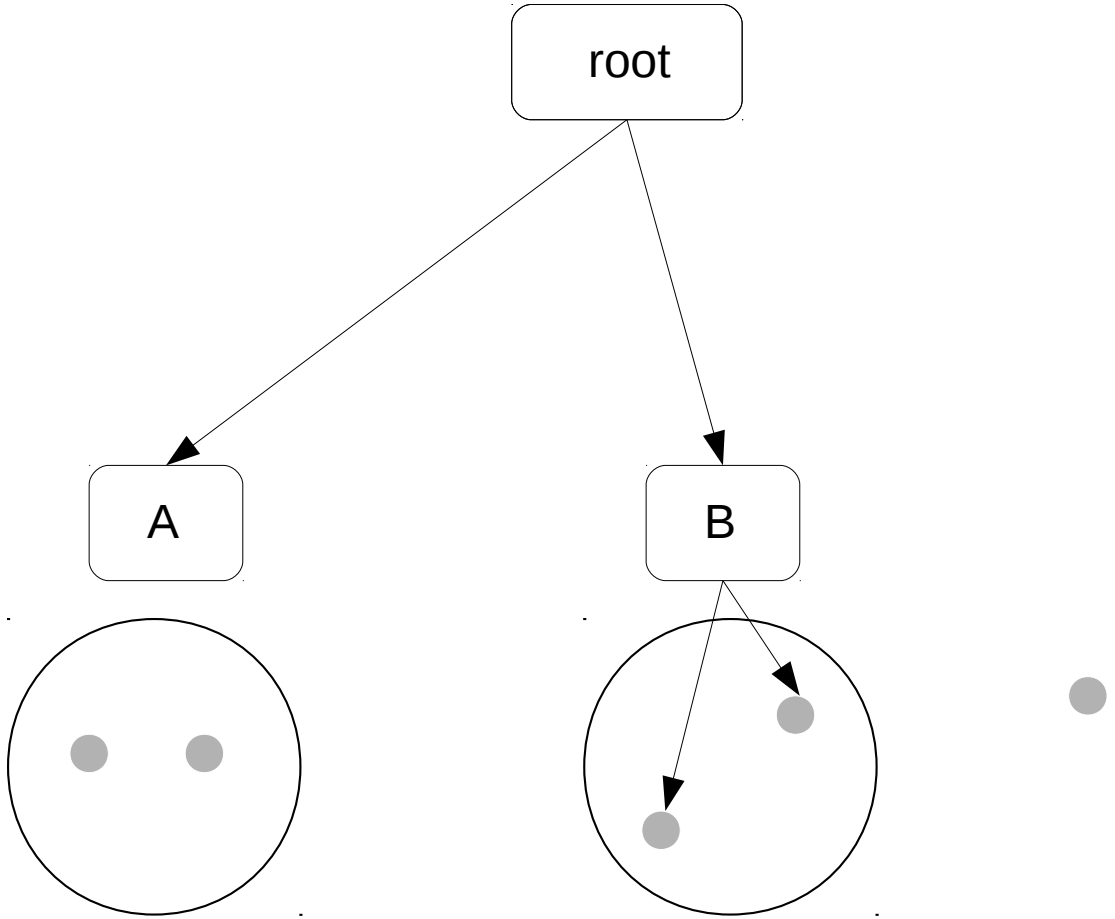
Rozpatrujemy kolejny punkt. Szukamy najbliższego klastra – jest to B.



