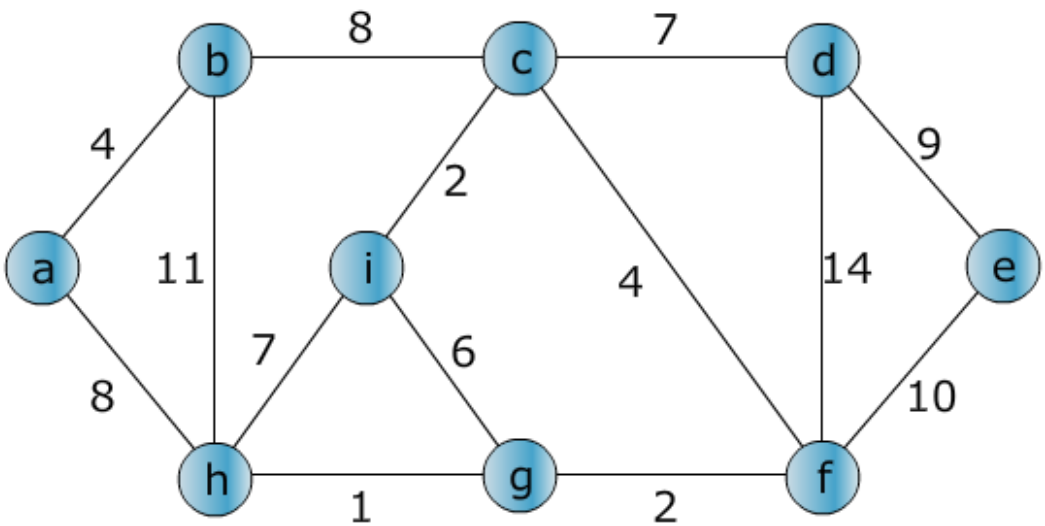
Алгоритм Крускала

Нехай маємо зв'язний неорієнтований граф в якому вершини з'єднуються ребрами , для кожного з яких задається вага

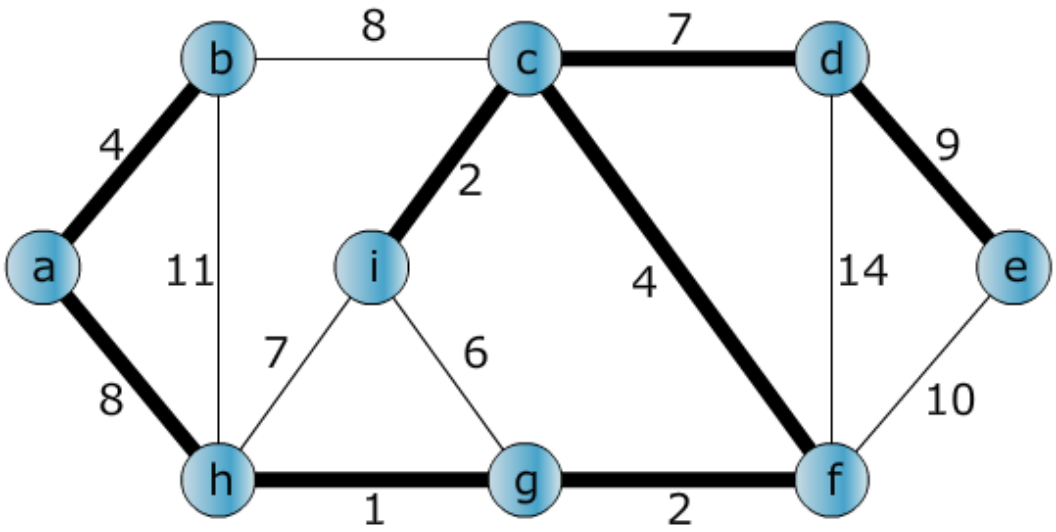
Далі



Алгоритм Крускала

Задача побудови мінімального остового дерева полягає в побудові зв'язного підграфа, який містить усі вершини графа та має мінімальну вагу.

