Мінімальне остовне дерево (МОД)

Будемо розглядати графи та мультиграфи, але не псевдографи. Тобто допускаються кратні ребра, але не петлі.

Означення 1.Шлях, або маршрут (path) у графі $G = (V, E)$ – це послідовність вершин та ребер виду $v_1\, e_1\, v_2\, e_2 \ldots v_k$, у якій сусідні елементи є інцидентними.

Означення 2. Шлях називається простим (simple path), якщо кожна вершина зустрічається в ньому лише один раз.

У простому шляху немає перетинів (вершин, що повторюються) і, як наслідок, не може бути повторюваних ребер. Протилежне твердження не має місця: ребра у шляху можуть бути унікальними, але вершини повторюватися у точках перетину; і такий шлях не простий.

Означення 3 . Граф $G = (V, E)$ називається зв'язним (connected graph), якщо існує шлях із будь-якої його вершини у будь-яку іншу.

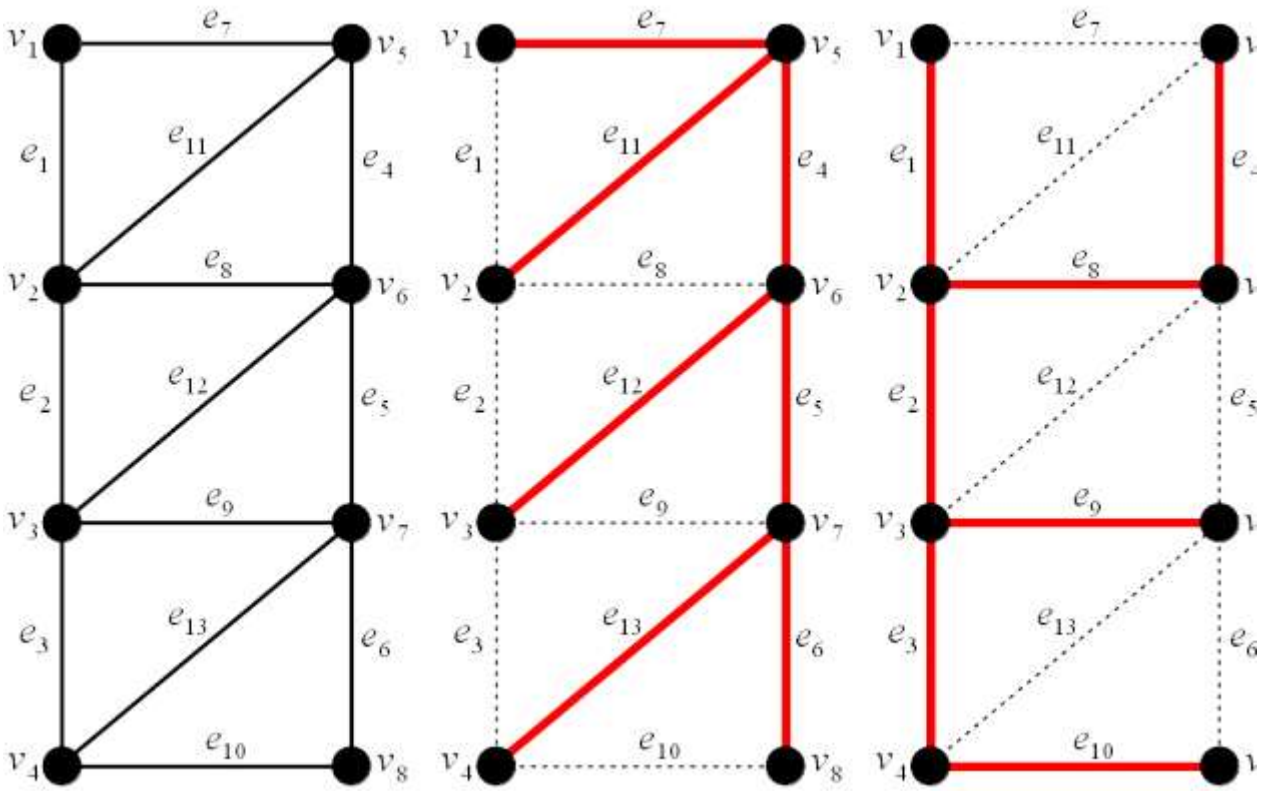
У незв'язних графах можна виділити окремі зв'язні компоненти. У самому крайньому випадку, коли у графа взагалі немає ребер: $E = \emptyset$, у нього буде n компонент – по одній вершині у кожній.

