Звіт
Лабораторна робота №1

Виконали: Монастирська Анна, Пелех Анастасія

Хід роботи:

1. Аналіз завдань
2. Розподіл роботи
3. Перевірка
4. Звіт

Опис постановки задачі та експерименту.

Реалізувати алгоритми Крускала та Белмана-Форда. Дослідити їх ефективність на різних розмірах графів за різних умов.

Алгоритми

1.1 Алгоритм Крускала (Пелех Анастасія)
Код:

```
def kruskal_graph(graph):
    edges = sorted(list(graph.edges(data=True)), key=lambda x: x[2]['weight'])
    breaking = [{i} for i in graph.nodes()]
    kruskal = []
    for edge in edges:
        if len(breaking) == 1:
            break
        for j in breaking:
            if edge[0] in j:
                break
        for k in breaking:
            if edge[1] in k and k != j:
                breaking.remove(k)
                breaking.remove(j)
                j = j.union(k)
                breaking.append(j)
                kruskal.append((edge[0], edge[1]))
                break
    return kruskal
kruskal_graph(G)
```

1.2 Алгоритм Белмана-Форда (Монастирська Анна)

```python
def bellman_ford(G):
    '''Bellman-Ford Algorithm'''
    dict = {}
    nodes = list(G.nodes())
    edges = list(G.edges(data=True))
    s = nodes[0]
    weight_dict = {}
    '''
    create a dict for each connected vertexes
    key is 'v1,v2' and the value is weight
    '''
    for i in edges:
        weight_dict[f'{i[0]},{i[1]}'] = i[-1]['weight']

    for v in nodes:
        try:
            dict[f'{v}'] = weight_dict[f'{s},{v}']
        except KeyError:
            pass
        dict[f'{s}'] = 0
    for k in range(1,len(nodes)-1):
        for v in nodes[1:]:
            for u in nodes:
                try:
                    arg2 = dict[f'{u}']+weight_dict[f'{u},{v}']
                    dict[f'{v}'] = min(dict[f'{v}'], arg2)
                except KeyError:
                    pass

    #check if there is cycle with negative weight
    dict1 = deepcopy(dict)
    dict2 = {}
    dict2['0'] = 0
    for v in nodes[1:]:
        for u in nodes:
            try:
                arg2 = dict[f'{u}']+weight_dict[f'{u},{v}']
                dict2[f'{v}'] = min(dict[f'{v}'], arg2)
            except KeyError:
                pass
    if dict2 != dict1:
        print('Negative cycle detected')
        return None
    for key, value in dict1.items():
        print(f'Distance from {s} to {key}: {value}')
```

Програмний код проведення експериментів.

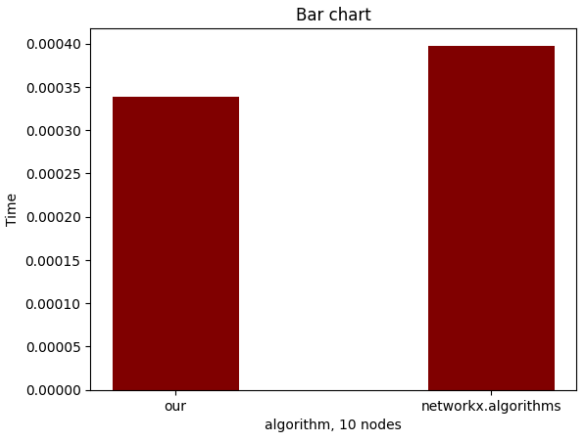
```python
def find_time(algorithm: str, vertixes: int, posibility: float, directed: bool):
    """find the time of algorithm"""
    NUM_OF_ITERATIONS = 4
    time_taken = 0
    for i in tqdm(range(NUM_OF_ITERATIONS)):
        G = gnp_random_connected_graph(vertixes, posibility, directed)
        start = time.time()
        if algorithm == 'kruskal':
            tree.minimum_spanning_tree(G, algorithm="kruskal")
        if algorithm == 'kruskal_our':
            kruskal_graph(G)
        if algorithm == 'bell':
            bellman(G, i)
        if algorithm == 'bell_our':
            bellman_ford(G)
        end = time.time()
        time_taken += end - start
    return time_taken / NUM_OF_ITERATIONS
```