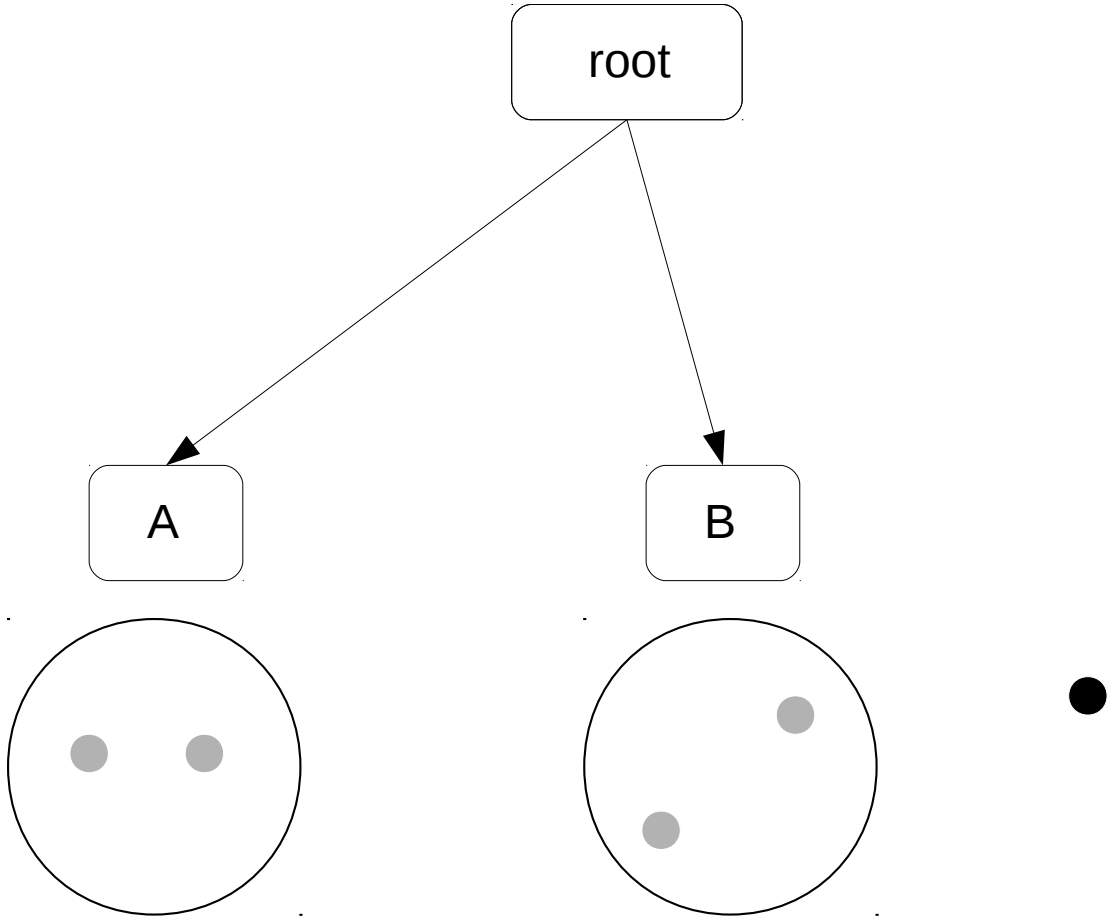
Sprawdzamy czy jest wystarczająco blisko.