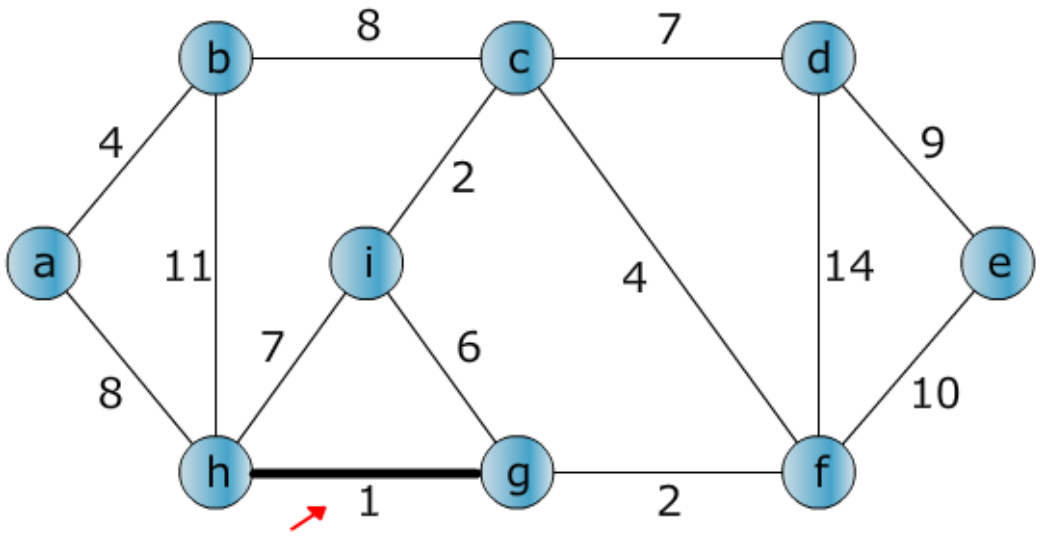
Далі



Алгоритм Крускала

Реалізація алгоритму Крускала починається зі знаходження ребра з мінімальною вагою. Дві вершини, що з'єднуються цим ребром утворюють дерево.

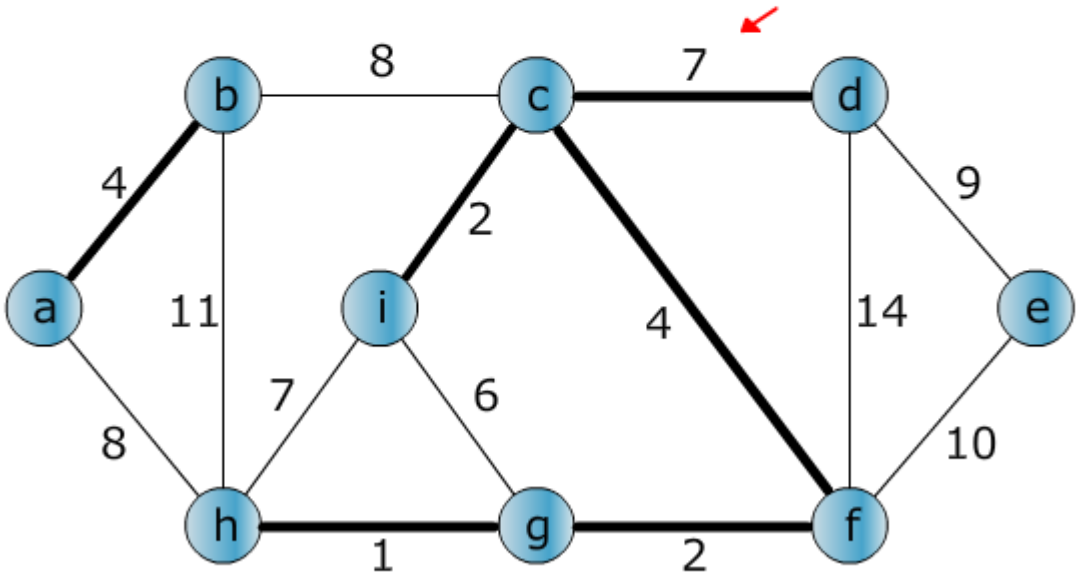
Далі



Алгоритм Крускала

Продовжуємо процес, на кожному кроці виконуючи перевірку: якщо обидва кінці ребра входять в одне дерево, то дане ребро пропускаємо.

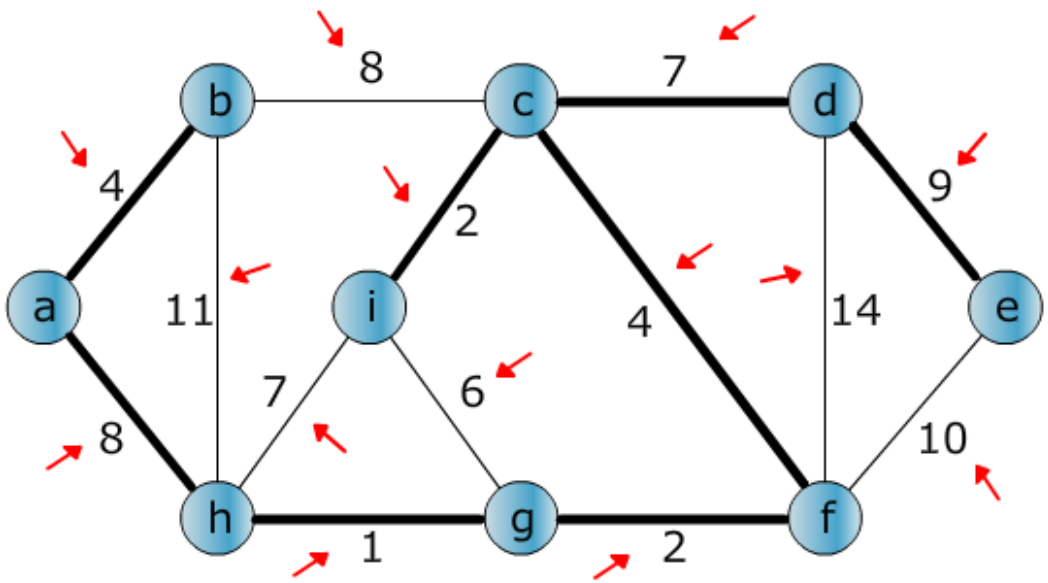
Далі



Алгоритм Крускала

Продовжуємо процес до того моменту, коли усі ребра будуть перевірені.

