1)Алгоритм крускала

Спочатку ребра в графі сортуються за вагою, як показано в алгоритмі крускала. Також використали функцію find() щоб перевірити чи належать вершини до однієї кореневої вершини чи ні, щоб не було циклів. Використали функцію Union щоб поєднувати вершини. Потім код перебирає ребра в посортованому графі: якщо дві вершини не належать до одного і того ж кореня, то вони коренем другої стає перша і вони об’єднуються, а в граф новий додається це ребро. Потім повертається каркас мінімальної ваги. Алгоритм працює краще з меншою кількістю ребер, адже тоді йде менше викликів функцій find та union, що зменшує час роботи алгоритму. Через це ми зробили перевірку лише на 100 ітераціях, адже воно б займало 3 години на пропрацювання всіх функцій.

2)Алгоритм Пріма

Зробили 2 сети вершин, одні-ті що код вже відвідав, інші-ті що ще не відвідав. Потім код у вершинах які відвідав шукав ребра з найменшою вагою з вершинами які не відвідав, після чого додавав ребро до графу, а нову відвідану вершину до відвіданих вершин. Раниться код теж дуже довго, адже пошук найменшого ребра займає купу часу, бо код перевіряє вагу з кожною не відміченою вершиною, що забирає час.

Підсумки:
Загалом алгоритми працюють не так погано у порівнянні з вбудованими, швидкість повільніша разів в 10 на крускала, Пріма на великих графах не справляється, працює повільніше в 40 разів, але я не використовував вбудованих бібліотек, які могли б пришвидшити роботу алгоритмів. Також відбувається багато ітерацій, які можна було б пришвидчити, але я не знаю як. Ця лабораторна робота допомогла нам зрозуміти краще алгоритми.