Розглянемо зв'язний граф (або мультиграф) G. Якщо з нього видалити деякі ребра,він може залишится зв'язним, а може й розпастися на окремі компоненти. Яку максимальну кількість ребер та які з них можна видалити, щоб граф залишився зв'язним? Якщо ребра графа зважені, то можна поставити задачу так: як видалити максимальну кількість ребер максимальної ваги, щоб залишився зв'язний граф із загальною мінімальною вагою ребер?

Спочатку розглянемо простішу, незважену задачу: яка мінімальна кількість ребер необхідна для зв'язності графа з n вершинами?

Теорема 1. У зв'язному графі $G = (V, E)$ виконується: $m \geq n-1$. Тобто мінімально можлива кількість ребер зв'язного графа є n – 1.

Означення 4. Зв'язний граф $G = (V, E)$ з n вершинами та n – 1 ребрами називається остовним деревом (spanning tree).



Властивості остовних дерев.

Властивість 1. В остовному дереві $G_1 = (V, E_1)$ зв'язного графа $G = (V, E)$ немає циклів.

Властивість 2. В остовному дереві G_1 зв'язного графа існує єдиний шлях з кожної вершини в кожну іншу.

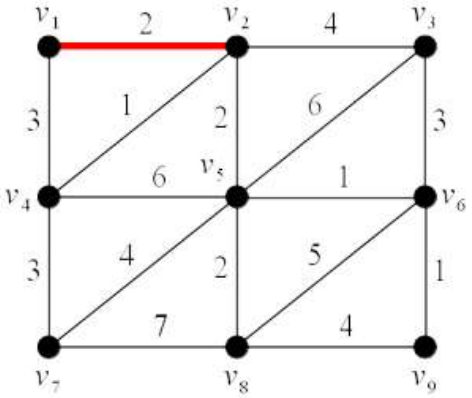
Алгоритм 4.1. Алгоритм Пріма (Prim's algorithm).

Роботу алгоритму проілюструємо на наступному прикладі.

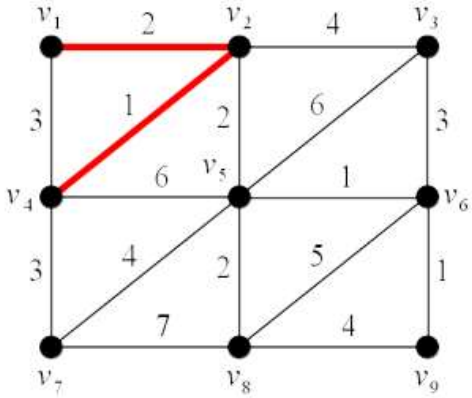
Біля вершин проставлені їхні номери, а біля ребер – ваги. У нашому графі n = 9, m = 16,тому в побудованому МОД повинно залишитися лише 8 ребер з 16. Візьмемо перше ребро мінімальної ваги, що виходить з вершини v_1. Таким ребром буде $e_{1,2}$: його вага 2 менша за вагу 3 ребра $e_{1,4}$. Тому починаємо побудову МОД нього (а). Далі з переглядаємо всі ребра, суміжні до вже побудованого фрагменту МОД, тобто до $e_{1,2}$. Всього таких ребер 4

(одне у вершині v_1 та три у v_2). Обираємо з них ребро мінімальної ваги. Мінімальна вага серед цих 4 ребер – 1, вона у ребра $e_{2,4}$. Його й приєднуємо до МОД, що будується (б).

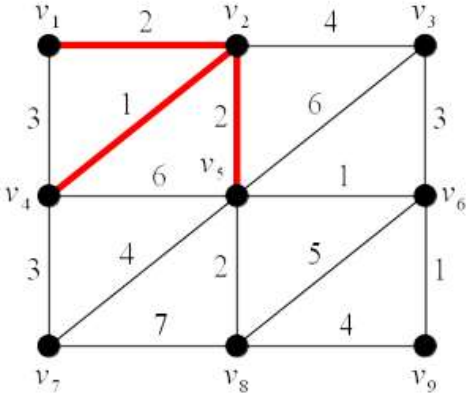
Алгоритм Пріма



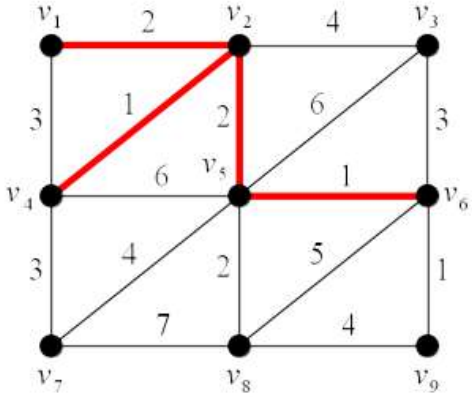
а)