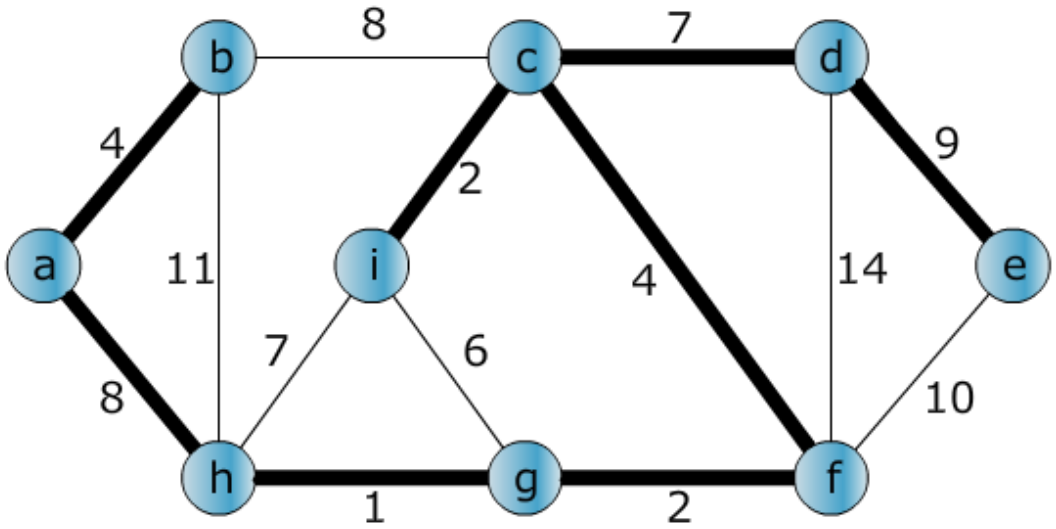
Далі



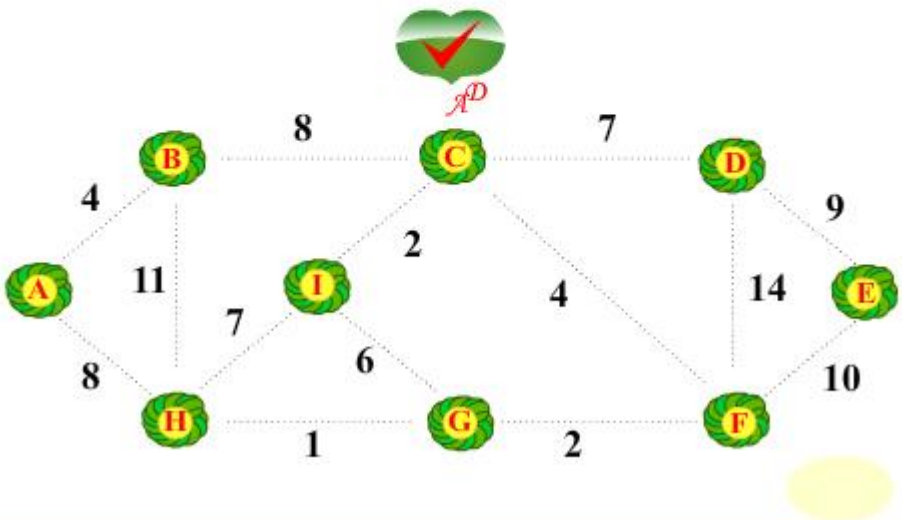
Алгоритм Крускала

Мінімальне остове дерево побудоване.

