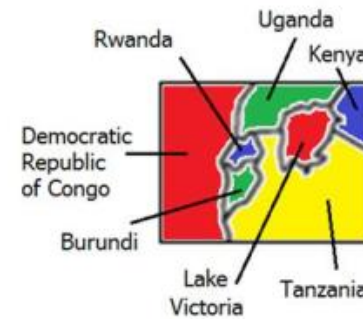


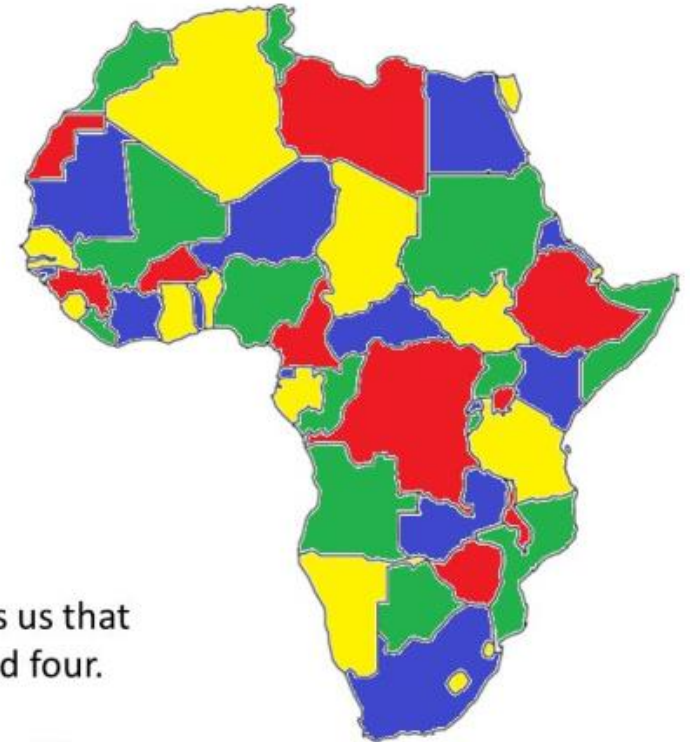
# Kolorowanie grafu

- Przypisanie wierzchołkom (ewentualnie krawędziom) grafu kolorów (reprezentują więzy)
- Historia: kolorowanie państw na mapie w taki sposób by żadne sąsiednie państwa nie były pokolorowane w ten sam sposób.
- Liczba chromatyczna.
- $PG(k)$  (wielomian chromatyczny grafu  $G$ )
- Problem kolorowania wierzchołków jest NP-trudny.

It turns out that  
FOUR will do.

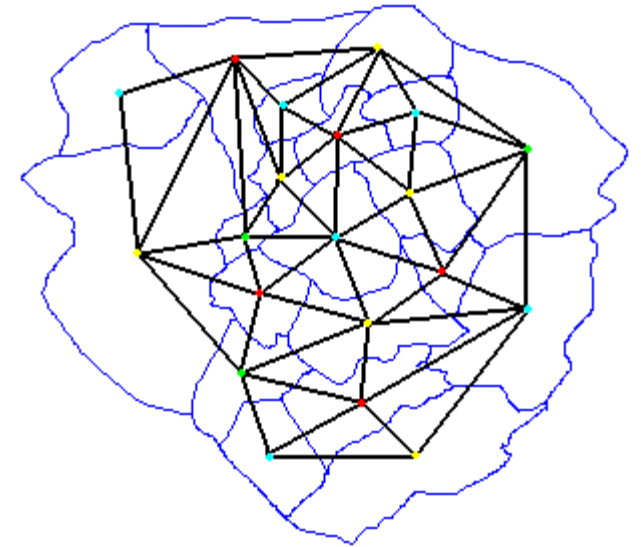


This section shows us that  
we certainly need four.