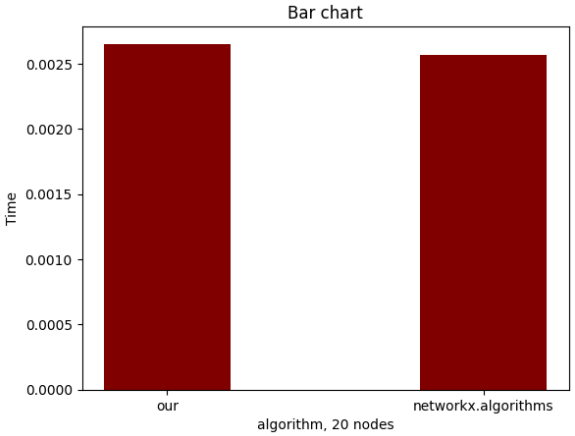
✓ 0.2s

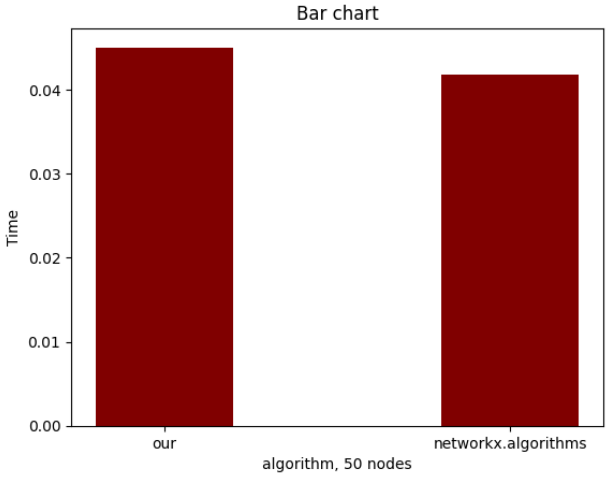
```python
def graph_plotting(posibility: float, directed: bool):
    """"""
    # x-coordinates of left sides of bars
    names = ['our', 'networkx.algorithms']
    left = [1, 2, 3, 4, 5, 6]
    # heights of bars
    for i in [10, 20, 50, 100, 200, 500]:
        func1 = find_time('kruskal_our', i, posibility, directed)
        func2 = find_time('kruskal', i, posibility, directed)
        height = [func1, func2]
        plt.bar(names, height,color ='maroon', width = 0.4)
        plt.xlabel(f'algorithm, {i} nodes')
        plt.ylabel('Time')
        plt.title('Bar chart')
        plt.show()
        plt
```

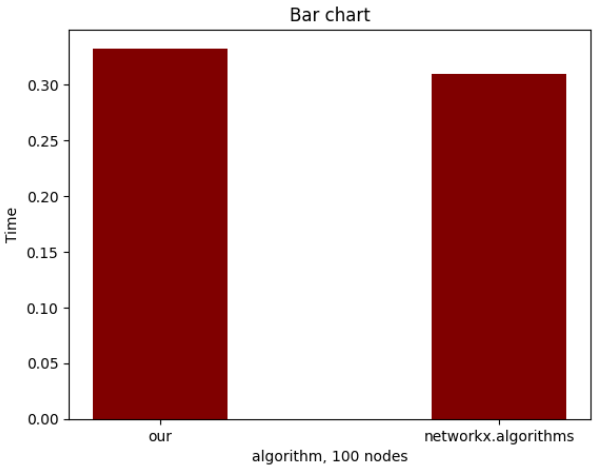
✓ 0.2s

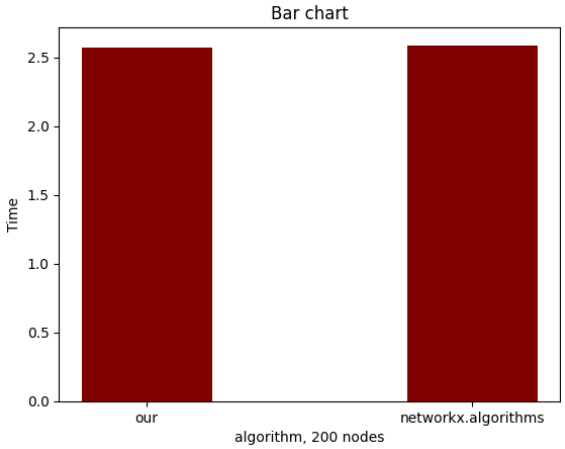
Експеримент КРУСКАЛА (вершин(пише на ox осі), 0.8 ймовірність, неорієнтований)

