[백준] 9466번: 텀 프로젝트

통계 수정 삭제

yoool · 방금 전

0

2023_Autumn_Algorithm_Study

목록 보기 9/9

> 9466번: 텀 프로젝트
> 프로젝트 팀을 구성하기 위해, 모든 학생들은 프로젝트를 함께하고 싶은 학생을 선택해야 한다. 학생들이(s1, s2, ..., sr)이라 할 때, r=1이고 s1이 s1을 선택하는 경우나, s1이 s2를 선택하고, s2가 s3를 선택하고,..., sr-1이 sr을 선택하고, sr이 s1을 선택하는 경우에만 한 팀이 될 수 있다. 주어진 선택의 결과를 보고 어느 프로젝트 팀에도 속하지 않는 학생들의 수를 계산하는 프로그램을 작성하라.

이 문제를 풀기 위해 방향그래프 위상 정렬을 사용했다.
진입차수가 0인 것을 먼저 찾은 후 큐에 넣고 해당 노드의 간선을 그래프에서 제거하는 과정을 반복한다.

수업 시간에 사용했던 알고리즘으로 문제를 풀려고 했는데
예제는 잘 돌아가는 거 같은데 백준에서 시간초과가 발생했다..... ㅠㅜㅠ

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

#define NUMV_MAX 100000
#define IN -1
#define OUT 1
```

```
typedef struct Edge {
    int orgIdx;
    int desIdx;
} Edge;

typedef struct IncidentEdge {
    int edgeIdx;
    struct IncidentEdge* next;
} IncidentEdge;

typedef struct Vertex {
    int vName;
    int inDegree;
    IncidentEdge* InIEdges;
    IncidentEdge* OutIEdges;
} Vertex;

typedef struct Graph {
    Vertex* vertices;
    Edge* edges;
} Graph;

typedef struct Node {
    int vIdx;
    struct Node* next;
} Node;

typedef struct Queue {
    Node* front;
    Node* rear;
    int size;
} Queue;

Graph G;
int n, m;
int topOrder[NUMV_MAX + 1];
Queue Q;

void insertVertex(int vName, int vIdx) {
    G.vertices[vIdx].vName = vName;
    G.vertices[vIdx].InIEdges = NULL;
    G.vertices[vIdx].OutIEdges = NULL;
    G.vertices[vIdx].inDegree = 0;
}

int index(int vName) {
    for (int i = 0; i < n; i++) {
        if (G.vertices[i].vName == vName)
            return i;
    }
    return -1;
}

void insertDirectedEdge(int uName, int wName, int eIdx) {
    int u = index(uName);
    int w = index(wName);

    G.edges[eIdx].orgIdx = u;
```

```
    G.edges[eIdx].desIdx = w;

    IncidentEdge* outEdge = (IncidentEdge*)malloc(sizeof(IncidentEdge));
    outEdge->edgeIdx = eIdx;
    outEdge->next = G.vertices[u].OutIEdges;
    G.vertices[u].OutIEdges = outEdge;

    IncidentEdge* inEdge = (IncidentEdge*)malloc(sizeof(IncidentEdge));
    inEdge->edgeIdx = eIdx;
    inEdge->next = G.vertices[w].InIEdges;
    G.vertices[w].InIEdges = inEdge;

    G.vertices[w].inDegree++;
}


void buildGraph() {
    scanf("%d", &n);
    G.vertices = (Vertex*)malloc((n + 1) * sizeof(Vertex));
    G.edges = (Edge*)malloc((n + 1) * sizeof(Edge));

    for (int i = 0; i < n; i++)
        insertVertex(i + 1, i);

    for (int i = 0; i < n; i++) {
        int num;
        scanf("%d", &num);
        insertDirectedEdge(i + 1, num, i);
    }
}

int isEmpty() {
    return Q.size == 0;
}

void initQueue(Queue* q) {
    q->front = NULL;
    q->rear = NULL;
    q->size = 0;
}

void enqueue(int vIdx) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->vIdx = vIdx;
    newNode->next = NULL;

    if (isEmpty())
        Q.front = Q.rear = newNode;
    else {
        Q.rear->next = newNode;
        Q.rear = newNode;
    }
    Q.size++;
}

int dequeue(void) {
    if (isEmpty())
        return -1;
```

```
    int vIdx = Q.front->vIdx;
    Node* temp = Q.front;
    //Q.front = Q.front->next;
    if (Q.front == Q.rear) {
        Q.front = NULL;
        Q.rear = NULL;
    }
    else
        Q.front = Q.front->next;

    free(temp);
    Q.size--;
    return vIdx;
}

void topologicalSort() {
    initQueue(&Q);
    //Q.front = Q.rear = NULL;

    for (int i = 0; i < n; i++) {
        G.vertices[i].inDegree = 0;

        IncidentEdge* current = G.vertices[i].InIEdges;
        while (current != NULL) {
            G.vertices[i].inDegree++;
            current = current->next;
        }

        if (G.vertices[i].inDegree == 0)
            enqueue(i);
    }

    int t = 0;
    while (!isEmpty()) {
        int u = dequeue();
        topOrder[t++] = u;

        IncidentEdge* current = G.vertices[u].OutIEdges;
        while (current != NULL) {
            int w = G.edges[current->edgeIdx].desIdx;
            G.vertices[w].inDegree--;

            if (G.vertices[w].inDegree == 0)
                enqueue(w);

            current = current->next;
        }
    }

    if (t <= n) {
        printf("%d\n", t);
        topOrder[0] = 0;
    }
    else
        topOrder[0] = 1;
}
```

```
int main() {
    int T;
    scanf("%d", &T);

    for (int i = 0; i < T; i++) {
        topOrder[0] = 0;
        buildGraph();
        topologicalSort();
        free(G.vertices);
        free(G.edges);
    }

    return 0;
}
```

김지율

이전 포스트
[백준] 5567번: 결혼식

0개의 댓글

댓글을 작성하세요

댓글 작성