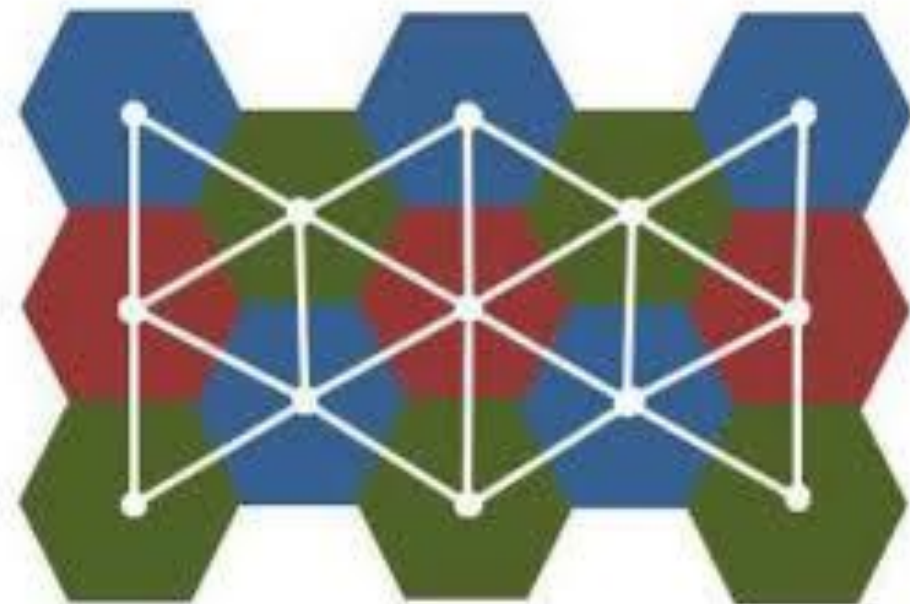
# Twierdzenie o czterech barwach

- dla każdego skończonego grafu planarnego  $(V, E)$  istnieje funkcja  $k: V \rightarrow \{k_1, k_2, k_3, k_4\}$  taka że  $\forall_{\{v_1, v_2\} \in E} (k(v_1) \neq k(v_2))$  czyli możliwe jest przypisanie każdemu z jego wierzchołków jednej z czterech liczb 1, 2, 3 i 4 w taki sposób, aby żadne sąsiednie wierzchołki nie miały przyporządkowanej tej samej liczby. Jest to jeden z najśłynniejszych problemów matematycznych.



# Zastosowania

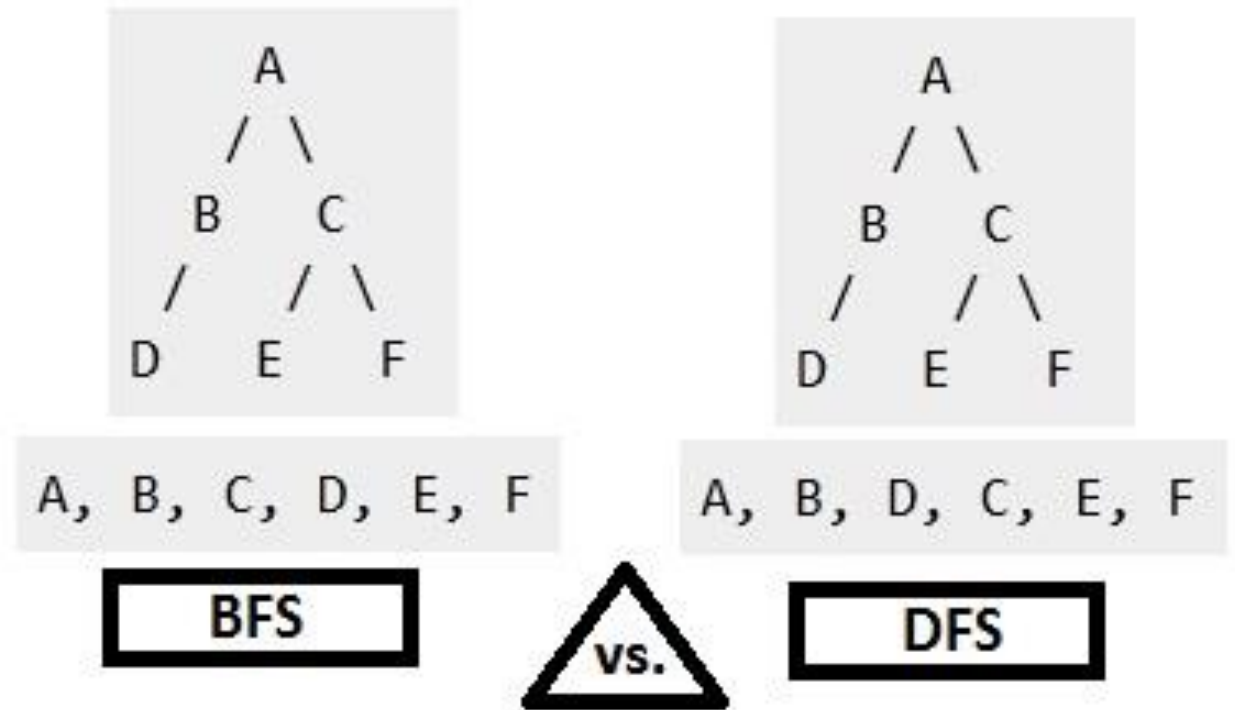
- Sudoku (AI)
- Planowanie (w tym dyspozytor)
- Przydzielanie częstotliwości fali radiowej
- Alokacja rejstrów