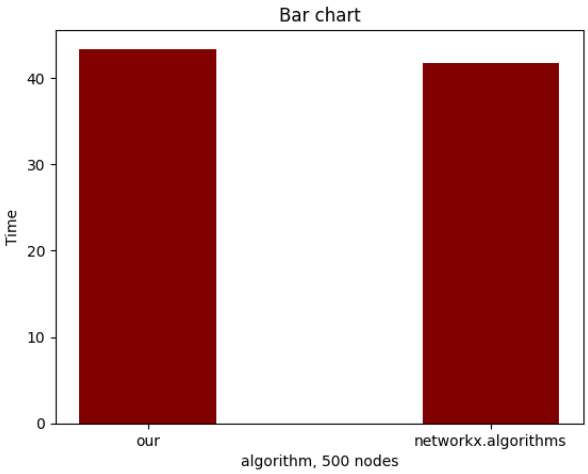






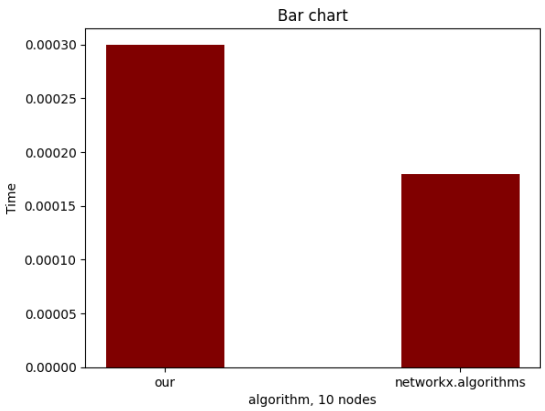


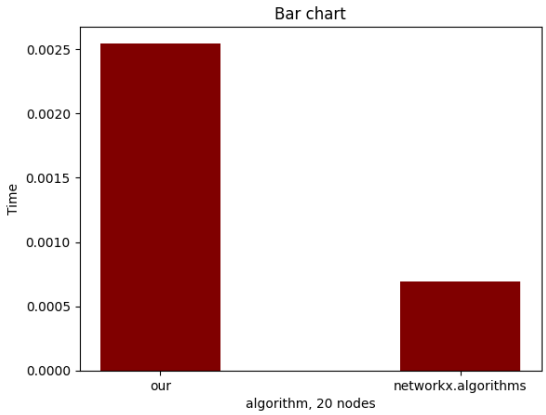


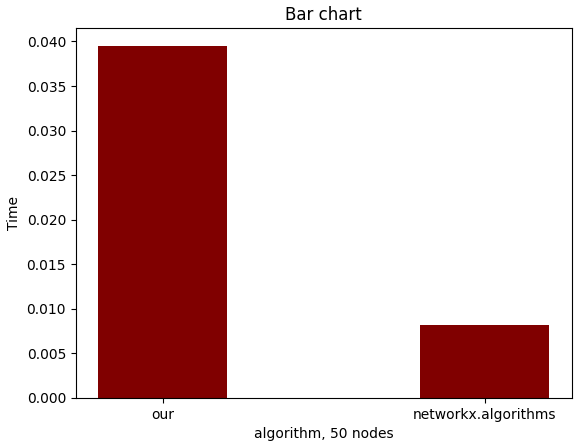


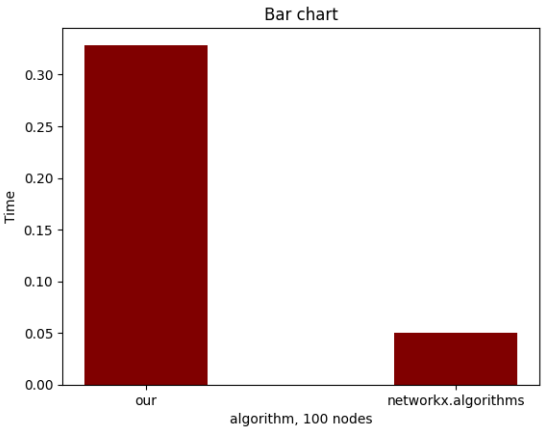
Переглядаючи результати, бачимо що час є практично однаковий. Отже наш код є досить ефективним. Параметри не грають ролі.