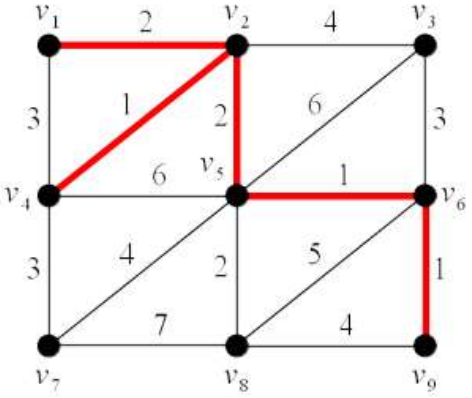
б)



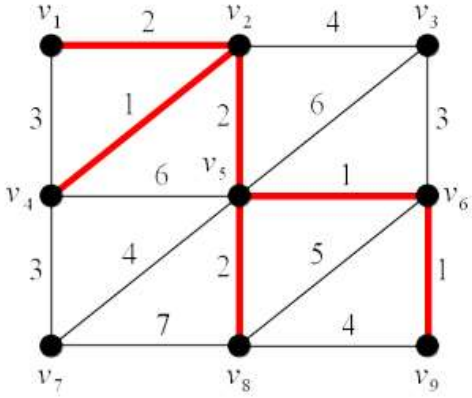
в)



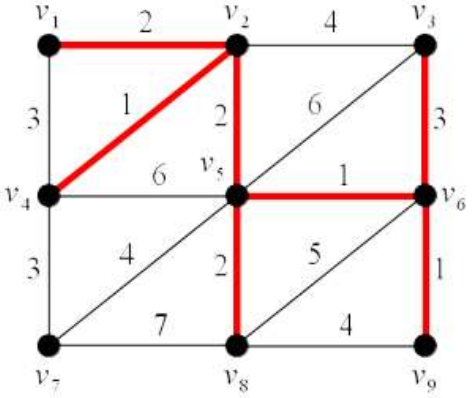
г)



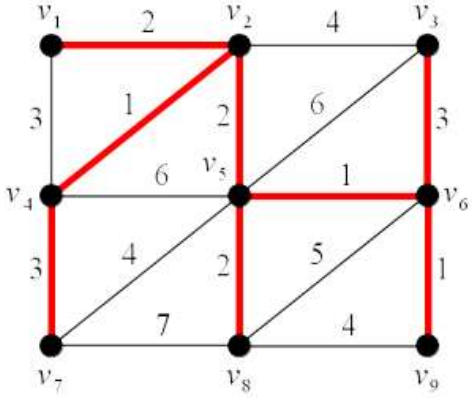
д)



е)



є)



ж)

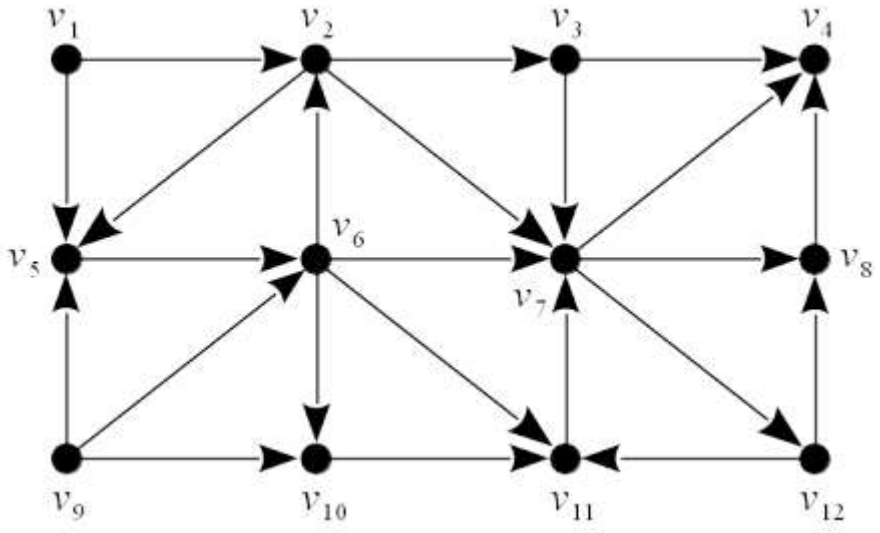
Алгоритм 2. Алгоритм Краскала (Kruskal's algorithm).

