Дискретная математика для программистов.

Лабораторная работа № 7

Кратчайшие пути на больших графах

Соколов Арсений, Гвозденко Демид (2МО)

Вариант 21

Задание:

Задать простые связные разреженные неориентированные графы с числом вершин 500, 1500, 4500, 13500, 40500 случайным образом.

Граф должен содержать подграф K_7.

Найти двумя способами (по двум разным алгоритмам) кратчайшие пути от выделенной вершины до остальных вершин.

Найти количество итераций. Сравнить с асимптотической сложностью применяемых алгоритмов.

Генерация графа:

```python
import time
import random


1 usage
def generate_graph(num_nodes, max_edges):
    graph = [[0] * num_nodes for _ in range(num_nodes)]
    nodes = list(range(num_nodes))
    visited = set()

    # Добавляем первые 7 вершин графа K7
    k7_nodes = list(range(7))
    visited.update(k7_nodes)

    # Добавляем ребра для графа K7
    k7_edges = [
        (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6),
        (1, 2), (1, 3), (1, 4), (1, 5), (1, 6),
        (2, 3), (2, 4), (2, 5), (2, 6),
        (3, 4), (3, 5), (3, 6),
        (4, 5), (4, 6),
```

```
        (5, 6)
    ]

    for edge in k7_edges:
        node1, node2 = edge
        weight = random.randint(1, 10000000)
        graph[node1][node2] = weight
        graph[node2][node1] = weight

    while len(visited) < num_nodes:
        current_node = random.choice(list(visited))
        remaining_nodes = list(set(nodes) - visited)

        num_edges = random.randint(1, min(max_edges, len(remaining_nodes)))

        for _ in range(num_edges):
            next_node = random.choice(remaining_nodes)
            weight = random.randint(1, 10000000)
```

```
            graph[current_node][next_node] = weight
            graph[next_node][current_node] = weight
            visited.add(next_node)
            remaining_nodes.remove(next_node)

    return graph
```

Алгоритм Дейкстры:

```python
def Dijkstra(N, S, matrix):
    number_of_iterations = 0 # счетчик количества итераций алгоритма
    valid = [True] * N # доступность вершин
    weight = [math.inf] * N # длины путей
    weight[S] = 0
    for k in range(N):
        min_weight = math.inf # текущая минимальная длина пути
        ID_min_weight = -1 # индекс вершины с минимальной длиной пути
        for k in range(N):
            number_of_iterations += 1
            if valid[k] and weight[k] < min_weight:
                min_weight = weight[k]
                ID_min_weight = k
        for z in range(N):
            if weight[ID_min_weight] + matrix[ID_min_weight][z] < weight[z]:
                weight[z] = weight[ID_min_weight] + matrix[ID_min_weight][z]
    valid[ID_min_weight] = False
    print('Кличество итераций: {:,}'.format(number_of_iterations).replace(',', ' '))
    return weight

```

Реализация кода на примерах:

```python
import random
import math
import time
```

```python
graph = generate_graph(n, max_edges)

print("Количество вершин: " + str(n))
start_time = time.time()
ans = Dijkstra(n, 0, graph)
print(ans)
print(f"{(time.time() - start_time)} секунд")
```

Для n = 500 вершин:

```
Количество вершин: 500
Кличество итераций: 250 000
[0, 49, 93, 64, 76, 47, 68, 0, 0, 0, 0, 0, 0, 0, 0, 0
0.03341078758239746 секунд
```

Количество итераций равно 250000

Для n = 1500 вершин:

```
Количество вершин: 1500
Кличество итераций: 2 250 000
[0, 92, 12, 29, 3, 88, 16, 0, 0, 0, 0, 0, 0, 0, 0, 0
0.3160860538482666 секунд
```

Количество итераций равно 2.250.000

Для n = 13500 вершин:

```
Количество вершин: 13500
Кличество итераций: 182 250 000
[0, 94, 58, 39, 41, 91, 80, 0, 0, 0, 0, 0, 0,
25.15864610671997 секунд
```

Количество итераций равно 182.250.000

Для n = 40500 вершин:

```
Количество вершин: 40500
Кличество итераций: 1 640 250 000
[0, 4, 93, 64, 60, 65, 23, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 227.563702583313 секунд
```

Количество итераций равно 1.640.000.000

Можем заметить, что результаты выполнения программы близки к асимптотической сложности алгоритма Дейкстры $O(n^2)$.

Алгоритм Форда-Беллмана:

Асимптотическая сложность алгоритма Форда-Беллмана $O(n^3)$

```
start_time = time.time()
N = 500
M = N
max_edges = 5  # Максимальное число ребер для каждой вершины

graph = generate_graph(N, max_edges)

start = 0
INF = 10 ** 9
F = [INF] * N
```

```
F[start] = 0
number_of_iterations = 0
for k in range(1, N):
    for i in range(N):
        for j in range(N):
            number_of_iterations += 1
            if F[j] + graph[j][i] < F[i]:
                F[i] = F[j] + graph[j][i]
print("%s вершин" % N)
for item in F:
    print(item, end=' ')
print(end='\n')
print()
print(f'Количество итераций: {number_of_iterations}')
print("%s секунд" % (time.time() - start_time))
```

```
500 вершин
0 124473 136617 174754 87215 118100 124688 148358 95725 143670 122160 98080 203774 123255 104836 129247 150966 158645 143954 111753 151546 162653 145895 150990

Количество итераций: 124750000
27.471227169036865 секунд
```

```
500 вершин
0 81065 187780 142454 82207 72820 140789 108234 123978 165920 114150 149916 170145 81074 98596 114522 140129 170616 106438 100461 99908 142346 153289 137848 1

125000000
Количество итераций: 124750000
28.418271780014038 секунд
```

```
C:\Users\Dell\AppData\Local\Programs\Python\Python310\python.exe "C:\Проги\Python\4 sem\Discret\Lab_7\lab.py"
500 вершин
0 114201 131004 158353 82490 135729 83387 122069 117400 129460 61570 108615 77041 140226 113424 91040 118393 113673 174487 67910 141547 107667 144021 109641 533

Количество итераций: 124750000
25.387843132019043 секунд
```

Для пятисот вершин совершено 124750000 итераций, что близко к асимптотической сложности 500^3 = 125000000 итераций. Примем (27,4 + 28,4 + 25,3)/3 = 27,0 секунд в качестве среднего времени выполнения 500^3 итераций. Тогда для графа с $(40500^3/500^3)*27$ = 14348907,7 секунд = 166,08 дня. Поэтому дальше испытания не проводились