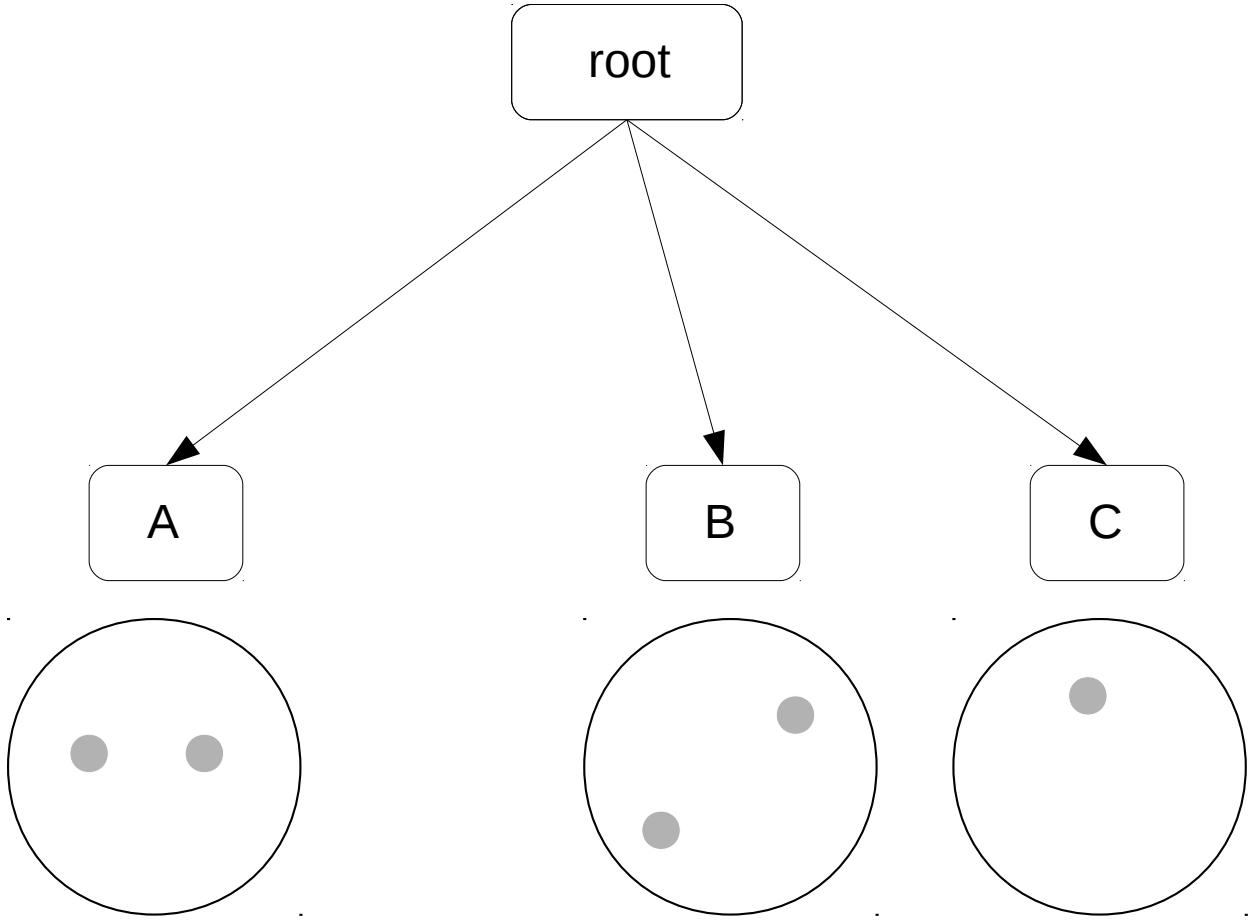
Tak – zatem dołączmy obserwację do B.



