David Kopp

10-09-15

Comp496 ALG

Extra Credit Midterm1: Topological Sort

Run-Time Analysis of Topological Sort: by methods

TopologicalSort(): n^3 + n^2 + 2n = O(n^3)

1) Initialize sortOrderArr: O(n)

2) Initialize vertexExists: O(n)

Initialize i: O(1)

3) While (vertexExist()): O(n) Total: n^3 + 2n^2 = O(n^3)

Int vertex = nextVertex(dag): O(n^2)

For Loop: O(n)

For Loop: O(n)

sortOrderArr[i] = vertex: O(1)

vertexExists[vertex] = false: O(1)

removeVertex(vertex, dag): O(n)

i++: O(1)

Total Analysis:

1 ) n

2) n

3) n^3 + 2n^2

Total: n^3+ 2n^2 + 2n

Final Analysis: O(n^3)

Programmer’s Code:

/\*

\* Programmer: David Kopp

\* Project #: Extra Credit for Midterm1

\* File Name: TopologicalSort.java

\* Date: 10-9-15

\* Class: Comp496ALG

\* Description: This program uses two hard coded DAGs (n=5 and n=10) and an adjacency matrix to do a topological sort.

\*/

// Defined package for eclipse project (comment out if not using eclipse)

package ExtraCreditMidTerm1;

// Imports

// Object class

public class TopologicalSort {

private int[] sortOrderArr;

private boolean[] vertexExists;

// Object constructor class

public TopologicalSort(int[][] dag)

{

// Initialization of data structures and variables

sortOrderArr = new int[dag[0].length];

vertexExists = new boolean[dag[0].length];

initArr(sortOrderArr);

initArr(vertexExists);

int i = 0;

// Main loop of the algorithm

while (vertexExist())

{

int vertex = nextVertex(dag);

sortOrderArr[i] = vertex;

vertexExists[vertex] = false;

removeVertex(vertex, dag);

i++;

}

}

// Finds the next vertex with no incoming edges

public int nextVertex(int[][] dag)

{

int result = 0;

for (int i = 0; i < dag[0].length; i++)

{

for (int j = 0; j < dag[0].length; j++)

{

result += dag[j][i];

}

if (result == 0 && vertexExists[i])

{

return i;

}

result = 0;

}

return -1; // Error in the graph

}

// Checks to see if there is a vertex left on the graph

public boolean vertexExist()

{

for (int i = 0; i < vertexExists.length; i++)

{

if (vertexExists[i] == true)

{

return true;

}

}

return false;

}

// Prints the sort array

public void printSortArr()

{

System.out.print("S = {");

System.out.print(sortOrderArr[0]);

for (int i = 1; i < sortOrderArr.length; i++)

{

System.out.print("," + sortOrderArr[i]);

}

System.out.println("}\n");

}

// Removes the vertex from the adjacency matrix

private void removeVertex(int vertex, int[][] dag)

{

for (int i = 0; i < dag[0].length; i++)

{

dag[vertex][i] = 0;

}

}

// Initializes the sort array

private void initArr(int[] arr)

{

for (int i = 0; i < arr.length; i++)

{

arr[i] = -1;

}

}

// Initializes the vertex exists array

private void initArr(boolean[] arr)

{

for (int i = 0; i < arr.length; i++)

{

arr[i] = true;

}

}

// Main method

public static void main(String[] args)

{

// Data Structures

int[][] dag1 ={{0,1,0,0,1},{0,0,0,1,0},{0,1,0,1,0},{0,0,0,0,1},{0,0,0,0,0}};

int[][] dag2 = {{0,1,0,0,0,0,0,1,0,1},{0,0,0,0,0,0,0,1,0,0},

{0,1,0,0,0,0,0,0,0,0},{0,1,1,0,1,1,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0},{0,1,0,0,0,0,1,1,0,0},

{0,0,0,0,1,0,0,0,0,0},{0,0,0,0,0,0,0,0,1,0},

{0,0,0,0,0,0,1,0,0,0},{0,0,0,0,0,0,0,0,1,0}};

TopologicalSort ts = new TopologicalSort(dag1);

TopologicalSort ts2 = new TopologicalSort(dag2);

**System**.***out***.println("Topological Sort: (n = 5)");

ts.printSortArr();

**System**.***out***.println("Topological Sort: (n = 10)");

ts2.printSortArr();

}

}

Test Case#1 Input:

0 1 0 0 1

0 0 0 1 0

0 1 0 1 0

0 0 0 0 0

0 0 0 0 0

Test Case#1 Output:

Topological Sort: (n = 5)

S = {0,2,1,3,4}

Test Case#2 Input:

0 1 0 0 0 0 0 1 0 1

0 0 0 0 0 0 0 1 0 0

0 1 0 0 0 0 0 0 0 0

0 1 1 0 1 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 1 0 0 0 0 1 1 0 0

0 0 0 0 1 0 0 0 0 0

0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 0 1 0

Test Case#2 Output:

Topological Sort: (n = 10)

S = {0,3,2,5,1,7,9,8,6,4}