Спочатку

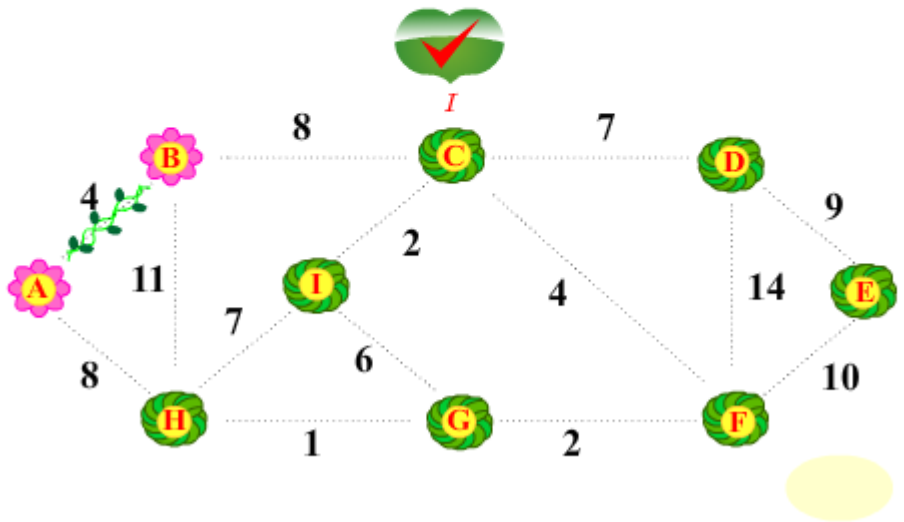


Алгоритм Прима (альтернативная версия)



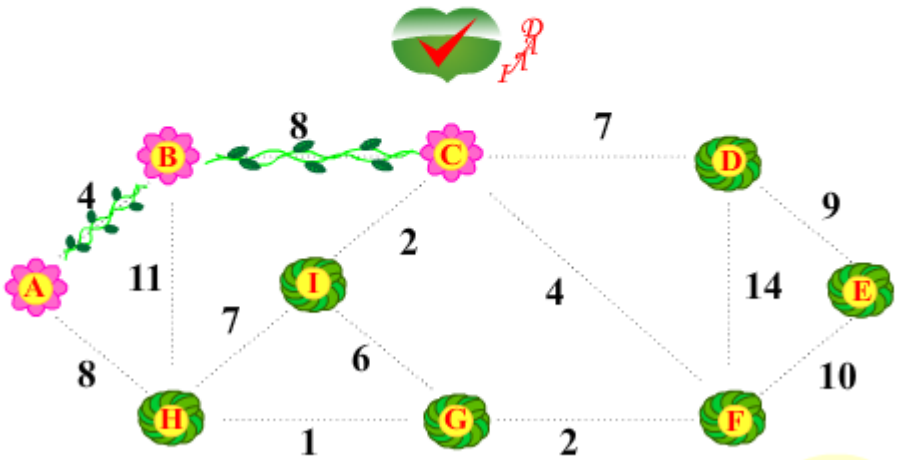
Алгоритм Прима (альтернативная версия)

Далі серед ребер з'єднуючих вершини дерева з вершинами, що не належать дереву, додаємо ребро найменшої ваги.
Вершина a формує дерево, серед ребер з вагою 4 та 8 обираємо найменшу, тобто 4.



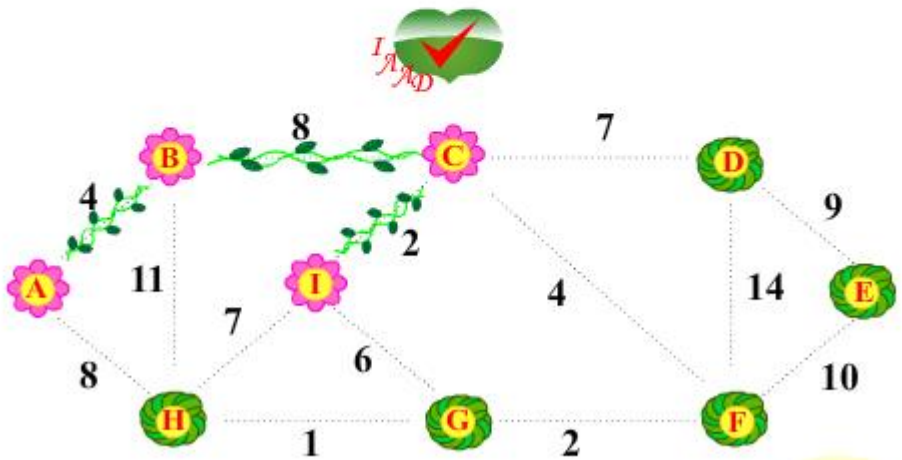
Алгоритм Прима (альтернативная версия)

На наступному кроці, за аналогією:
Вершини *A* та *B* формують дерево, серед ребер з вагою 8, 8 та 11 обираємо найменшу, тобто 8. Можемо додати до дерева як вершину C так і вершину H.



