Let s1,s2,...,sk be all the sources and t1,t2,...,tl be all the targets in G (these can be identified in O(|V|+|E|) time by Part (a)). We convert G to a new DAG G' whose vertex set contains two additional vertices s and t. We add the edges (s,si) for all i = 1,2,..., k and also the edges (tj ,t) for all j = 1,2,...,l. G' is a DAG with a unique source s and a unique target t. Moreover, the count of all (si ,tj) paths (for all i, j) in G is the same as the count of all (s,t) paths in G'. The size of G' continues to remain O(|V|+|E|). We make a topological sorting of the vertices in G'. This can be done in O(|V|+|E|) time. Let the listing be s = v0, v1, v2,..., vn,t = vn+1. We use an array C indexed by the vertices in G' to store the count of paths from s to the vertices.

Initialize C[v0] = 1 and C[vi] = 0 for all i = 1,2,3,...,n+1.

For i = 0,1,2,...,n {

For all edges (vi, vj) in G', set C[vj] = C[vj] +C[vi].

}

Return C[vn+1].

Since there are no back edges (that is, edges (vi, vj) with i > j), the for loop does not miss a path from s to t. With the adjacency list representation of G', this phase can again be finished in O(|V|+|E|) time. The introduction of the new vertices s,t could have been avoided. In that case, we start by setting C[si] = 1 for all the sources si in G. At the end, we return C[t1] +C[t2] +···+C[tl]. However, a topological sorting of G is necessary for the correctness of this algorithm.

Topological Sorting

```
#include <stdio.h>


int main(){
	int i,j,k,n,a[10][10],index[10],flag[10],count=0;


	printf("Enter the no of vertices:\n");
	scanf("%d",&n);


	printf("Enter the adjacency matrix:\n");
	for(i=0;i<n;i++){
		printf("Enter row %d\n",i+1);
		for(j=0;j<n;j++)
			scanf("%d",&a[i][j]);
	}
```

```
        for(i=0;i<n;i++){
    index[i]=0;
    flag[i]=0;
  }

  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      index[i]=index[i]+a[j][i];

  printf("\nThe topological order is:");

  while(count<n){
    for(k=0;k<n;k++){
      if((index[k]==0) && (flag[k]==0)){
        printf("%d ",(k+1));
        flag [k]=1;
      }

      for(i=0;i<n;i++){
        if(a[i][k]==1)
          index[k]--;
      }
    }

    count++;
  }

  return 0;
}
```

Input and Output

Enter the no of vertices:

4

Enter the adjacency matrix:

