HW 2: graph

2018312164 김석진

Problem 1

Code

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

Code explanation

A computer code with text

Description automatically generated with medium confidence

이는 인접 리스트의 노드를 나타내는 linked\_adja 구조체를 정의합니다. 각 노드에는 인접한 정점 정보 (vertex), 해당 정점까지의 간선 비용 (cost), 및 리스트에서 다음 노드를 가리키는 링크 (link)가 포함되어 있습니다.

A screen shot of a computer code

Description automatically generated

리스트의 마지막에 추가할 요소를 위해 메모리를 한당하고 그 값을 넣어준다.

A computer screen with text

Description automatically generated

리스트가 비어 있을 경우 graph\_array의 포인터가 새로 만든 node를 가르키게 하고 이미 end node가 있을 경우 마지막 요소 뒤에 추가한다.

Result

A computer screen shot of a number

Description automatically generated

Problem 2

Code

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Code explanation

A black background with text

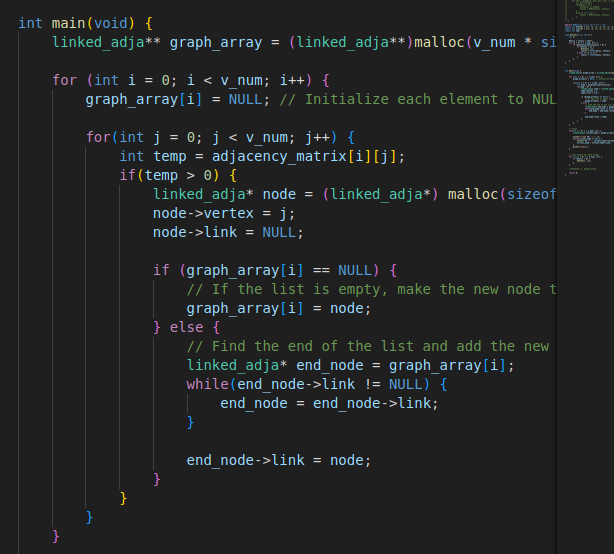
Description automatically generated

Graph 구조체로 vertex와 self reference pointer로 이루어져 있다.

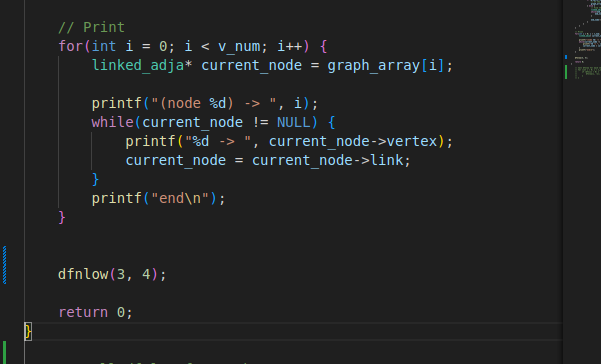
A computer screen shot of a program code

Description automatically generated

Depth first number 을 찾는 코드로 recursively low() 함수를 돌면서 자기 자신 노드 그리고 자식 노드 또는 back edge로 접근할 수 있는 노드의 최소 dfn 값을 찾아 낸다.



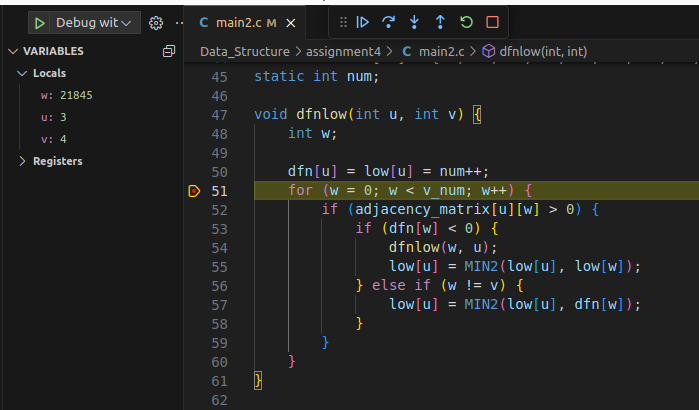
Matrix representation으로 되어 있는 graph를 사용해서 listed representation으로 바꾸는 코드로 for loop을 돌면서 모든 엔트리를 확인하고 linked list으로 연결한다. 1번 문제와 유사한 방법으로 진행했다.