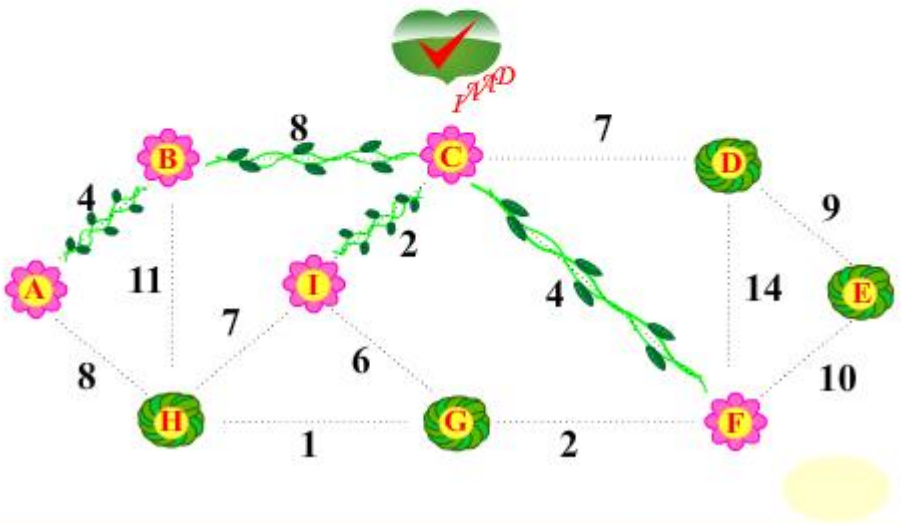
Алгоритм Прима (альтернативная версия)

Далі вершини обираються аналогічно.



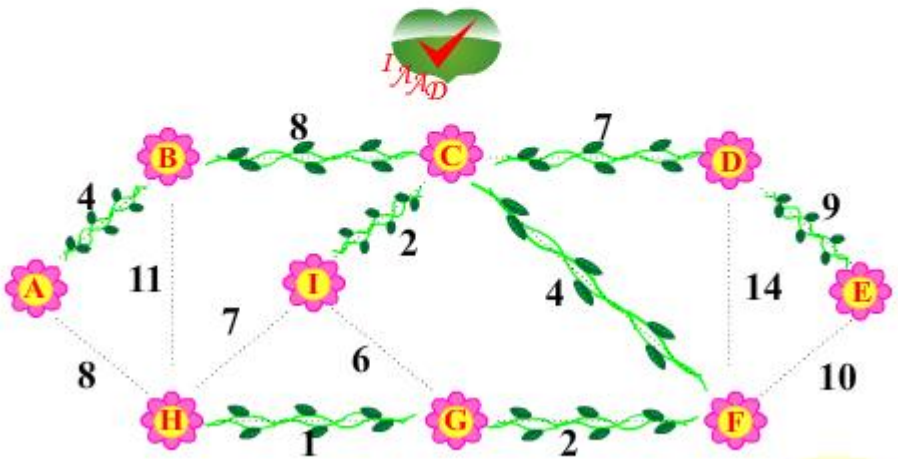
Алгоритм Прима (альтернативная версия)

Далі вершини обираються аналогічно.



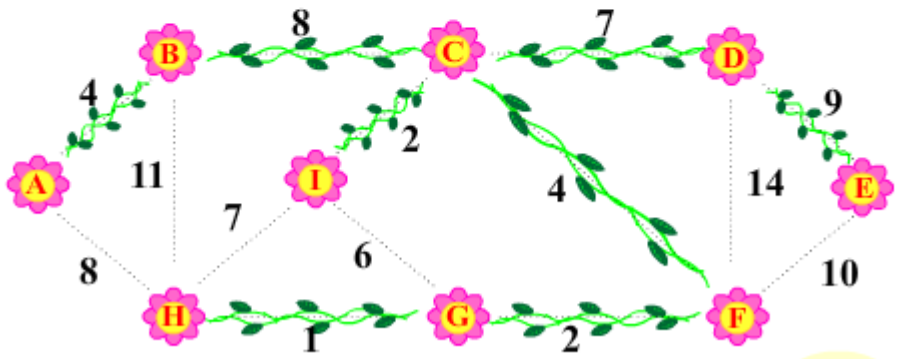
Алгоритм Прима (альтернативная версия)

Далі вершини обираються аналогічно.



Алгоритм Прима (альтернативная версия)

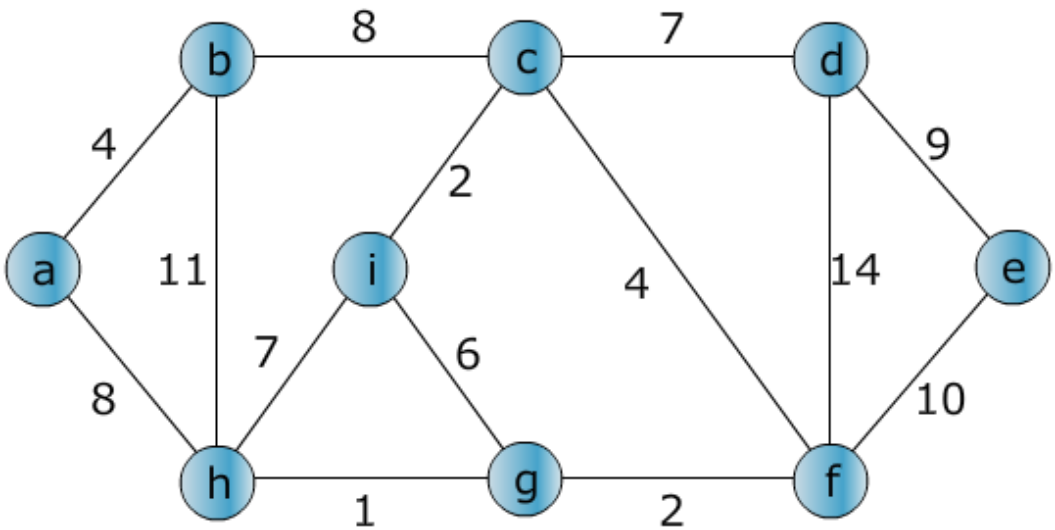
Мінімальне остове дерево побудоване



Алгоритм Прима

Нехай маємо зв'язний неорієнтований граф в якому вершини з'єднуються ребрами , для кожного з яких задається вага