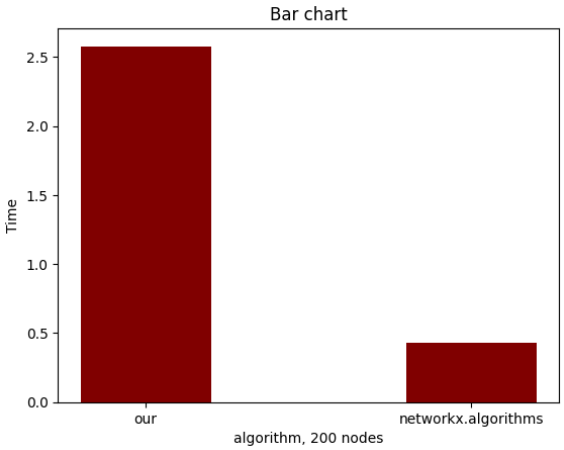
Експеримент БЕЛМАНА-ФОРДА(вершин(пише на ox осі), 0.8 ймовірність, неорієнтований)

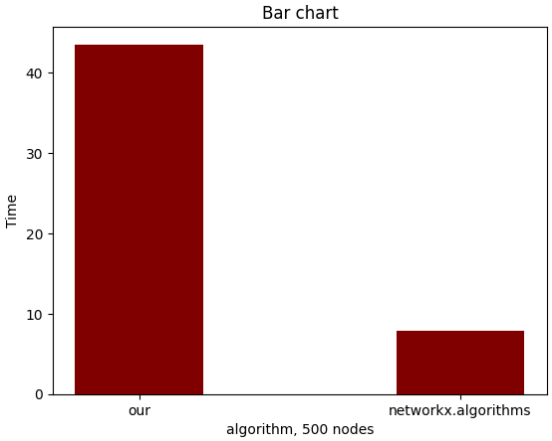






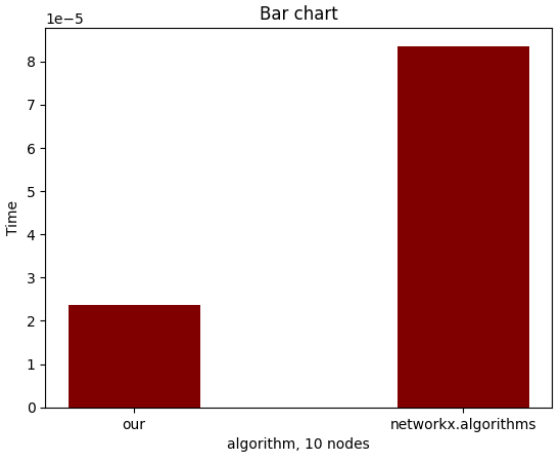


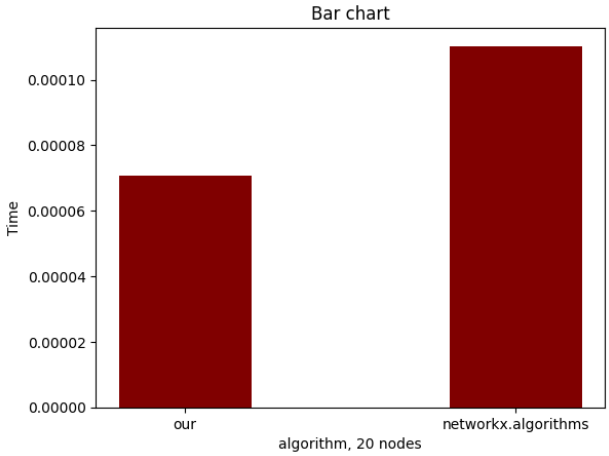


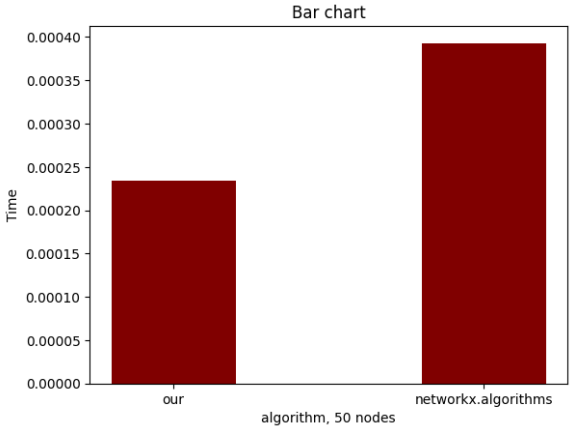