Enter row 1

0 0 0 1

Enter row 2

0 0 0 1

Enter row 3

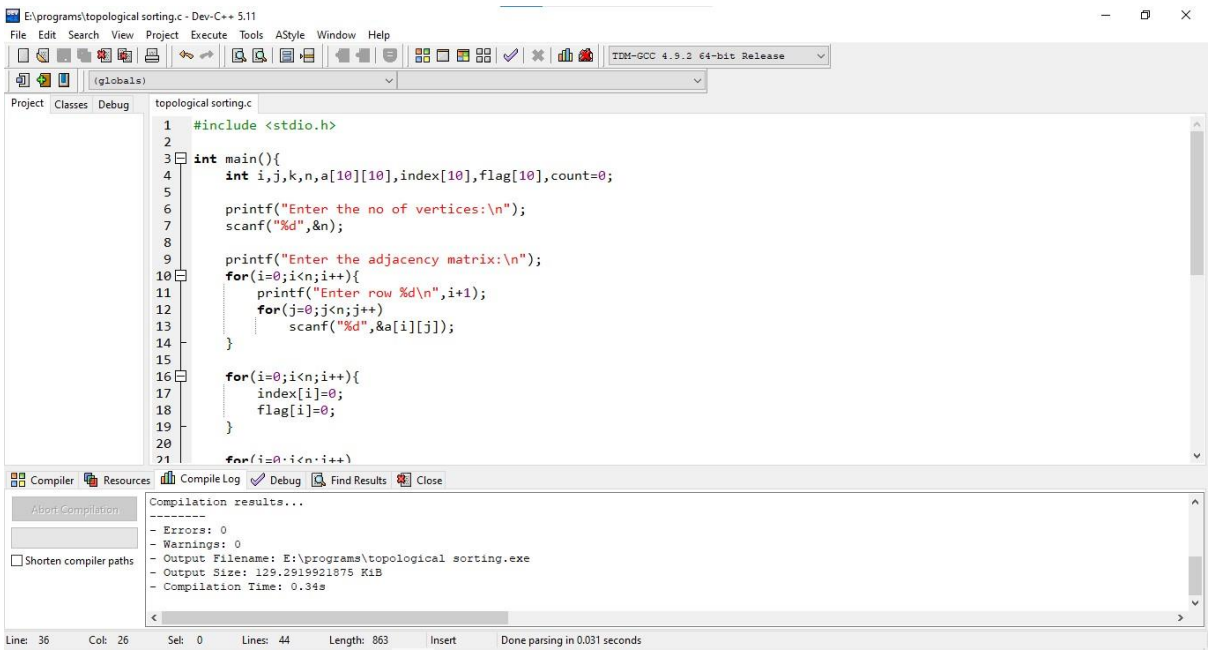
0 0 0 0

Enter row 4

0 1 1 0

The topological order is:1 2 3 4

Screenshots



E:\programs\topological sorting.c - Dev-C++ 5.11

topological sorting.c

```c
    }

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            index[i]=index[i]+a[j][i];

    printf("\nThe topological order is:");

    while(count<n){
        for(k=0;k<n;k++){
            if((index[k]==0) && (flag[k]==0)){
                printf("%d ",(k+1));
                flag [k]=1;
            }

            for(i=0;i<n;i++){
                if(a[i][k]==1)
                    index[k]--;
            }
        }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: E:\programs\topological sorting.exe
- Output Size: 129.2919921875 KiB
- Compilation Time: 0.34s

Line: 36 Col: 32 Sel: 0 Lines: 44 Length: 863 Insert Done parsing in 0.031 seconds

E:\programs\topological sorting.c - Dev-C++ 5.11

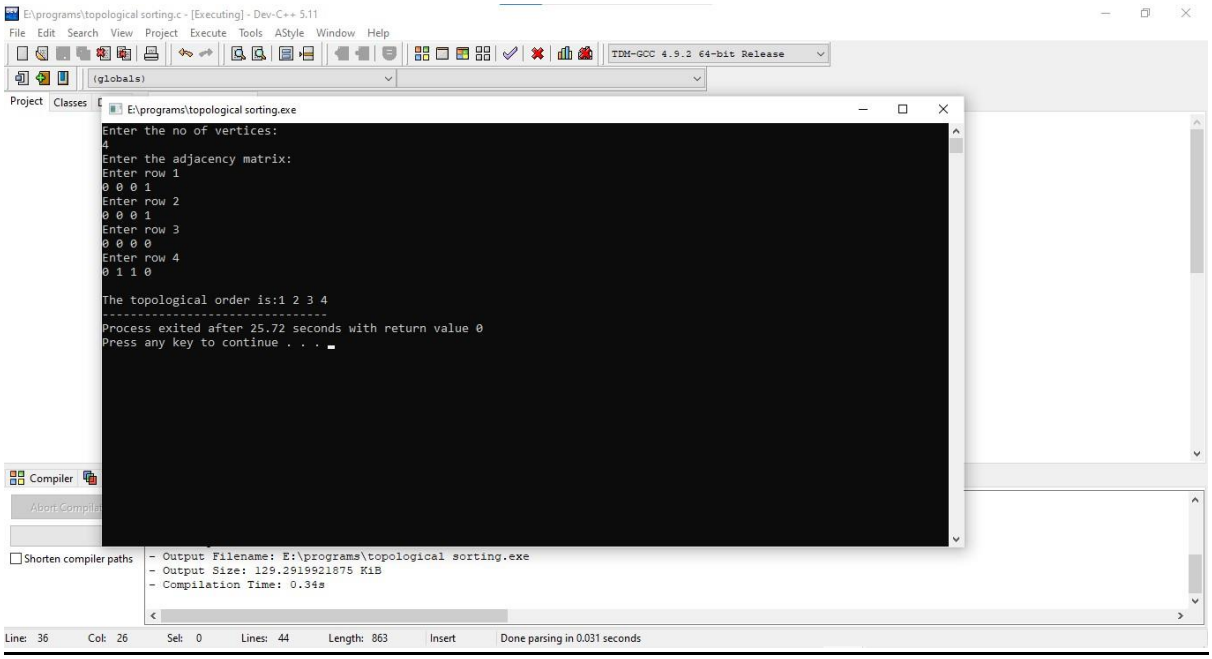
topological sorting.c

```c
    printf("\nThe topological order is:");

    while(count<n){
        for(k=0;k<n;k++){
            if((index[k]==0) && (flag[k]==0)){
                printf("%d ",(k+1));
                flag [k]=1;
            }

            for(i=0;i<n;i++){
                if(a[i][k]==1)
                    index[k]--;
            }
        }

        count++;
    }

    return 0;
}
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: E:\programs\topological sorting.exe
- Output Size: 129.2919921875 KiB
- Compilation Time: 0.34s

Line: 36 Col: 32 Sel: 0 Lines: 44 Length: 863 Insert Done parsing in 0.031 seconds


E:\programs\topological sorting.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes
E:\programs\topological sorting.exe
Enter the no of vertices:
4
Enter the adjacency matrix:
Enter row 1
0 0 0 1
Enter row 2
0 0 0 1
Enter row 3
0 0 0 0
Enter row 4
0 1 1 0
The topological order is:1 2 3 4

Process exited after 25.72 seconds with return value 0
Press any key to continue . . .
Compiler
Shorten compiler paths
- Output Filename: E:\programs\topological sorting.exe
- Output Size: 129.2919921875 KiB
- Compilation Time: 0.34s
Line: 36 Col: 26 Sel: 0 Lines: 44 Length: 863 Insert Done parsing in 0.031 seconds