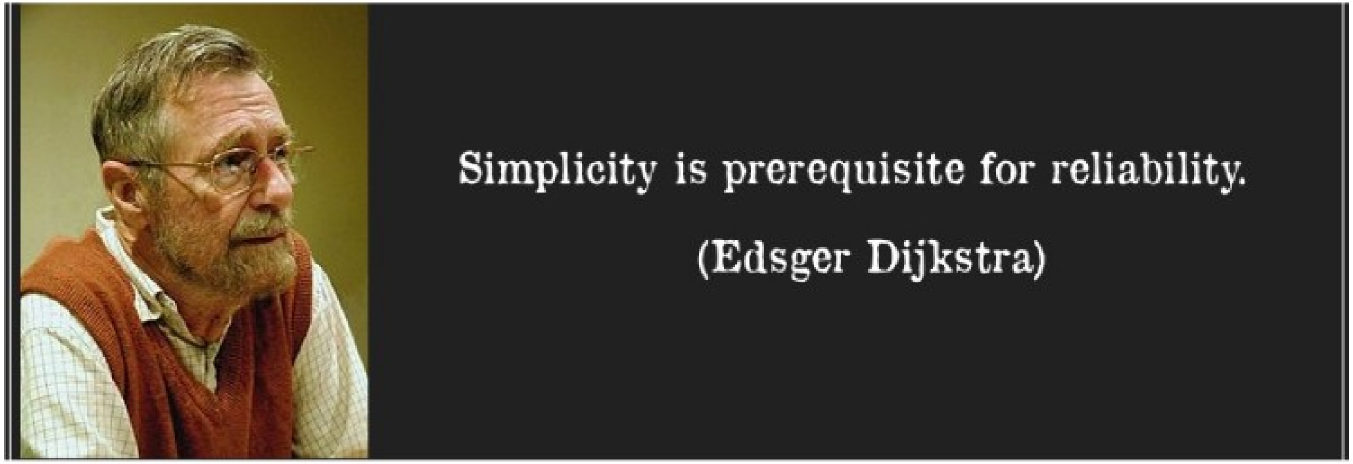
Алгоритм Дейкстри

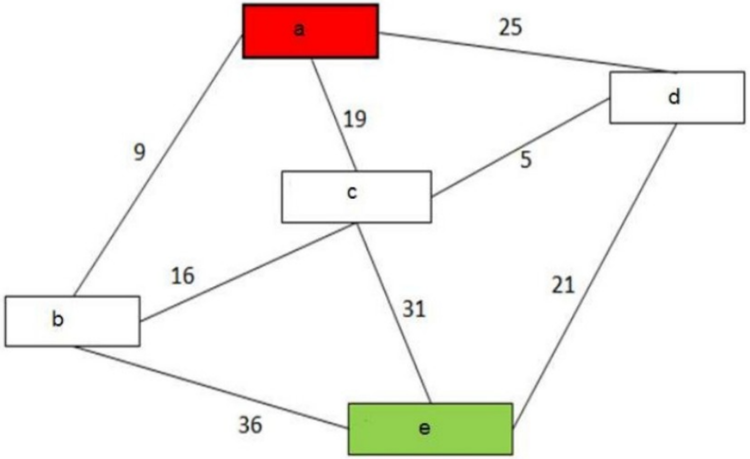
Едсгер Вібе Дейкстра

- Травень 1930 – серпень 2002
- Нідерландський науковець у галузі комп’ютерних наук
- Дослідження застосування математичної логіки при розробці комп’ютерних систем, брав активну участь у розробці мови програмування АЛГОЛ
- У 1972 році отримав премію Тюрінга



Суть алгориму

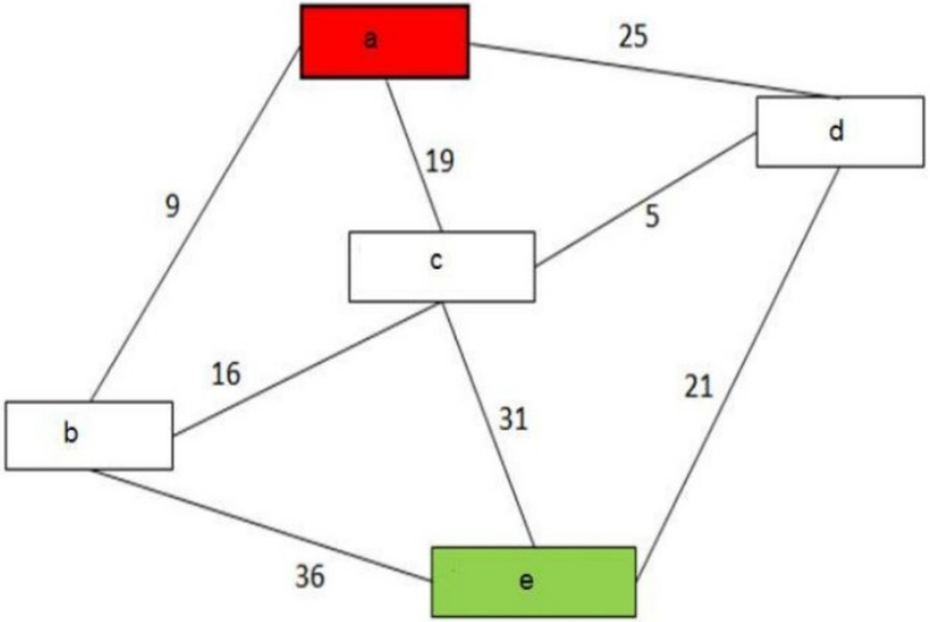
- Вирішує проблему пошуку найкоротшого шляху з однієї вершини графу до всіх інших
- Всі ребра повинні мати позитивні ваги
- У простому випадку складність алгоритму $O(n^2)$, у більш складніших варіантах O(nlogn)
- Області застосування: телефонні та інтернет мережі, побудова маршруту