Linked list를 print하고 dfnlow를 사용하는 코드

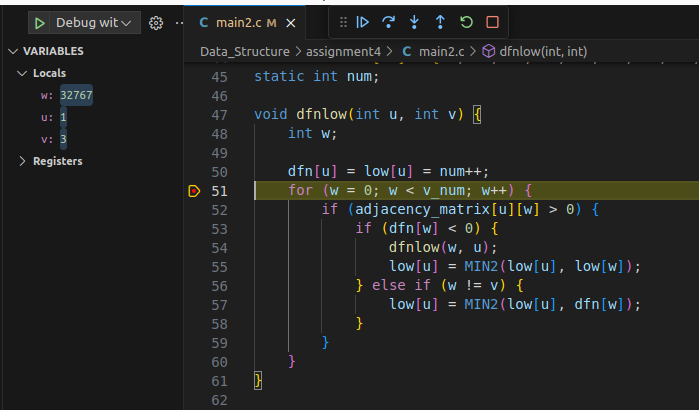
Result

1.



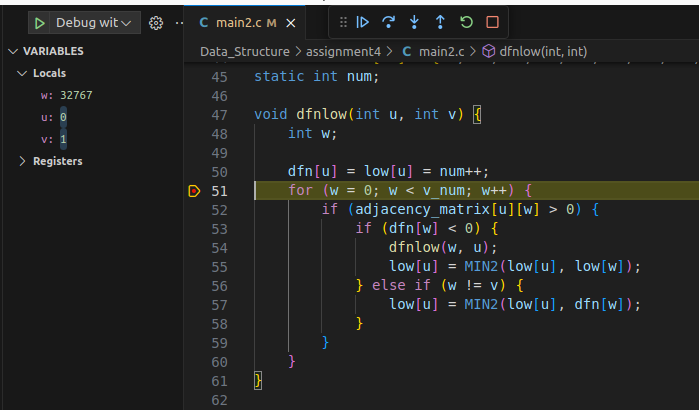
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn |  |  |  | 0 |  |  |  |  |  |  |
| low |  |  |  | 0 |  |  |  |  |  |  |

2.



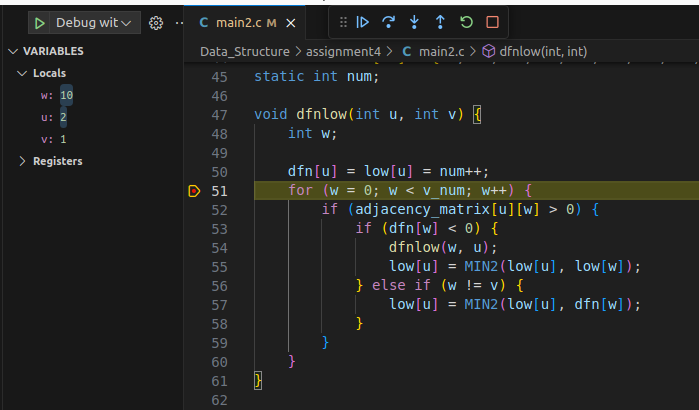
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn |  | 1 |  | 0 |  |  |  |  |  |  |
| low |  | 1 |  | 0 |  |  |  |  |  |  |

3.