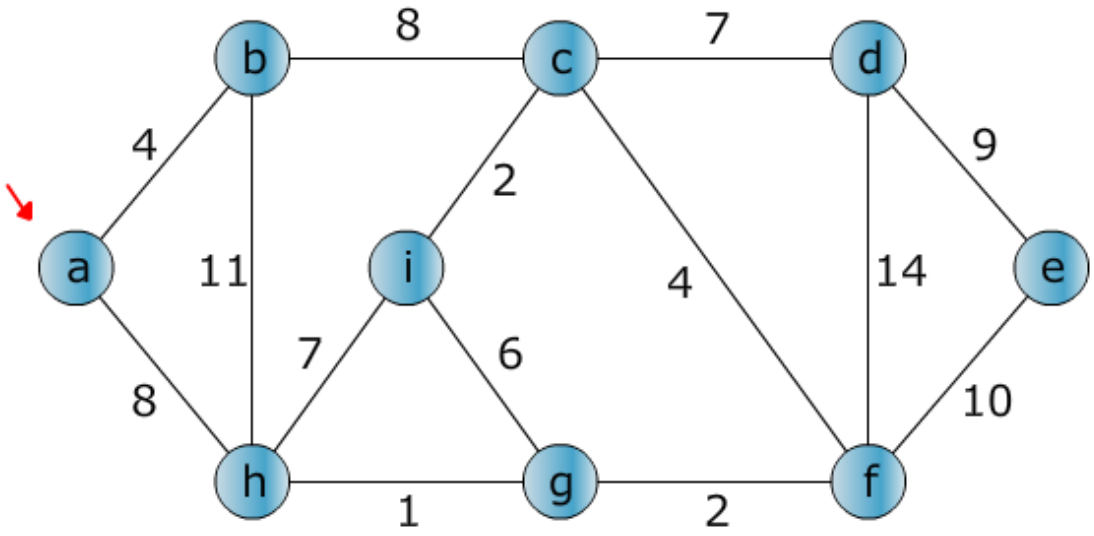
Далі



Алгоритм Прима

В алгоритмі Пріма формування дерева починається з будь-якої кореневої вершини. Нехай це буде вершина a.

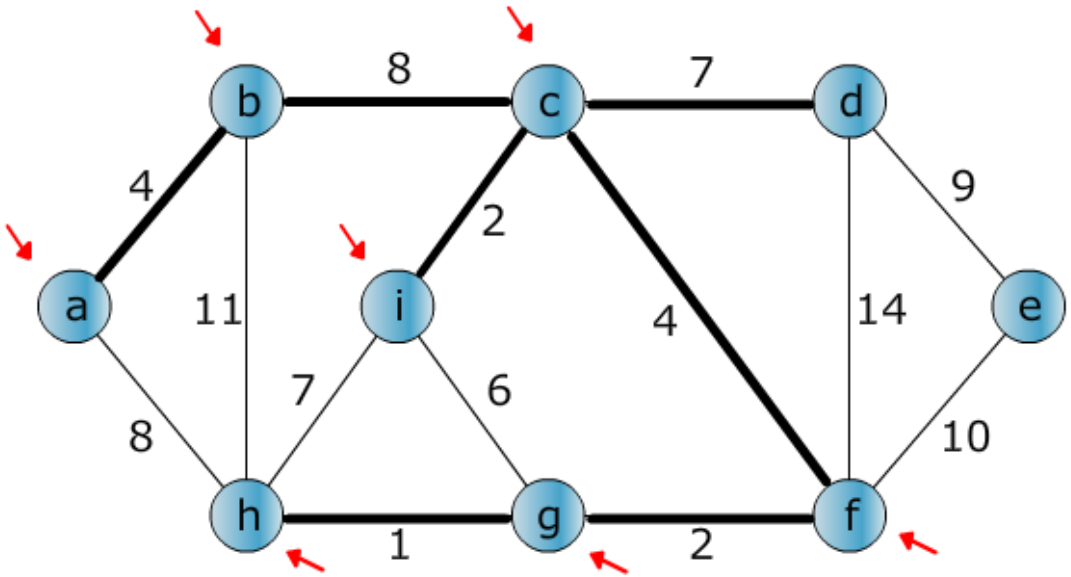
Далі



Алгоритм Прима

Продовжуємо процес.