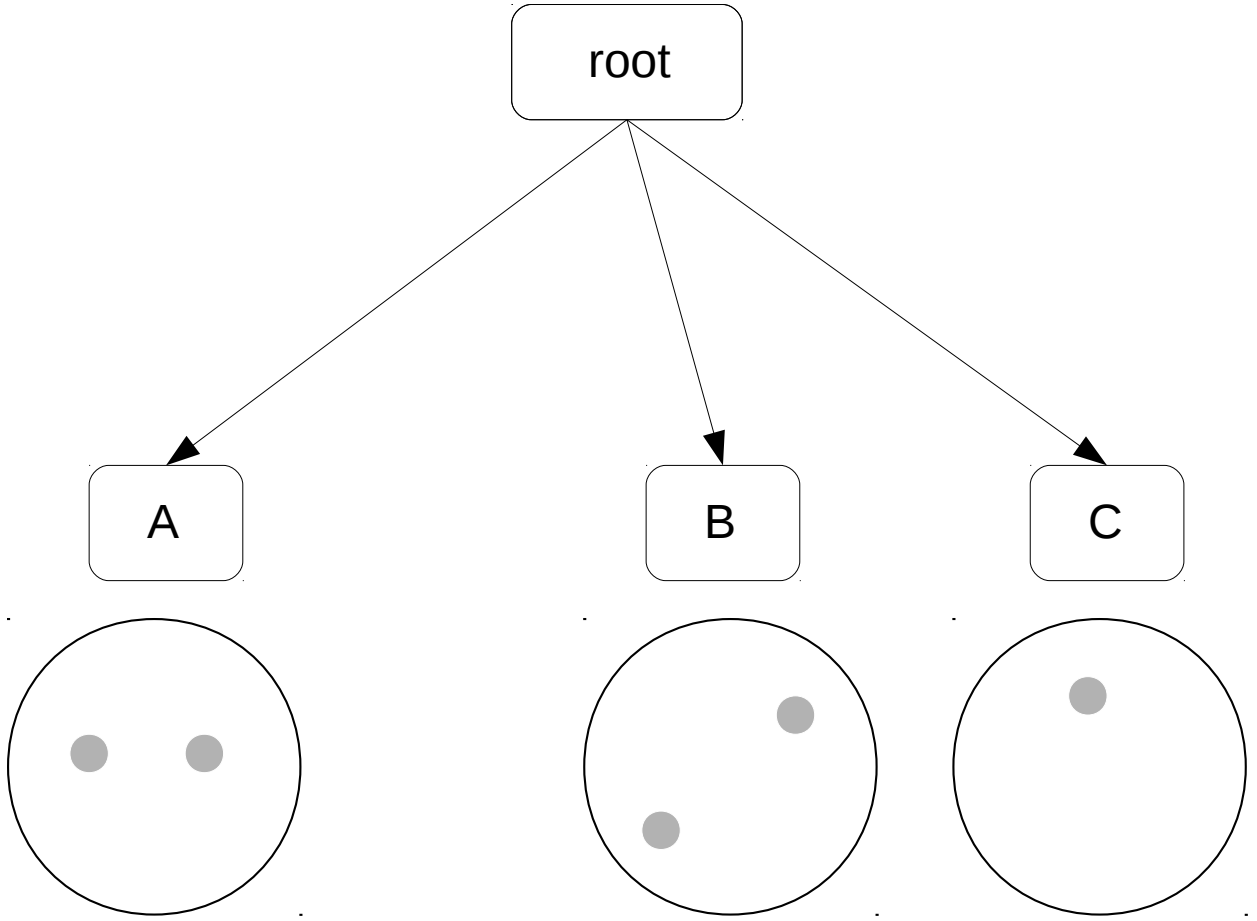
Rozpatrujemy kolejny punkt.



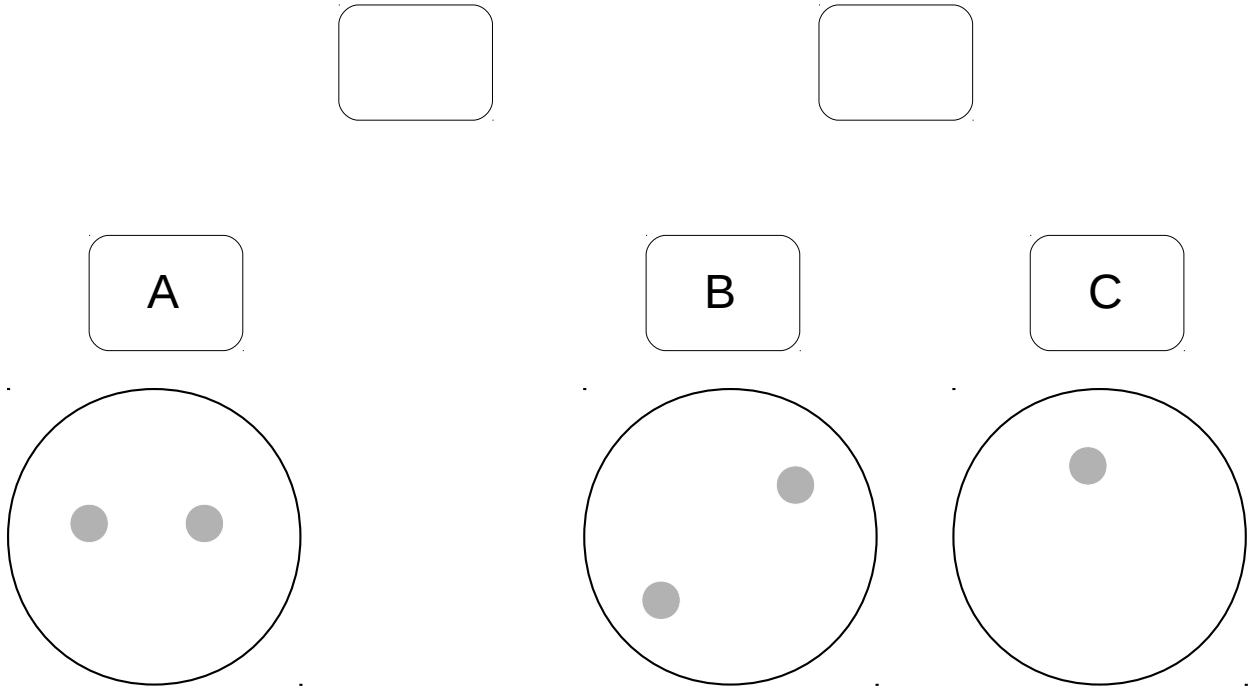
Najbliższy klaster (B) jest za daleko – tworzymy nowy klaster.



