|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』과제 보고서 | | | |
| 제목 | 10장 실습( ) 과제( O ) | 제출일자 | 2022.  12 .    07 . |
| 학번 | 201911608 | 이름 | 김 지환 |

트랜지스터 회로로 ib맥스, 선형동착, 차단 동작, 포화동작 등

|  |
| --- |
| 02. 인접 행렬이 {0, 1, 0, 0}, {1, 0, 1, 1}, {0, 1, 0, 0}, {0, 1, 0, 0} 이라면 여기에 대응되는 인접리스트를 그려라. |
| 04. 정점의 개수를 n, 간선의 개수가 e인 그래프를 인접 리스트로 표현하였을 경우, 인접 리스트 상의 총 노드의 개수는 ?  ⓵ e개 ⓶ 2e개 ⓷ n개 ⓸ 2n개  -> 그래프가 v개의 정점을 가질 경우 2개의 v 사이에 하나의 간선(e)로 표현 하기 때문에 2e이다.  -> 방향 그래프의 경우 양방향은 2e로 표현, 단방향은 e로 표현하므로 정점은 총 e개 이다. |
| 06. 다음 그래프를 인접 행렬과 인접 리스트로 표현해보자. |
| 08. 정점 V = {1, 2, 3, 4, 5}이고, 간선 E = {<1, 2>, <1, 3>, <1, 4>, <2, 1>, <2, 3>, <2, 5>, <3, 1>, <3, 2>, <3, 4>, <3, 5>, <4, 2>, <5, 1>, <5, 3>}으로 정의되는 방향 그래프를 그려라. |
| 10. 만약 그래프가 인접 리스트로 표현되어 있다고 가정하고 앞의 문제를 다시 작성하라    (1) 주어진 정점의 진출차수 (out-degree)을 계산하는 함수를 작성하라. 진출차수란 어떤 정점에서 출발하여 외부가 나가는 간선의 개수이다. 이 함수의 시간 복잡도는?    -> 연결된 간선만큼만 확인하면 되므로 O(E)이다.  (2) 주어진 정점의 진입차수 (in-degree)을 계산하는 함수를 작성하라. 진입차수란 어떤 정점으로 들어오는 간선의 개수이다. 이 함수의 시간 복잡도는 ?    -> 어떤 u에 들어오는 각각의 연결된 v의 간선들을 다 뒤져봐야하므로 O(V+E)이다.  (3) 그래프 안에 있는 간선들의 개수를 계산하는 함수를 작성하라. 이 함수의 시간 복잡도는?    -> out-degree이므로 O(E)이다.  #include <stdio.h>  #include <stdlib.h>  #define MAX\_VERTICES 50  typedef struct GraphNode {  int vertex;  struct GraphNode\* link;  } GraphNode;  typedef struct GraphType {  int n;  GraphNode\* adj\_list[MAX\_VERTICES];  } GraphType;  void init(GraphType\* g) {  int v;  g->n = 0;  for (v = 0; v < MAX\_VERTICES; v++)  g->adj\_list[v] = NULL;  }  void insert\_vertex(GraphType\* g, int v) {  if (((g->n) + 1) > MAX\_VERTICES) {  fprintf(stderr, "그래프: 정점의 개수 초과");  return;  }  g->n++;  }  void insert\_edge(GraphType\* g, int u, int v) {  GraphNode\* node;  if (u >= g->n || v >= g->n) {  fprintf(stderr, "그래프: 정점 번호 오류");  return;  }  node = (GraphNode\*)malloc(sizeof(GraphNode));  node->vertex = v;  node->link = g->adj\_list[u];  g->adj\_list[u] = node;  }  void print\_adj\_list(GraphType\* g) {  for (int i = 0; i < g->n; i++) {  GraphNode\* p = g->adj\_list[i];  printf("정점 %d의 인접 리스트 ", i);  while (p != NULL) {  printf("-> %d ", p->vertex);  p = p->link;  }  printf("\n");  }  }  void in\_degree(GraphType\* g) {  int\* in = (int\*)malloc(sizeof(int) \* (g->n));  for (int i = 0;i < g->n;i++) in[i] = 0;  for (int i = 0; i < g->n; i++) {  GraphNode\* p = g->adj\_list[i];  while (p != NULL) {  in[p->vertex]++;  p = p->link;  }  }  printf("in-degree 출력\n");  for (int i = 0; i < g->n;i++) {  printf("%d번 정점으로 들어가는 in-dgree : %d\n", i, in[i]);  }  free(in);  }  void out\_degree(GraphType\* g) {  int\* out = (int\*)malloc(sizeof(int) \* (g->n));  for (int i = 0; i < g->n; i++) {  GraphNode\* p = g->adj\_list[i];  int cnt = 0;  for (p, cnt; p != NULL; p = p->link, cnt++);  out[i] = cnt;  }  printf("out-degree 출력\n");  for (int i = 0; i < g->n;i++) {  printf("%d번 정점에서 나가는 out-degree : %d\n", i, out[i]);  }  free(out);  }  void cnt\_edge(GraphType\* g) {  int cnt = 0;  for (int i = 0; i < g->n; i++) {  GraphNode\* p = g->adj\_list[i];  while (p != NULL) {  p = p->link;  cnt++;  }  }  printf("간선의 수 = %d\n", cnt);  }  int main() {  GraphType\* g;  g = (GraphType\*)malloc(sizeof(GraphType));  init(g);  for (int i = 0;i < 6;i++)  insert\_vertex(g, i);  insert\_edge(g, 0, 1);  insert\_edge(g, 2, 4);  insert\_edge(g, 3, 1);  insert\_edge(g, 4, 2);  print\_adj\_list(g);  out\_degree(g);  in\_degree(g);  cnt\_edge(g);  free(g);  return 0;  } |
| 12. 하드 디스크에 파일로 그래프의 인접 행렬이 저장되어 있다고 가정하고 다음과 같은 함수를 작성하라. 그래프 파일의 형식은 다음과 같다.  4 // 정점의 개수  0 1 1 1 // 인접 행렬  1 0 1 1  1 1 0 1  1 1 1 0  read\_graph\_mat (GraphType\* g, char\* name) : 이름이 name인 그래프 파일을 읽어서 그래프 g의 인접 행렬에 저장  write\_graph\_mat(GraphType\* g, char \*name) : 그래프 g의 인접 행렬을 이름이 name인 그래프 파일에 저장  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #define MAX\_VERTICES 50  typedef struct GraphType {  int n;  int adj\_mat[MAX\_VERTICES][MAX\_VERTICES];  } GraphType;  void init(GraphType\* g) {  int r, c;  g->n = 0;  for (r = 0; r < MAX\_VERTICES; r++)  for (c = 0; c < MAX\_VERTICES; c++)  g->adj\_mat[r][c] = 0;  }  void insert\_vertex(GraphType\* g, int v) {  if (((g->n) + 1) > MAX\_VERTICES) {  fprintf(stderr, "그래프: 정점의 개수 초과");  return;  }  g->n++;  }  void insert\_edge(GraphType\* g, int start, int end) {  if (start >= g->n || end >= g->n) {  fprintf(stderr, "그래프: 정점 번호 오류");  return;  }  g->adj\_mat[start][end] = 1;  g->adj\_mat[end][start] = 1;  }  void print\_adj\_mat(GraphType\* g) {  for (int i = 0; i < g->n; i++) {  for (int j = 0; j < g->n; j++) {  printf("%2d ", g->adj\_mat[i][j]);  }  printf("\n");  }  }  void read\_graph\_mat(GraphType\* g, char\* name) {  printf("\n\n\*\*\* %s 파일을 읽어 왔습니다. \*\*\*\n\n", name);  int n;  int arr[101][101] = { 0, };  FILE\* fp;  fp = fopen(name, "r");  fscanf(fp, "%d", &n);  for (int i = 0;i < n;i++) insert\_vertex(g, i);  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  fscanf(fp, "%d", &arr[i][j]);  if (arr[i][j]) insert\_edge(g, i, j);  //fprintf(op, "%d ", arr[i][j]);  }  //fprintf(op, "\n");  }  fclose(fp);  }  void write\_graph\_mat(GraphType\* g, char\* name) {  printf("\n\n\*\*\* %s 파일에 저장 후 종료하겠습니다. \*\*\*\n\n", name);  FILE\* fp;  fp = fopen(name, "w");  for (int i = 0; i < g->n; i++) {  for (int j = 0; j < g->n; j++) {  fprintf(fp, "%2d ", g->adj\_mat[i][j]);  }  fprintf(fp, "\n");  }  fclose(fp);  }  int main() {  GraphType\* g;  g = (GraphType\*)malloc(sizeof(GraphType));  init(g);  read\_graph\_mat(g, "input.txt");  print\_adj\_mat(g);  write\_graph\_mat(g, "output.txt");  free(g);  return 0;  }  실행결과 |