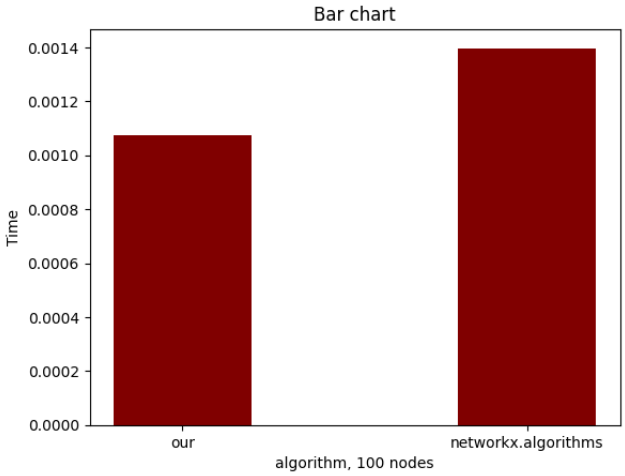
Переглядаючи результати, бачимо що часова різниця є як прірва. Отже наш код є мало ефективним. У всіх випадках результат програє.

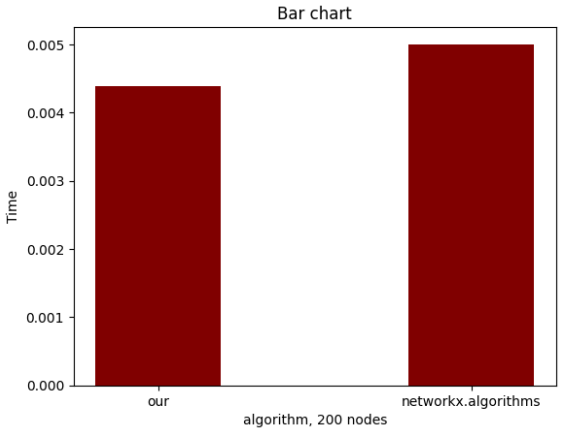
Експеримент КРУСКАЛА (вершин(пише на ox осі), 0.2 ймовірність, неорієнтований)