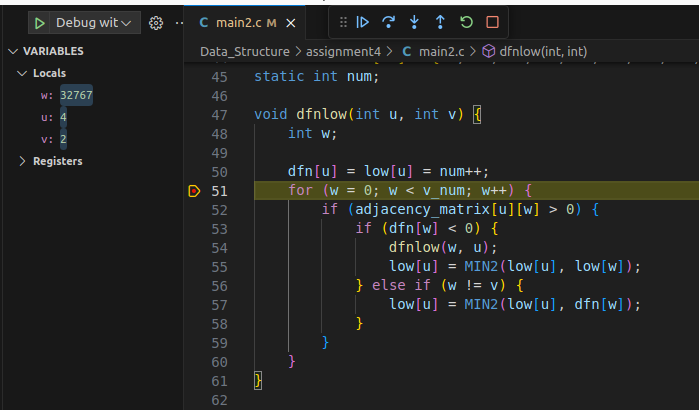
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn | 2 | 1 |  | 0 |  |  |  |  |  |  |
| low | 2 | 1 |  | 0 |  |  |  |  |  |  |

4.



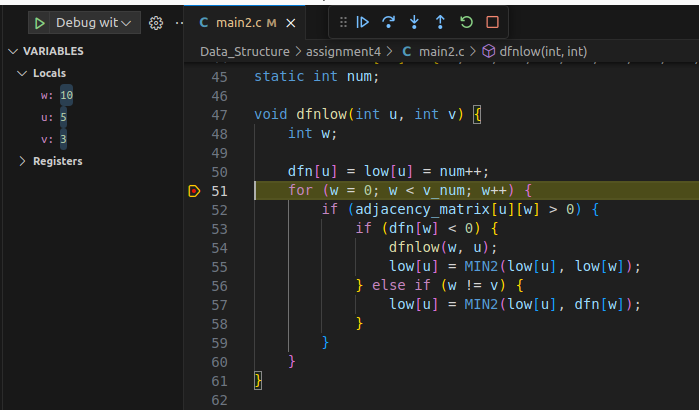
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn | 2 | 1 | 3 | 0 |  |  |  |  |  |  |
| low | 2 | 1 | 0 | 0 |  |  |  |  |  |  |

5.



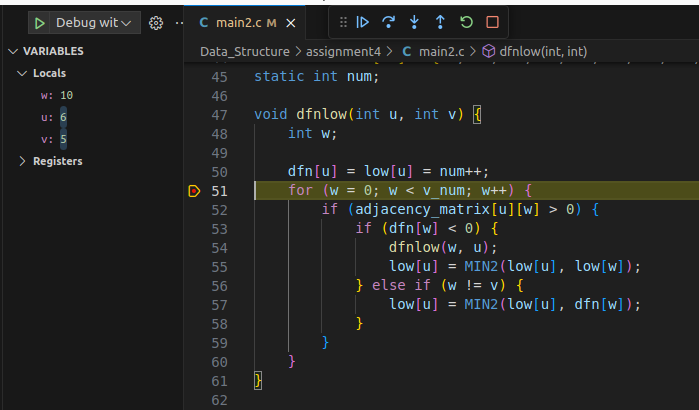
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn | 2 | 1 | 3 | 0 | 4 |  |  |  |  |  |
| low | 2 | 1 | 0 | 0 | 0 |  |  |  |  |  |

6.



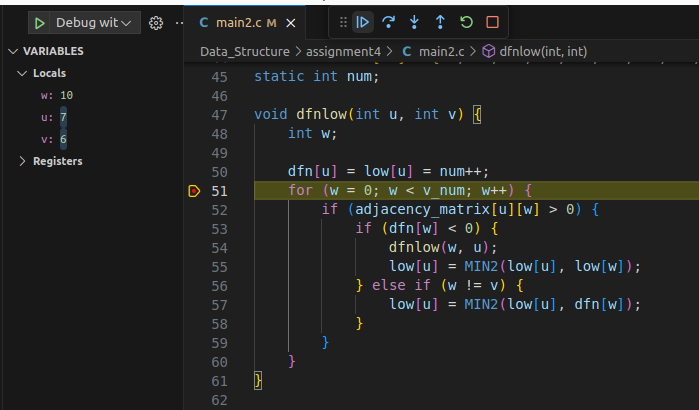
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn | 2 | 1 | 3 | 0 | 4 | 5 |  |  |  |  |
| low | 2 | 1 | 0 | 0 | 0 | 5 |  |  |  |  |

7.