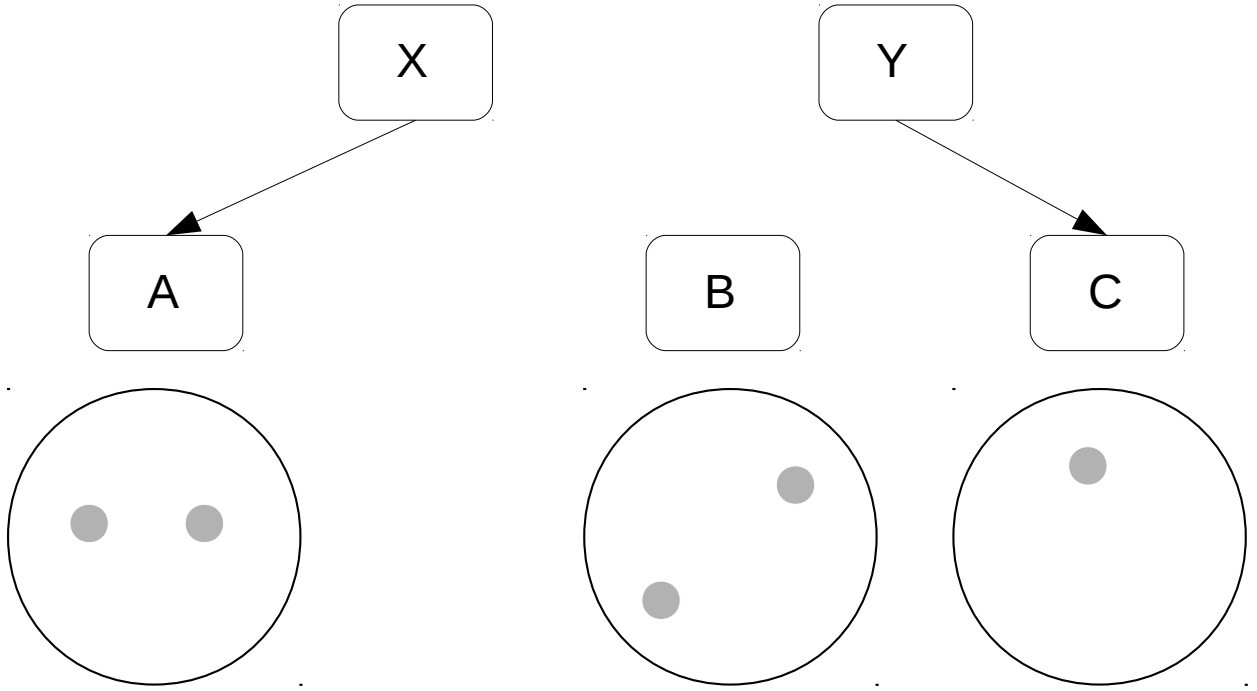
Ale… Z wierzchołka “root” wychodzą trzy gałęzie – przekroczyliśmy B = 2. Trzeba przebudować drzewo.