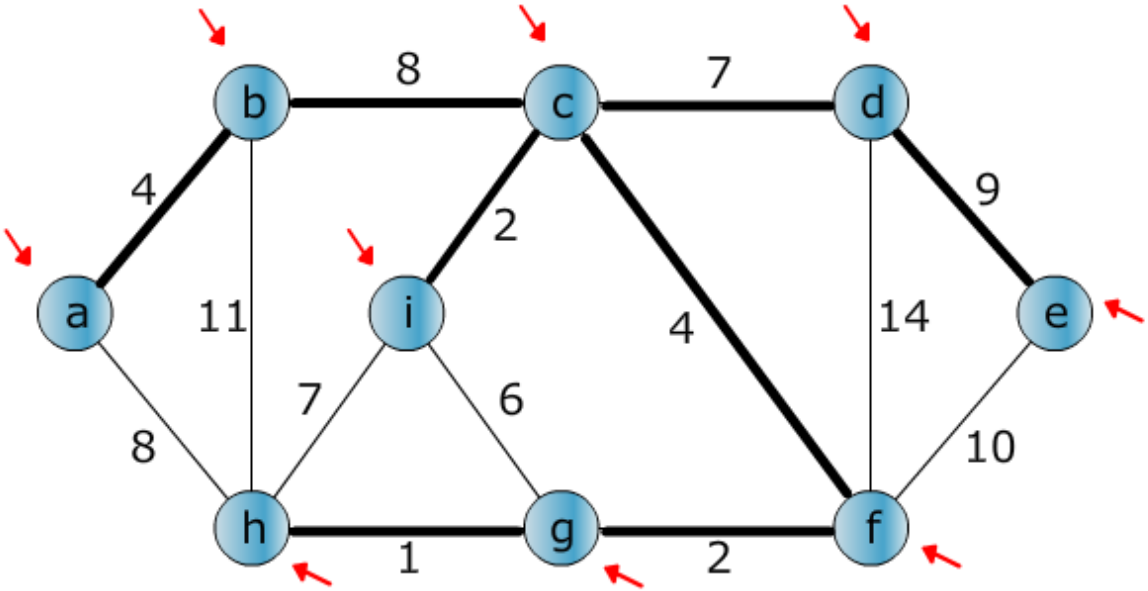
Далі



Алгоритм Прима

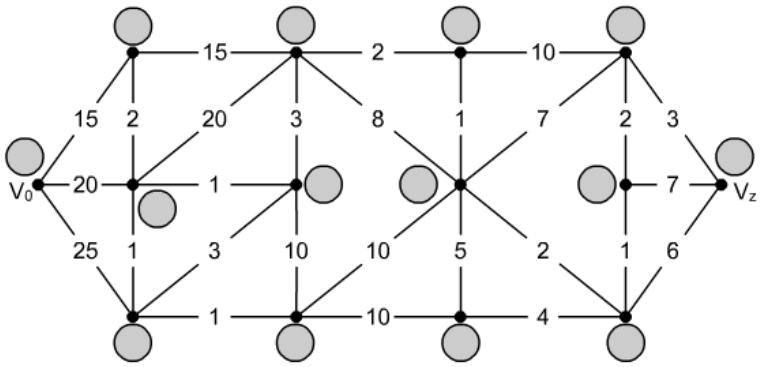
Продовжуємо процес до того моменту, коли в дереві будуть усі вершини графа.