# Przeszukiwanie w głąb, wszerz

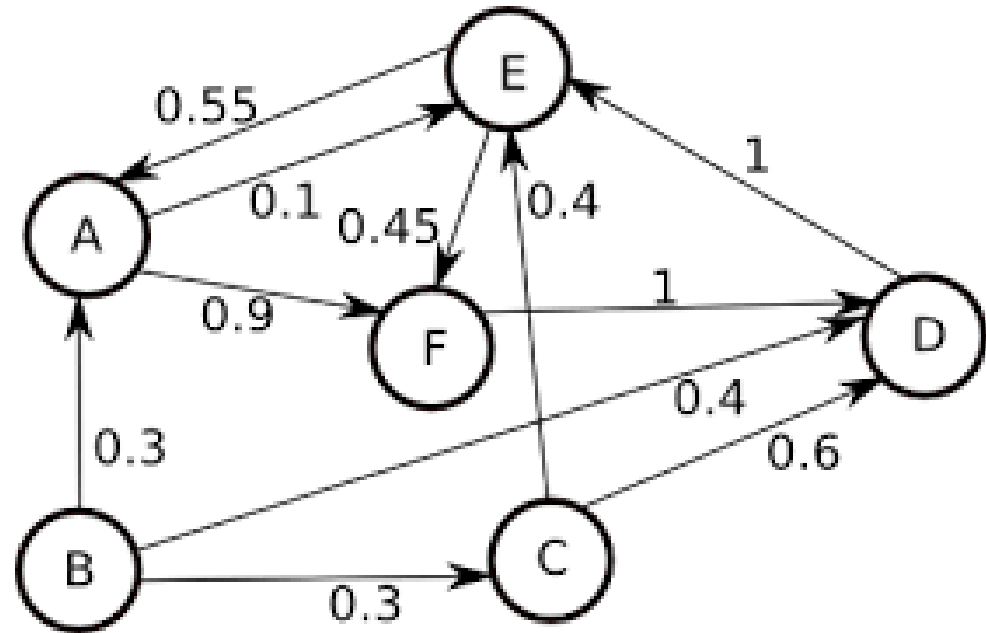
- DFS (stos)
- BFS (kolejka)

1. Dodaj korzeń do kolejki.
2. Pop() i Push(wszystkie dzieci)



# Wyszukiwanie najkrótszych ścieżek

- Jak dojść z A do B?
- Najpopularniejsze algorytmy:
  - Dijkstra
  - Prim
  - Kruskal