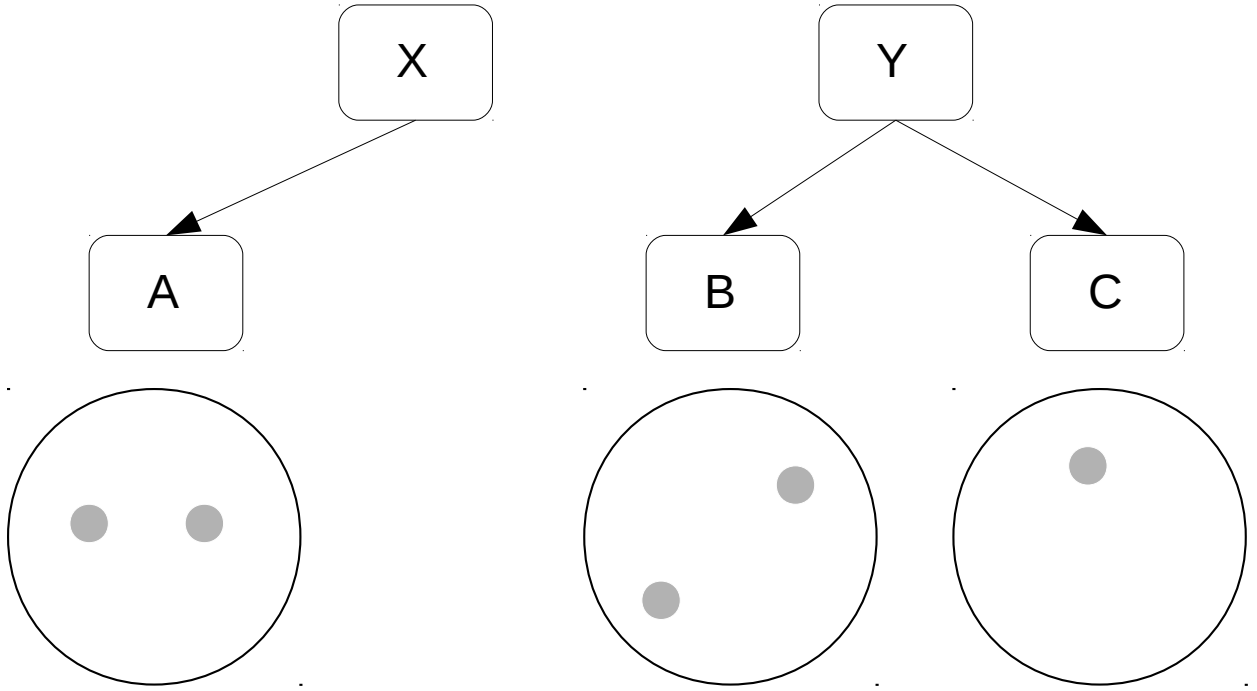
Rozbijamy wierzchołek, z którego wychodzi za dużo gałęzi na dwa i odbudowujemy drzewo od dołu.