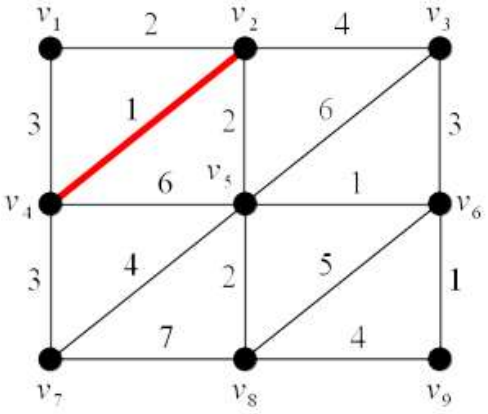
Основна його відмінність від алгоритма Пріма – ми будемо додавати в

МОД, що будується, не обов'язково суміжні ребра. Головне, щоб не утворювалися цикли.

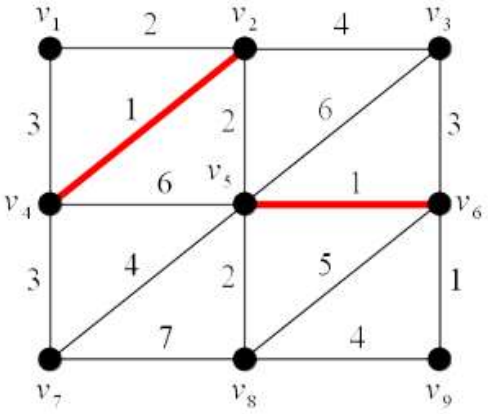
Переглядаємо всі ребра мінімальної ваги 1. Починаємо з $e_{2,4}$ – воно зустрілося першим(а). Наступне ребро ваги 1 – це $e_{5,6}$, і воно не утворює циклів – додаємо його (б). Далі перевіряємо $e_{6,9}$: циклів немає, долучаємо (в). Всі ребра з вагою 1 вичерпані. Переходимо до ребер з наступною мінімальною вагою 2. Ребро $e_{1,2}$ можна приєднати (г), $e_{2,5}$ – також (д), $e_{5,8}$ теж підходить (е). Далі – ребра з вагою 3. Ребро e1, 4 не годиться (утворюєтьсяцикл), а ось $e_{3,6}$ підходить (є) і $e_{4,7}$ – також (ж). В результаті отримали те же саме МОД ваги 15, что й за алгоритмом Пріма.

Як бачимо, ці алгоритми відрізняються порядком приєднання ребер. В алгоритмі Пріма ми весь час приєднували суміжні ребра, тобто нарощували МОД, залишаючи його зв'язним. В алгоритмі Краскала будуються окремі частини МОД, які потім поєднуються.

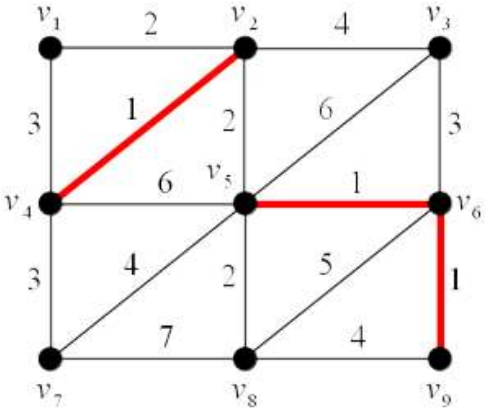



v_1 2 v_2 4 v_3
3 1 2 6 3
v_4 6 v_5 1 v_6
3 4 2 5 1
7 4
v_7 v_8 v_9

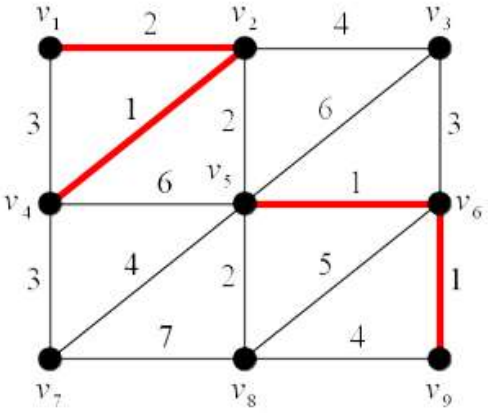
а)