Суть алгориму

Крок 1:
Позначаємо стартову вершину 0, а інші inf
L(a) = 0
L(b) = inf
L(c) = inf
L(d) = inf
L(e) = inf

S = {a}
Q = {b, c, d, e}

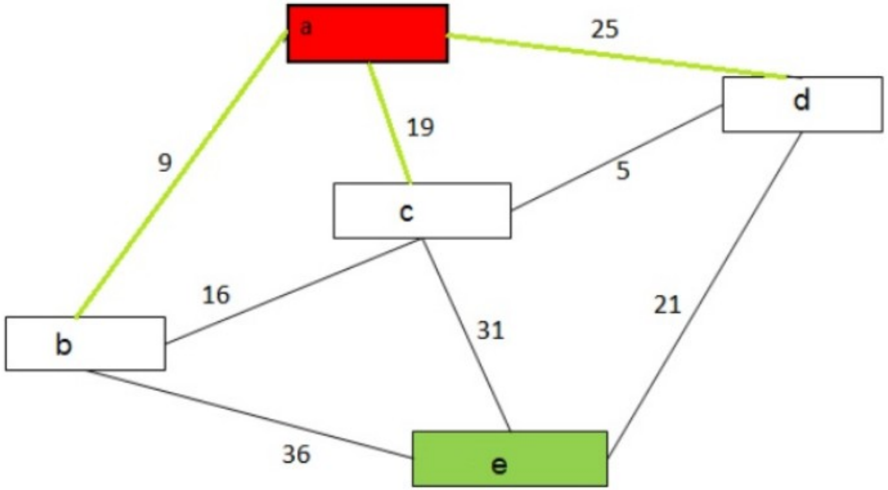


Суть алгориму

Крок 2:
Визначаємо відстані до суміжних вершин
$L(i) = \min\{L(n), L(j) + w(i, j)\}$

$L(b) = \min\{ L(b), L(a) + w(a, b)\}=$
$\min \{inf, 9\} = 9$
$L(c) = 19$
$L(d) = 25$