Далі



Демонстрация алгоритма нахождения самого короткого пути на графике

Дискретная математика. **Графы**

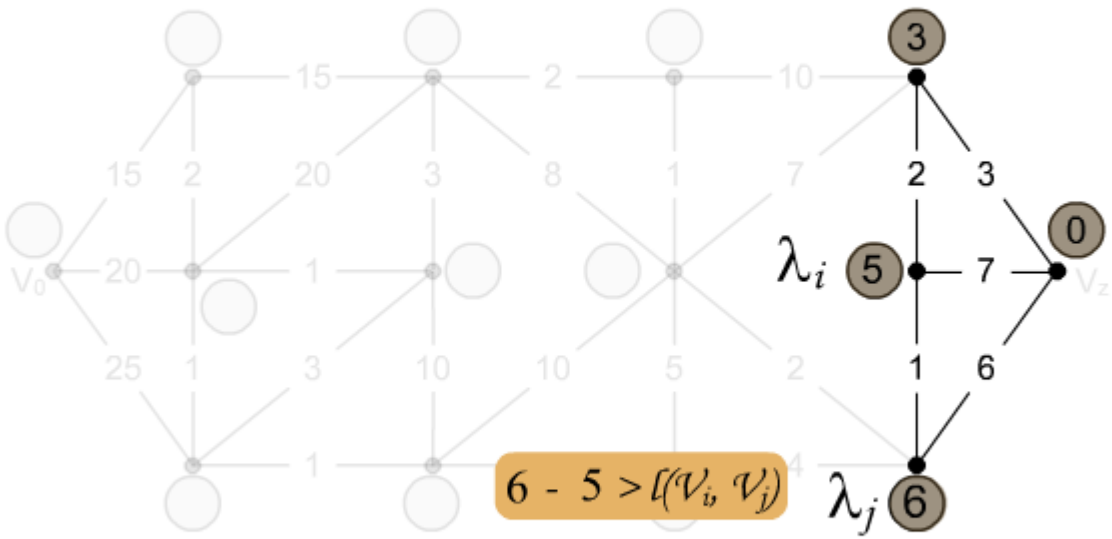


Дискретная математика. **Графы**

ИНФОРМАЦИОННОЕ ОКНО

Продолжаем замену индексов до тех пор, пока остается хотя бы одна дуга, для которой можно уменьшить λ_j

Дискретная математика. **Графы**

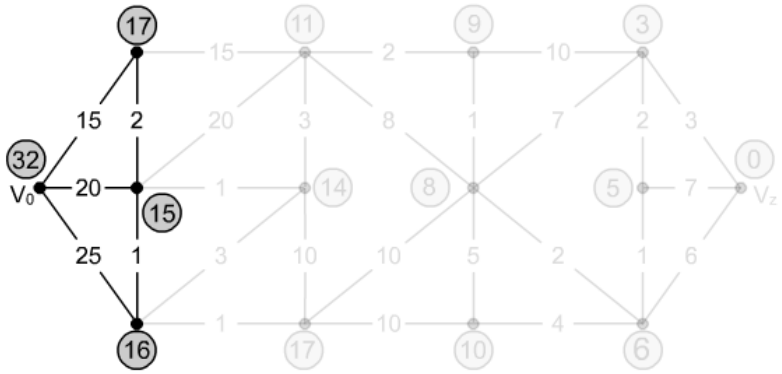


ИНФОРМАЦИОННОЕ ОКНО

Продолжаем замену индексов до тех пор, пока остается хотя бы одна дуга, для которой можно уменьшить λ_j

Демонстрация алгоритма нахождения самого короткого пути на графике

Дискретная математика. **Графы**

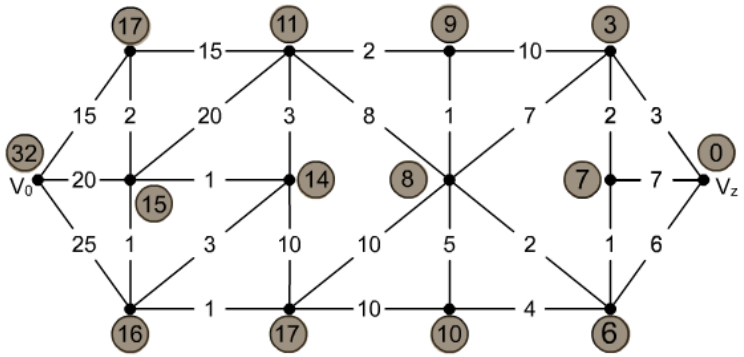


ИНФОРМАЦИОННОЕ ОКНО

Выделим маршрут, содержащий кратчайший путь из вершины V_0 к V_z. От текущей вершины V_i с индексом λ_i движемся к вершине V_j с индексом λ_j, для которой выполняется условие: $\lambda_i - \lambda_j = l(V_i, V_j)$.

Демонстрация алгоритма нахождения самого короткого пути на графике

Дискретная математика. **Графы**

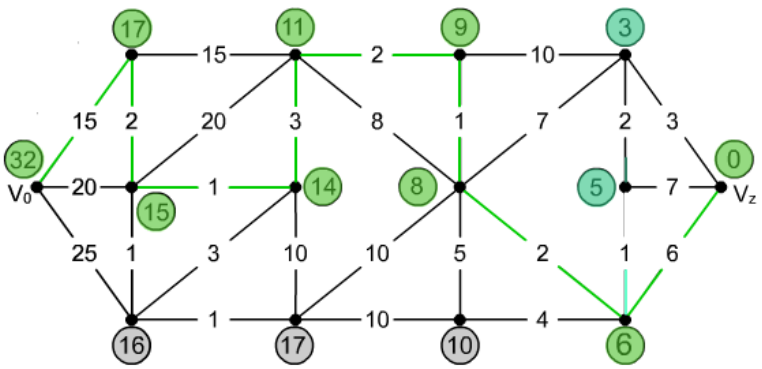


ИНФОРМАЦИОННОЕ ОКНО

Таким образом, длина кратчайший путь в графе равна индексу начальной вершины = 32

Демонстрация алгоритма нахождения самого короткого пути на графике

Дискретная математика. **Графы**



ИНФОРМАЦИОННОЕ ОКНО

Внимание! Вершины с индексом 5 и 3 также удовлетворяет условию - следовательно имеется еще один минимальный путь. Покажем его иным цветом.