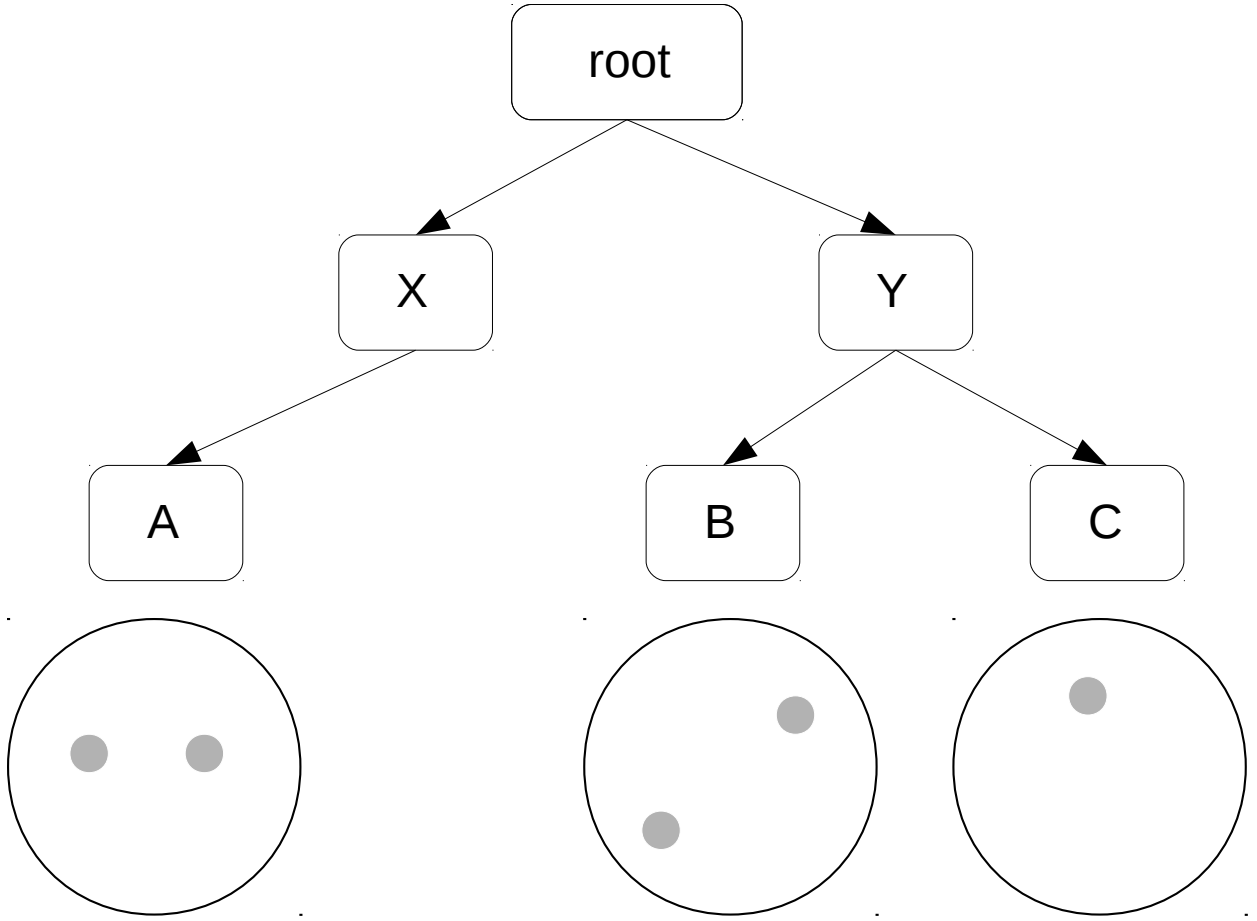
Bierzemy dwa najbardziej oddalone od siebie klastry (tutaj A i C) I przyporządkowujemy je do osobnych węzłów (subklastrów).



Pozostałe klastry (tutaj B) przyporządkowujemy do najbliższych subklastrów – tutaj B do Y.



I odbudowujemy drzewo do korzenia (aktualizujemy strukturę nad miejscem rozpadu).



CF (Clustering Feature) Tree

Każdy subklaster (węzeł w drzewie) jest opisany trzema wielkościami:

- N – liczba punktów w subklastrze
- LS – suma punktów - wektor sum po współrzędnych
- SS – suma kwadratów punktów – wektor sum kwadratów po współrzędnych

Na podstawie tych wartości wyliczamy wielkości potrzebne do przydzielenia nowego punktu do klastra i aktualizacji struktury drzewa.

To decyduje o niskiej złożoności algorytmu.