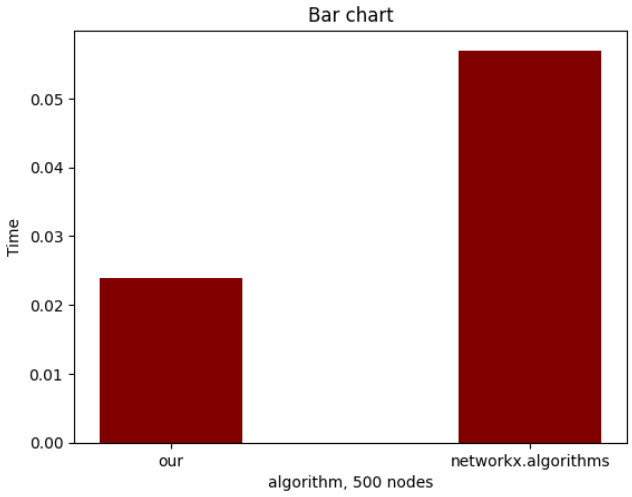






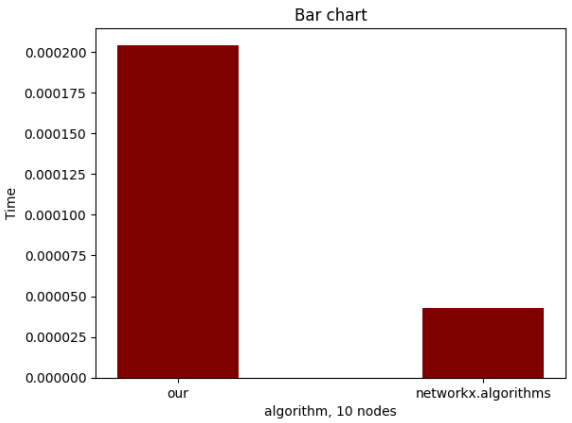


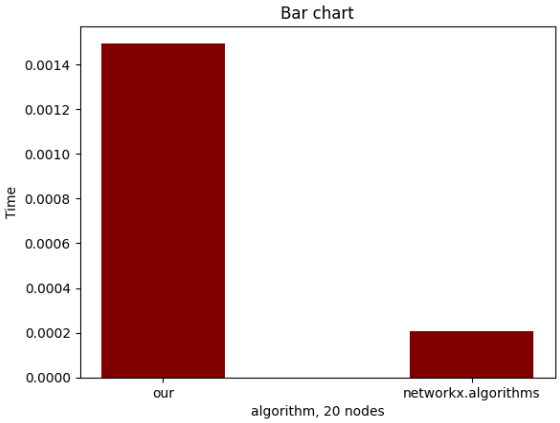


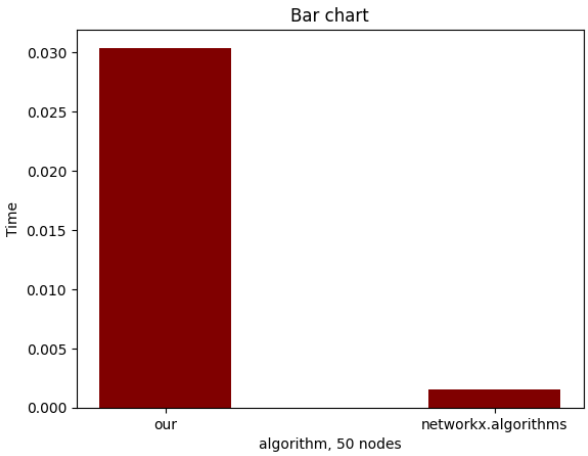


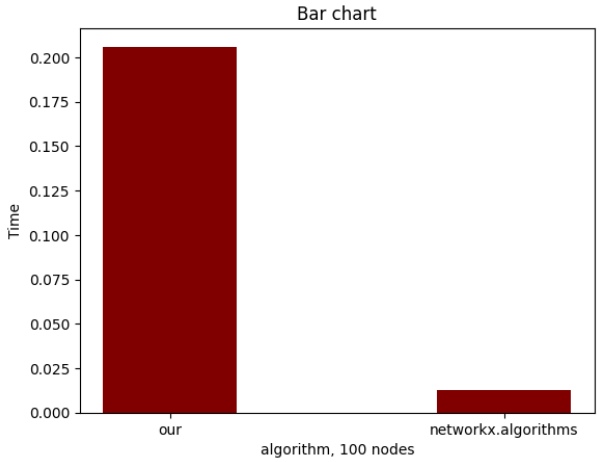
Переглядаючи результати, бачимо що часова різниця є доволі великою. Наш код є ефективним. У всіх випадках результат перемагає, іноді різниця не значна як 200 вершин.

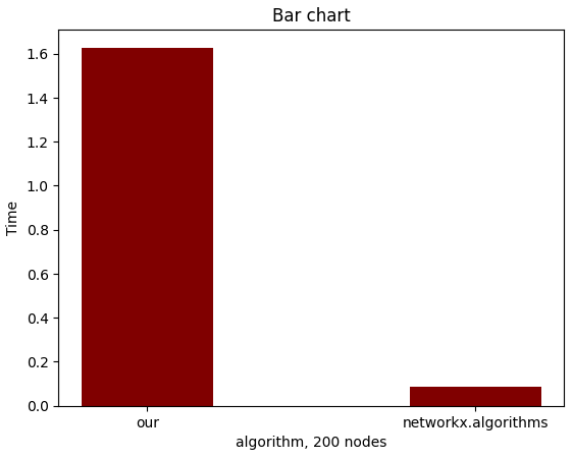
Експеримент БЕЛМАНА-ФОРДА(вершин(пише на ox осі), 0.2 ймовірність, орієнтований)