| 14. 연결된 그래프 G의 간선들 중에서 그 간선을 제거하면 연결이 끊어지는 간선 (u, v)를 브리지(bridge)라고 한다. 주어진 그래프에서 브리지를 찾아내는 함수를 작성하라.    #include <stdio.h>  #include <stdlib.h>  #define MAX\_VERTICES 50  int visited[MAX\_VERTICES];  typedef struct GraphNode {  int vertex;  struct GraphNode\* link;  } GraphNode;  typedef struct GraphType {  int n;  GraphNode\* adj\_list[MAX\_VERTICES];  } GraphType;  void init(GraphType\* g) {  int v;  g->n = 0;  for (v = 0; v < MAX\_VERTICES; v++)  g->adj\_list[v] = NULL;  }  void insert\_vertex(GraphType\* g, int v) {  if (((g->n) + 1) > MAX\_VERTICES) {  fprintf(stderr, "그래프: 정점의 개수 초과");  return;  }  g->n++;  }  void insert\_edge(GraphType\* g, int u, int v) {  GraphNode\* node;  if (u >= g->n || v >= g->n) {  fprintf(stderr, "그래프: 정점 번호 오류");  return;  }  node = (GraphNode\*)malloc(sizeof(GraphNode));  node->vertex = v;  node->link = g->adj\_list[u];  g->adj\_list[u] = node;  }  void print\_adj\_list(GraphType\* g) {  for (int i = 0; i < g->n; i++) {  GraphNode\* p = g->adj\_list[i];  printf("정점 %d의 인접 리스트 ", i);  while (p != NULL) {  printf("-> %d ", p->vertex);  p = p->link;  }  printf("\n");  }  }  void bridge(GraphType\* g, int check) {  int c = 2;  for (int i = 0; i < g->n;i++) {  visited[i] = c;  printf("%d정점 삭제 - ", i);  int cnt = cnt\_node(g, 0, c++);  if (cnt == 0) printf("0 -> cnt = 1\n");  else printf("cnt = %d\n", cnt);    if (cnt == check) printf("해당 정점을 연결하는 간선은 브리지가 아닙니다.\n\n");  else printf("해당 정점을 연결하는 간선은 브리지입니다.\n\n");  }  }  int cnt\_node(GraphType\* g, int v, int c){  GraphNode\* p = g->adj\_list[v];  int cnt=0;  while (p != NULL) {  if (visited[p->vertex] != c) {  printf("%d -> ", p->vertex);  visited[p->vertex] = c;  cnt += cnt\_node(g, p->vertex, c)+1;  }  p = p->link;  }  return cnt;  }  int main() {  int arr[10][10] = {  {0,1}, {1,0,1,1},{0,1,0,0,1},{0,1,0,0,1,1},{0,0,1,1},  {0,0,0,1,0,0,1,1},{0,0,0,0,0,1,0,1},{0,0,0,0,0,1,1,0,1,1},  {0,0,0,0,0,0,0,1},{0,0,0,0,0,0,0,1}  };    GraphType\* g;  g = (GraphType\*)malloc(sizeof(GraphType));  init(g);    for (int i = 0; i < 10; i++) insert\_vertex(g, i);  for (int i = 0; i < 10; i++) {  for (int j = 0; j < 10; j++) {  if(arr[i][j]) insert\_edge(g, i, j);  }  }  print\_adj\_list(g);  printf("\n\*\*\* 기존 리스트 개수 \*\*\*\n");  int cnt = cnt\_node(g, 0, 1);  printf("\n\n");  bridge(g, cnt-1);  free(g);  return 0;  }  -> 브리지가 아닌 경우 (0, 2, 4, 6, 8, 9)      -> 브리지인 경우 (1, 3, 5, 7)    실행결과 |