Переходимо в вершину яка має найменшу відстань та оперуємо з неї

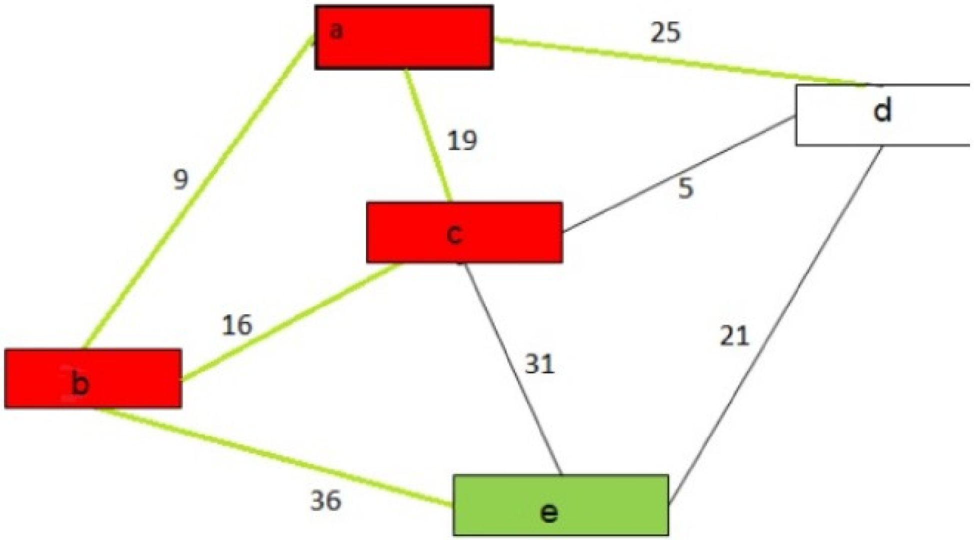


Суть алгориму

Крок 3:
S = {a, b}
Q = {c, d, e}

L(c) = min {19, 9+16} = 19
L(e) = min {inf, 9+36} = 45

L(c) < L(e) =>   S = {a, b, c}

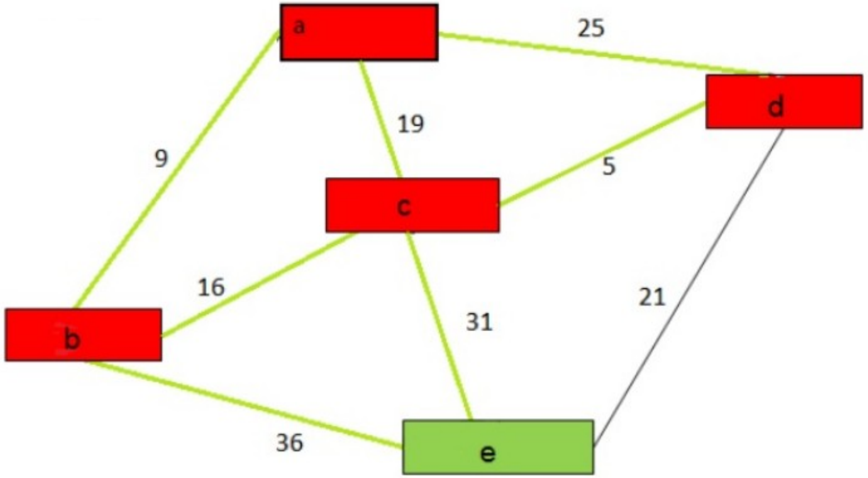


Суть алгориму

Крок 4:
S = {a, b, c}
Q = {d, e}

L(d) = min {25, 19+5} = 24
L(e) = min {45, 19 + 31} = 45

L(d) < L(e) => S = {a, b, c, d}

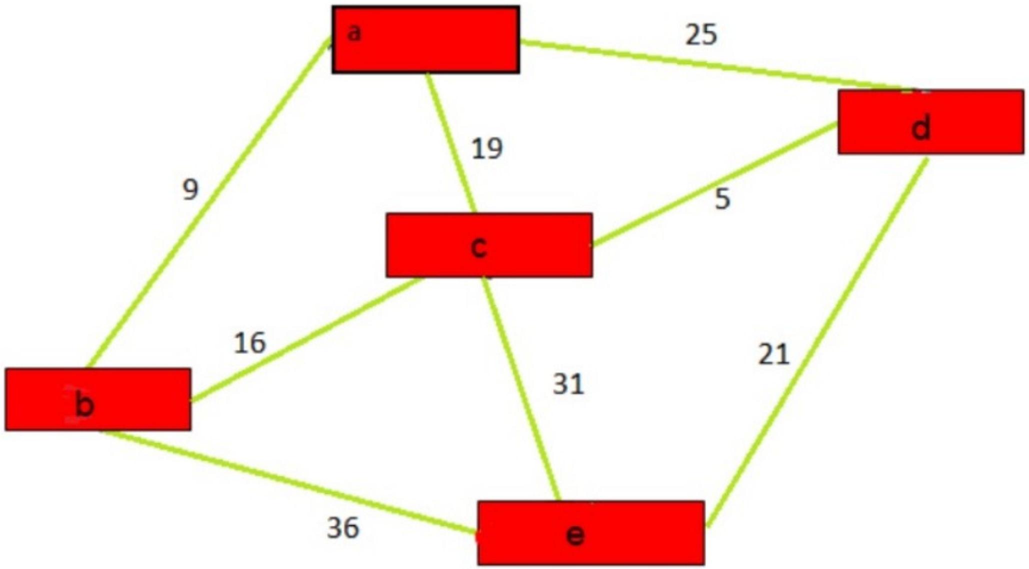


Суть алгориму

Крок 5:
S = {a, b, c, d}
Q = {e}

L(e) = {45, 25+21} =45

S = {a, b, c, d, e}



Псевдокод

```
dist[s] ←0                                  (distance to source vertex is zero)
for  all v ∈ V–{s}
        do  dist[v] ←∞                      (set all other distances to infinity)
S←∅                                         (S, the set of visited vertices is initially empty)
Q←V                                         (Q, the queue initially contains all vertices)
while Q ≠∅                                  (while the queue is not empty)
do   u ← mindistance(Q,dist)                (select the element of Q with the min. distance)
      S←S∪{u}                               (add u to list of visited vertices)
       for all v ∈ neighbors[u]
             do  if   dist[v] > dist[u] + w(u, v)        (if new shortest path found)
                        then      d[v] ←d[u] + w(u, v)   (set new value of shortest path)
                                                         (if desired, add traceback code)

return dist
```