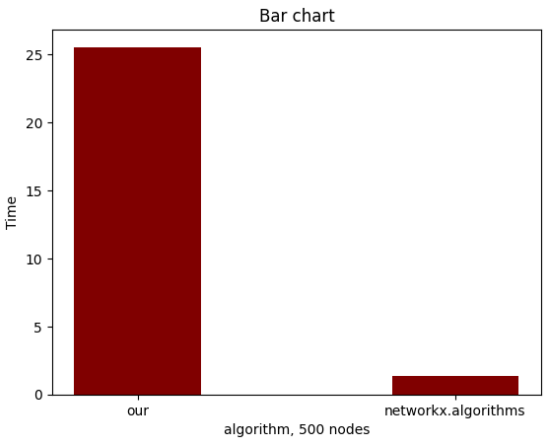






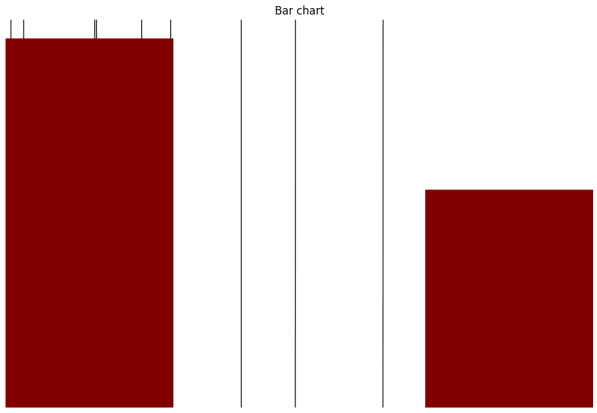


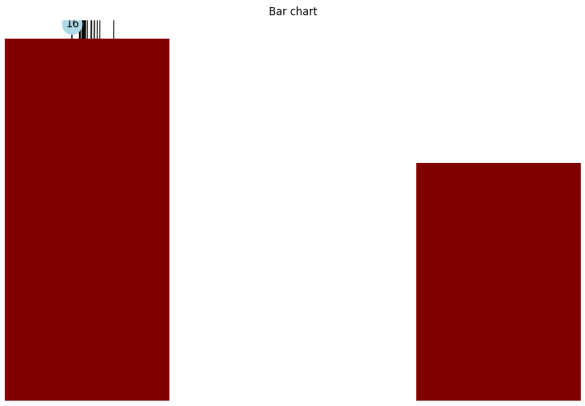


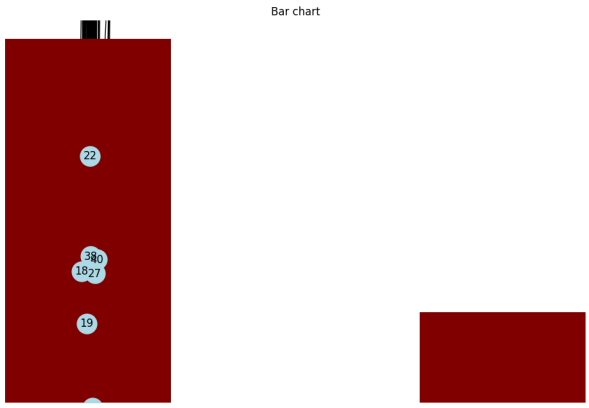


Переглядаючи результати, бачимо що часова різниця є, як прірва.
Отже наш код є мало ефективним. У всіх випадках результат програє.

Якщо наявний простий цикл негативної ваги БЕЛМАН-ФОРД (0.2 ймовірність, неорієнтований)







Переглядаючи результати, бачимо що часова різниця на будю-якій к-ті вершин приблизно вдвічі більша. Отже наш код є мало ефективним, підозрюємо, що вбудований алгоритм швидше виявляє негативні цикли. У всіх випадках результат програє.

Висновки: вбудовані алгторитми є швидшими для Белмана-Форда за будь-яких обставин. Однак, такого не скажеш про Крускала, який показує кращі результати особливо з низькою ймовірністю проведення ребер. Тестуючи з використаннясм негативних ваг, то ми помітили, що різниця незначна. Тестуючи за наявності простих циклів негативної ваги, то ми помітили, що наш алгоритм може перевірити наявність лише в кінці, що сповільнює його, також велика кількість циклів у коді не пришвидшує його.