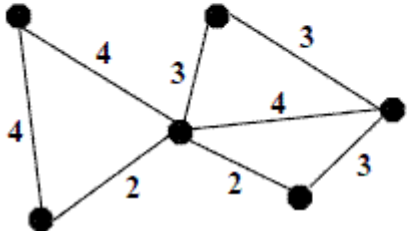

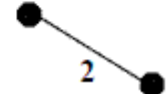
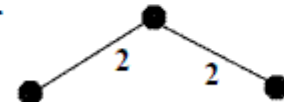
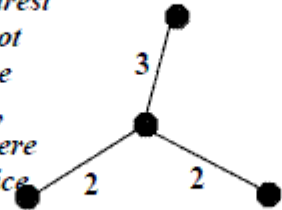
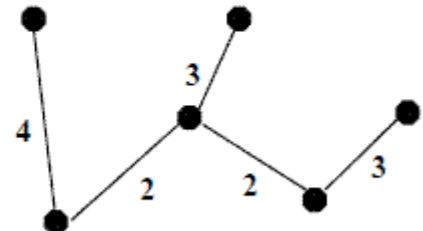


# Algorytm Dijkstry

1. Stwórz tablicę nieodwiedzonych wierzchołków, zainicjalizuj ją wszystkimi wierzchołkami poza startowym.
2. Z pierwszego wierzchołka przejdź do następnego dla którego krawędź ma najniższą wagę. Usuń z tablicy nieodwiedzonych wierzchołków ten wierzchołek.
3. Znajdź następną ścieżkę o najniższej wadze prowadzącą z wierzchołka startowego do następnego nieodwiedzonego wierzchołka.
4. Powtarzaj, aż tablica nieodwiedzonych wierzchołków będzie pusta.

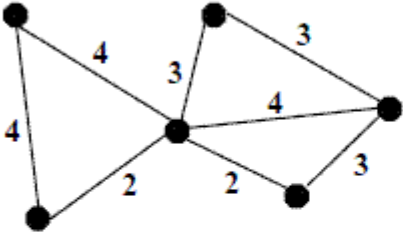
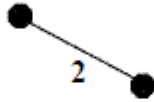
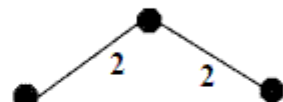
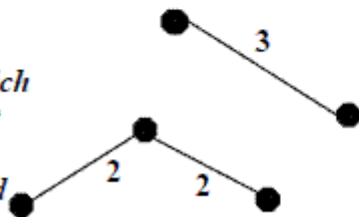
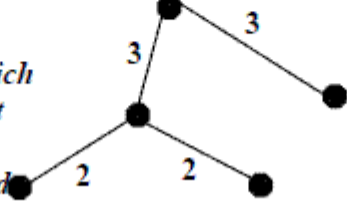
# Algorytm Prima

## Prim's Algorithm

<p>1 <i>Given a network.....</i></p> 	<p>2 <i>Choose a vertex</i></p> 	<p>3 <i>Choose the shortest edge from this vertex.</i></p> 
<p>4 <i>Choose the nearest vertex not yet in the solution.</i></p> 	<p>5 <i>Choose the next nearest vertex not yet in the solution, when there is a choice choose either.</i></p> 	<p>6 <i>Repeat until you have a minimal spanning tree.</i></p> 

# Algorytm Kruskala

## Kruskal's Algorithm

<p>1 <i>Given a network.....</i></p> 	<p>2 <i>Choose the shortest edge (if there is more than one, choose any of the shortest).....</i></p> 	<p>3 <i>Choose the next shortest edge and add it.....</i></p> 
<p>4 <i>Choose the next shortest edge which wouldn't create a cycle and add it.</i></p> 	<p>5 <i>Choose the next shortest edge which wouldn't create a cycle and add it.</i></p> 	<p>6 <i>Repeat until you have a minimal spanning tree.</i></p> 