✖ ALGORYTMY I STRUKTURY DANYCH – GRAF

✖ Operacje na grafie

+ Znajdywanie najkrótszej ścieżki w grafie – **algorytm Dijkstry**
  - × **dane**: graf G i węzeł s (źródło)
  - × **wynik**: waga najkrótszych ścieżek z s do każdego węzła v osiągalnego z s
  - × **założenie**: w(u, v) ≥ 0 dla każdego u, v ∈ V

Dijkstra(G, s)

for(each u ∈ V)
  d[u]← ∞
  π[u]← NIL
d[s] ← 0
Q ← V
while(Q ≠ 0)
  u← dequeueMin(Q)
  for(each v ∈ A[u])
    relax(u, v)

inicjalizacja

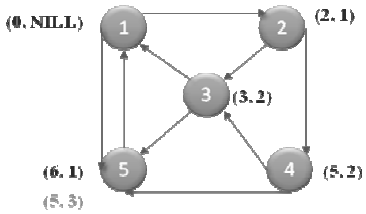
relax(u, v)

if(d[v]>d[u] + w(u, v))
  d[v]← d[u] + w(u, v)
  π[v]← u

oznaczenia:

+ Q – kolejka priorytetowa
+ w(u, v) – waga krawędzi (u, v)
+ d[u] – oszacowanie długości ścieżki od s do u

+ **złożoność**: $O(n^2 + m)$ ($O(n\log(n) + m)$ – dla implementacji Q w postaci kopca Fibonacciego)
+ przykład:



	1	2	3	4	5
1	0	2	-	-	6
2	-	0	1	3	-
3	4	-	0	-	2
4	-	-	1	0	1
5	3	-	-	-	0

+ Znajdywanie najkrótszej ścieżki w grafie – **algorytm Floyda - Warshalla**
  - × programowanie dynamiczne
  - × **dane**: graf G
  - × **wynik**: wagi najkrótszych ścieżek pomiędzy każdą parą węzłów

Floyd-Warshall (W)

n ← liczba wierszy W
$D^{(0)} \leftarrow W$
for (k ← 1 to n)
  for (i ← 1 to n)
    for (j ← 1 to n)
      $d^{(k)}_{ij} \leftarrow min(d^{(k-1)}_{ij}, d^{(k-1)}_{ik} + d^{(k-1)}_{kj})$
ret$(D^{(n)})$

× **oznaczenia:**
  - × $d^{(k)}_{ij}$ – waga najkrótszej ścieżki z i do j, której wszystkie węzły wewnętrzne należą do {1, …, k}
  - × $D^{(n)} = (d^{(n)}_{ij})$ jest **rozwiązaniem** → $d^{(n)}_{ij} = \delta(i, j)$ dla każdego i, j ∈ V

✖ ALGORYTMY I STRUKTURY DANYCH – GRAF

✖ Operacje na grafie

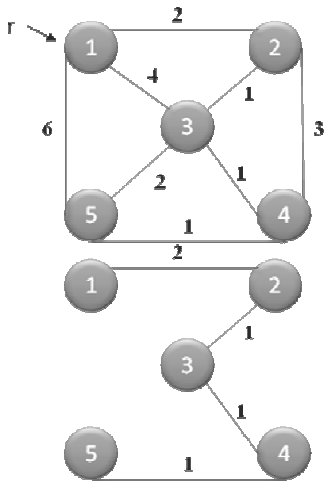
+ Znajdywanie minimalnego drzewa rozpinającego – **algorytm Prima**
  - × $T \subseteq E$ – acykliczny podzbiór krawędzi łączących wszystkie węzły z V
  - × **dane**: spójny, nieskierowany graf ważony G, węzeł r (korzeń T)
    - ✶ key[v] – minimalna waga krawędzi łączącej v z węzłami T
  - × **wynik**: $T = \{(v, \pi[v]): v \in V - \{r\}\}$

Prim(G, r)

Q ← V
for(each u ∈ V – {r})
  key[u] ← ∞
key[r] ← 0
π[r] ← NIL

} inicjalizacja

while(Q ≠ ∅)
  u ←dequeueMin(Q)
  for(each v ∈ A[u])
    if(v ∈ Q and w(u, v) < key[v])
      π[v] ← u
      key[v] ← w(u,v)



+ **złożoność**: O(m log(n))