切换行号显示

```
// WeGraph.c: an adjacency matrix implementation of a
weighted graph
#include <stdio.h>
#include <stdlib.h>
#include "WeGraph.h"

struct graphRep {
    int nV;       // #vertices
    int nE;       // #edges
    Weight **edges;  // matrix of weights
};

Graph newGraph(int numVertices) {
    Graph g = NULL;
    if (numVertices < 0) {
       fprintf(stderr, "newgraph: invalid number of
vertices\n");
    }
    else {
        g = malloc(sizeof(struct graphRep));
        if (g == NULL) {
            fprintf(stderr, "newGraph: out of memory\n");
            exit(1);
        }
        g->edges = malloc(numVertices * sizeof(int *));
        if (g->edges == NULL) {
            fprintf(stderr, "newGraph: out of memory\n");
            exit(1);
        }
        int v;
        for (v = 0; v < numVertices; v++) {
            g->edges[v] = malloc(numVertices *
sizeof(int));
            if (g->edges[v] == NULL) {
                fprintf(stderr, "newGraph: out of
memory\n");
                exit(1);
            }
            int j;
            for (j = 0; j < numVertices; j++) {
                g->edges[v][j] = NOWEIGHT;
            }
        }
        g->nV = numVertices;
        g->nE = 0;
    }
    return g;
}

void freeGraph(Graph g) {
```

貌似跟普通的graph区别不大
只是在Edge结构中多了一个Weight

二维数组中存储权值
而对于普通的graph来说，二维数组存储的都是0,1，表示是否相连

注意对于malloc的使用
- struct级别
- 二维数组指针级别
- 二维数组每一行都是int数组

前面malloc的都要free掉

```
    if (g != NULL) {
        int i;
        for (i = 0; i < g->nV; i++) {
            free(g->edges[i]);           // free the
mallocs for each row ...
        }
        free(g->edges);                  // now the malloc
for the edges array ...
        free(g);                         // now the malloc
for the graph rep
    }
    return;
}

static int validV(Graph g, Vertex v) { // checks if v is
in graph
    return (v >= 0 && v < g->nV);
}

Edge newEdge(Vertex v, Vertex w, Weight x) { // create an
edge from v to w
    Edge e = {v, w, x};
    return e;
}

void showEdge(Edge e) { // print an edge and its weight
    printf("%d-%d: %.2f", e.v, e.w, e.x);
    return;
}

int isEdge(Edge e, Graph g) { // 0 if not found, else 1;
also fill in wgt
   int found = 0;
   if (g != NULL) {
      if (g->edges[e.v][e.w] != NOWEIGHT) {
         found = 1;
      }
   }
   return found;
}

Edge getEdge(Vertex v, Vertex w, Graph g) {
   Edge e = {0, 0, 0.0};
   if (validV(g, v) || validV(g, w)) {
      e.v = v;
      e.w = w;
      e.x = g->edges[v][w];
   }
   return e;
}

int cmpEdge(Edge e1, Edge e2) { // comparison based on
edge weight
   int retval = 0;
   if (e1.x < e2.x) {
```

只需判断v值是否在范围内部即可

多了一个Weight的参数

同上

不为-1，就是edge

获取所有的值

判断两个edge的大小关系

```
      retval = -1;
   }
   else if (e1.x > e2.x) {
      retval = 1;
   }
   return retval;
}

void insertEdge(Edge e, Graph g) { // insert an edge into
a graph
   if (g == NULL) {
      fprintf(stderr, "insertEdge: graph not
initialised\n");
   }
   else {
       if (!validV(g, e.v) || !validV(g, e.w)) {
          fprintf(stderr, "insertEdge: invalid vertices
%d-%d\n", e.v, e.w);
       }
       else {
          if (!isEdge(e, g)) { // increment nE only if it
is new
             g->nE++;                          插入edge，然后赋值为权值
          }
          g->edges[e.v][e.w] = e.x;
          g->edges[e.w][e.v] = e.x;
       }
   }
   return;
}

void removeEdge(Edge e, Graph g) { // remove an edge from
a graph
    if (g == NULL) {
        fprintf(stderr, "removeEdge: graph not
initialised\n");
    }
    else {
        if (!validV(g, e.v) || !validV(g, e.w)) {
            fprintf(stderr, "removeEdge: invalid
vertices\n");
        }
        else {
            if (isEdge(e, g) == NOWEIGHT) {   // is edge
there?
                g->edges[e.v][e.w] = NOWEIGHT;
                g->edges[e.w][e.v] = NOWEIGHT;
                g->nE--;
            }
        }                                      把权值改为NOWEIGHT即可
    }
    return;
}

Weight getWeight(Graph g, Vertex v1, Vertex v2) { // get
```

```
Weight: NOWEIGHT if not existing
    Edge e = {v1, v2}; // not required, but for
consistency
    Weight retval = 0.0;

    if (g == NULL) {
        fprintf(stderr, "getWeight: graph not
initialised\n");
    }
    else {
        if (!validV(g, e.v) || !validV(g, e.w)) {
            fprintf(stderr, "getWeight: invalid
vertices\n");
        }
        else {
            retval = g->edges[e.v][e.w];
        }
    }
    return retval;
}

void showGraph(Graph g) { // print a graph
    if (g == NULL) {
        printf("NULL graph\n");
    }
    else {
        printf("V=%d, E=%d\n", g->nV, g->nE);
        int i;
        for (i = 0; i < g->nV; i++) {
            int nshown = 0;
            int j;
            for (j = 0; j < g->nV; j++) {
                if (g->edges[i][j] != NOWEIGHT) {
                    printf("%d %d:%.2f ", i, j,
g->edges[i][j]);
                    nshown++;
                }
            }
            if (nshown > 0) {
                printf("\n");
            }
        }
    }
    return;
}
```

RETURN

WeightedGraphADT (2019-07-31 17:48:13由AlbertNymeyer编辑)