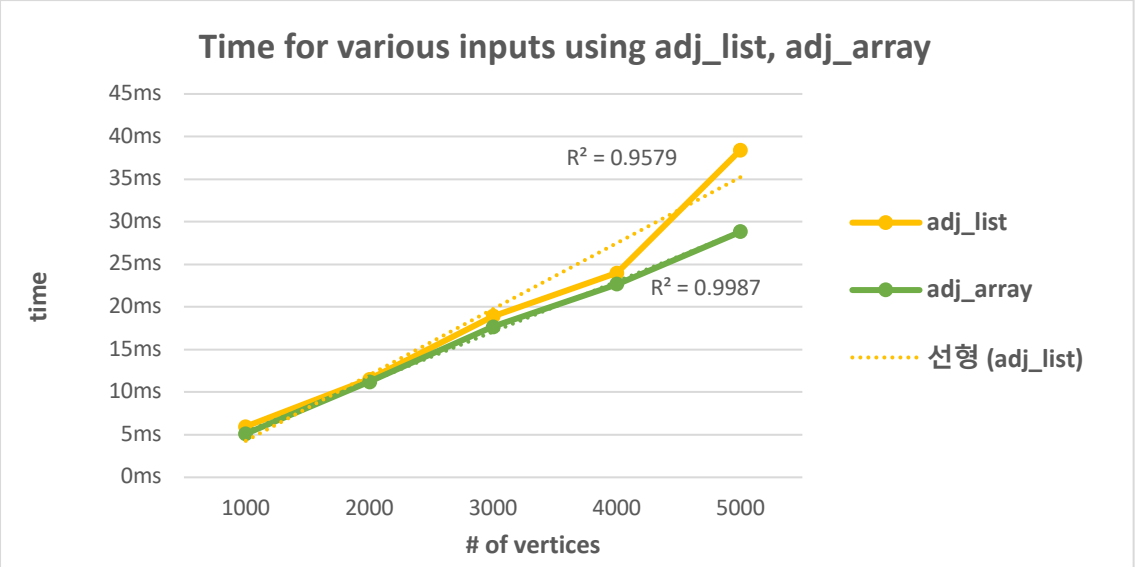
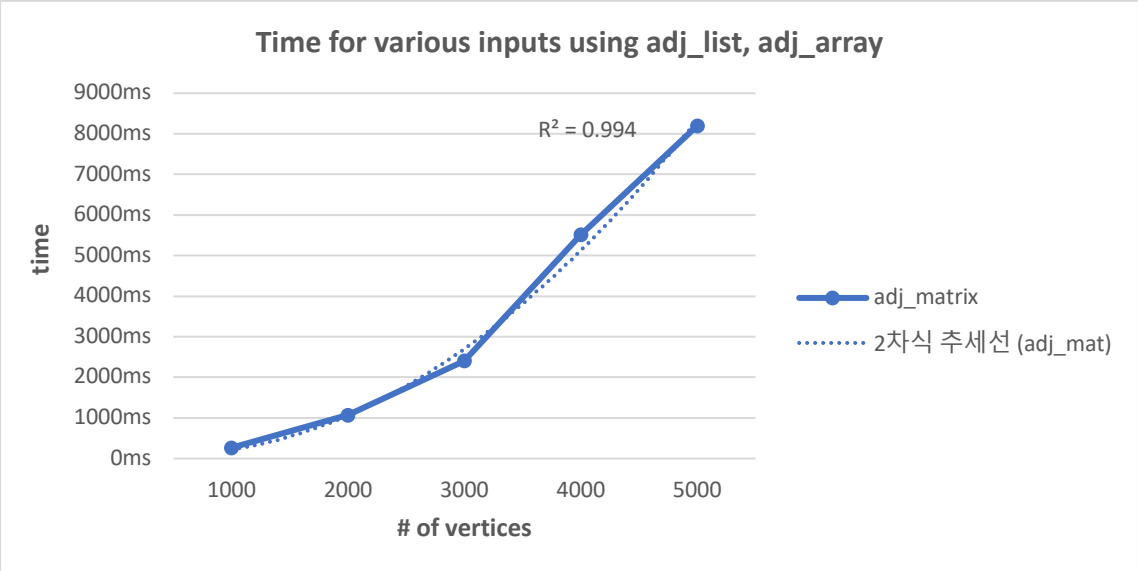
Algorithm homework 3