v_1 2 v_2 4 v_3
3 1 2 6 3
v_4 6 v_5 1 v_6
3 4 2 5 1
7 4
v_7 v_8 v_9

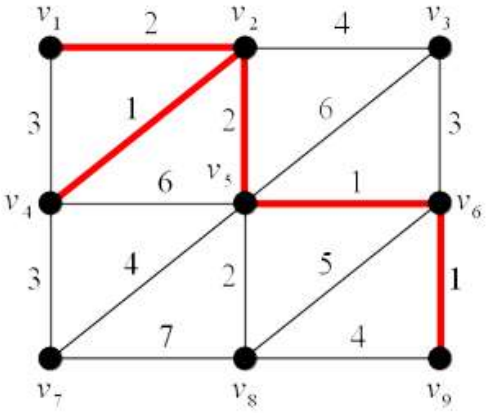
б)


v_1 2 v_2 4 v_3
3 1 2 6 3
v_4 6 v_5 1 v_6
3 4 2 5 1
7 4
v_7 v_8 v_9

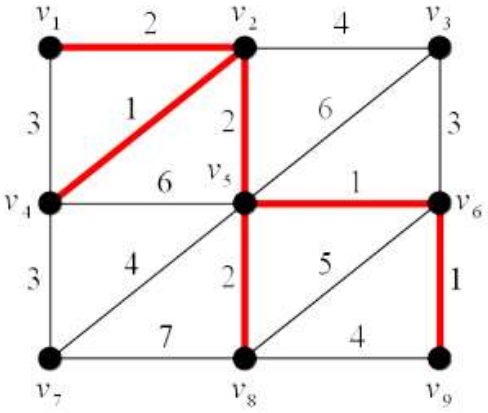
в)


v_1 2 v_2 4 v_3
3 1 2 6 3
v_4 6 v_5 1 v_6
3 4 2 5 1
7 4
v_7 v_8 v_9

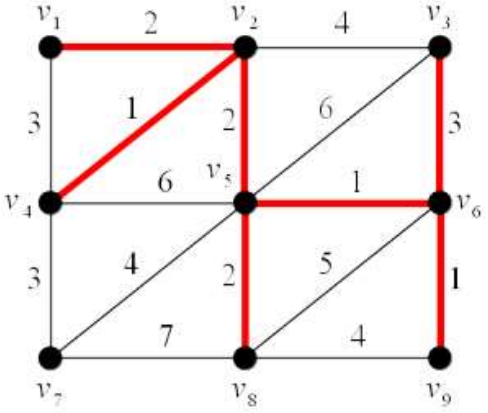
г)


v_1 2 v_2 4 v_3
3 1 2 6 3
v_4 6 v_5 1 v_6
3 4 2 5 1
7 4
v_7 v_8 v_9

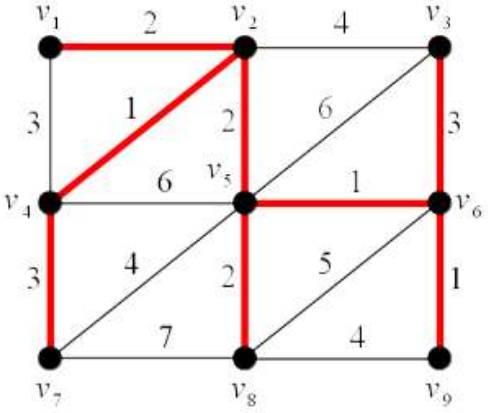
д)


v_1 2 v_2 4 v_3
3 1 2 6 3
v_4 6 v_5 1 v_6
3 4 2 5 1
7 4
v_7 v_8 v_9

е)


v_1 2 v_2 4 v_3
3 1 2 6 3
v_4 6 v_5 1 v_6
3 4 2 5 1
7 4
v_7 v_8 v_9

є)


v_1 2 v_2 4 v_3
3 1 2 6 3
v_4 6 v_5 1 v_6
3 4 2 5 1
7 4
v_7 v_8 v_9

жс)