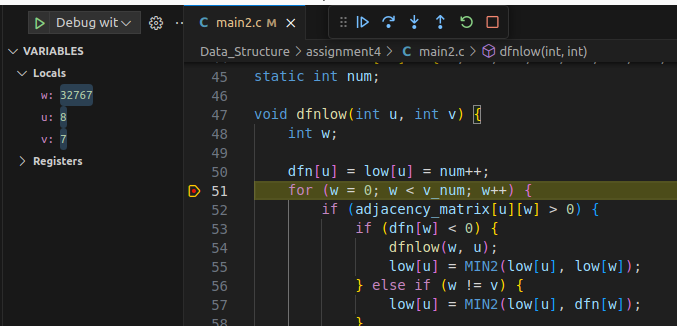
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn | 2 | 1 | 3 | 0 | 4 | 5 | 6 |  |  |  |
| low | 2 | 1 | 0 | 0 | 0 | 5 | 5 |  |  |  |

8.



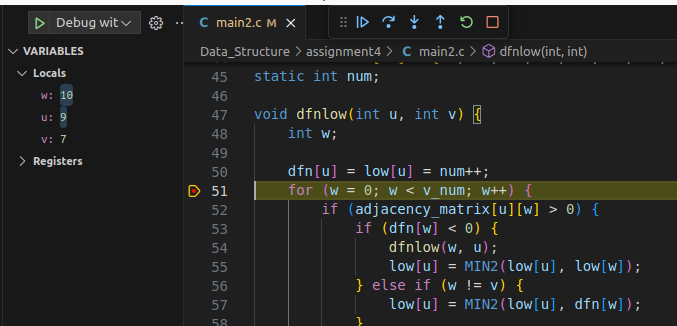
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn | 2 | 1 | 3 | 0 | 4 | 5 | 6 | 7 |  |  |
| low | 2 | 1 | 0 | 0 | 0 | 5 | 5 | 5 |  |  |

9.



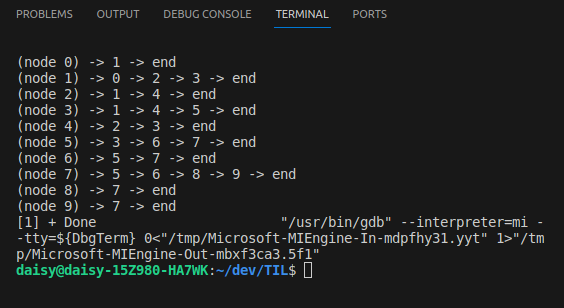
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn | 2 | 1 | 3 | 0 | 4 | 5 | 6 | 7 | 8 |  |
| low | 2 | 1 | 0 | 0 | 0 | 5 | 5 | 5 | 8 |  |

10.



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vertex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| dfn | 2 | 1 | 3 | 0 | 4 | 5 | 6 | 7 | 8 | 9 |
| low | 2 | 1 | 0 | 0 | 0 | 5 | 5 | 5 | 8 | 9 |

Terminal



For loop에 breaking point을 걸어 놓고 코드를 실행하면 10번 멈추는 것을 확인 할 수 있다. 그리고 멈출 때마다 나오는 u 값이 현재 dfs을 하면서 멈추는 노드를 가르킨다. 그 순서를 나열하면 다음과 같다.

3 > 1 > 0 > 2 > 4 > 5 > 6 > 7 > 8 > 9

이를 토대로 spaning tree을 그리면 다음과 같다.