공과대학 산업공학과
2018-10786 서규혁

1. Measure the time on various inputs (different values of n, sparse and dense, etc.) and discuss the results

# of vertices	adj_list	adj_array	adj_matrix
1000	5.96ms	5.11ms	266.74ms
2000	11.51ms	11.23ms	1072.32ms
3000	18.92ms	17.66ms	2405.9ms
4000	23.98ms	22.67ms	5515.8ms
5000	38.42ms	28.82ms	8199.26ms





Type에 따른 find Strongly Connected Components의 실행시간이 차이가 발생하는 것을 확인할 수 있다. adj_list와 adj_array의 경우는 실행시간이 vertex 갯수에 일차식에 비례하는 것을 볼 수 있고 (이렇게 주장할 수 있는 이유는 추세선의 R-제곱 값이 각각 0.9579, 0.9987로 1에 매우 가깝기 때문), adj_matrix의 경우는 실행시간이 vertex 갯수의 이차식에 비례하는 것을 볼 수 있다 (이렇게 주장할 수 있는 이유는 추세선의 R-제곱 값이 0.994 로 1에 매우 가깝기 때문). 이러한 차이가 발생하는 이유는 adj_list, adj_array와 adj_matrix가 Graph를 표현하는 방식의 차이, 방식의 크기 차이라고 할 수 있다. adj_list, adj_array에서 graph를 표현할 때 사용되는 배열을 기본 세팅하는 과정속에서 배열의 길이가 vertex 수만큼 값을 0으로 초기에 넣어주기때문이고, adj_matrix에서 graph를 표현할 때 vertex 제곱만큼의 값을 0으로 초기화시켜주는 작업이 있으므로 vertex 수의 일차식, 이차식에 비례하여 time complexity를 가진다.

# of Edges	adj_list	adj_array	adj_matrix
3000	7.53ms	5.93ms	268.3ms
30000	61.07ms	86.59ms	268.08ms
300000	6661.72ms	3397.94ms	347.46ms

Sparse와 dense에 따른 time complexity 분석을 시작해보겠다. 타입에 상관없이 Edges의 수가 증가함에 따라 시간이 증가하는데 이는 처리해야하는 vertex, edge가 많아지고 graph를 표현하는 자료의 크기가 커지다보니 당연한 일이라 볼 수 있다. 여기서 주목할 점은 adj_list, adj_array는 edge의 수가 증가함에 따라 time의 증가폭 또한 이루말할 수 없을 정도로 증가한 반면, adj_matrix는 그렇지 않았다는 점이다. 이는 type별로 graph를 어떻게 표현하는지에 영향을 받은 것이다. adj_list, adj_array는 edge들만 표현하다보니, edge의 수가 증가할수록 array의 길이가 점차 길어지게 된다. 그러나 adj_matrix의 경우는 vertex에서 vertex로의 직접적인 이동을 모두 0, 1로 표시한다. 즉 edge로 연결되어있던, 아니던 표시하기때문에 edge의 수에 상관없이 graph를 표현하는 adj_matrix는 (vertex num) x (vertex num) 의 크기를 항상 유지하기 때문이다. 따라서 고정된 vertex num에 대해 graph가 dense해질수록 이에 큰 영향을 받는 type은 adj_list, adj_array이라고 할 수 있다.

2. Write down the environment you run your program and how to run your program.

작업환경은 python 3.8.9이며 math, random, sys, time 을 import했다. 그리고 operator에서 itmegetter을, random에서 randint를 import하여 사용하였다.

해당 프로그램을 run하는 방법은, terminal을 켜고 hw3.py가 있는 경로에 접속한다. 해당 경로에서 input.txt를 담고 있는 directory(예시로 input이라는 이름을 가진 directory라고 하자)를 만든다. input directory 안에는 input.txt가 있다. 그리고

$ python ./hw3.py input/input.txt input/adj_list.txt adj_list

$ python ./hw3.py input/input.txt input/adj_arr.txt adj_arr

$ python ./hw3.py input/input.txt input/adj_mat.txt adj_mat

을 이용하여 실행할 수 있으며 해당 명령들은 Strongly Connected Components를 찾을 때, adjacency list, adjacency array, adjacency matrix 중 무엇을 사용할 것인지에 대한 여부에 따라 선

택하면 된다.

3. Example Running

아래 표는 source code 인 hw3.py 와 같은 경로에 있는 directory input 안에 저장된 4 개의 txt 파일로 Example running 이다.

input.txt

```
10
2 5
3
4
2
2 6
1 7
8
5
1 6 10
7
```

adj_arr.txt

```
1 5 6 7 8
2 3 4
9
10
0.04ms
```

adj_list.txt

```
1 5 6 7 8
2 3 4
9
10
0.04ms
```

adj_mat.txt

```
1 5 6 7 8
2 3 4
9
10
0.05ms
```