Lab1: generating random graphs

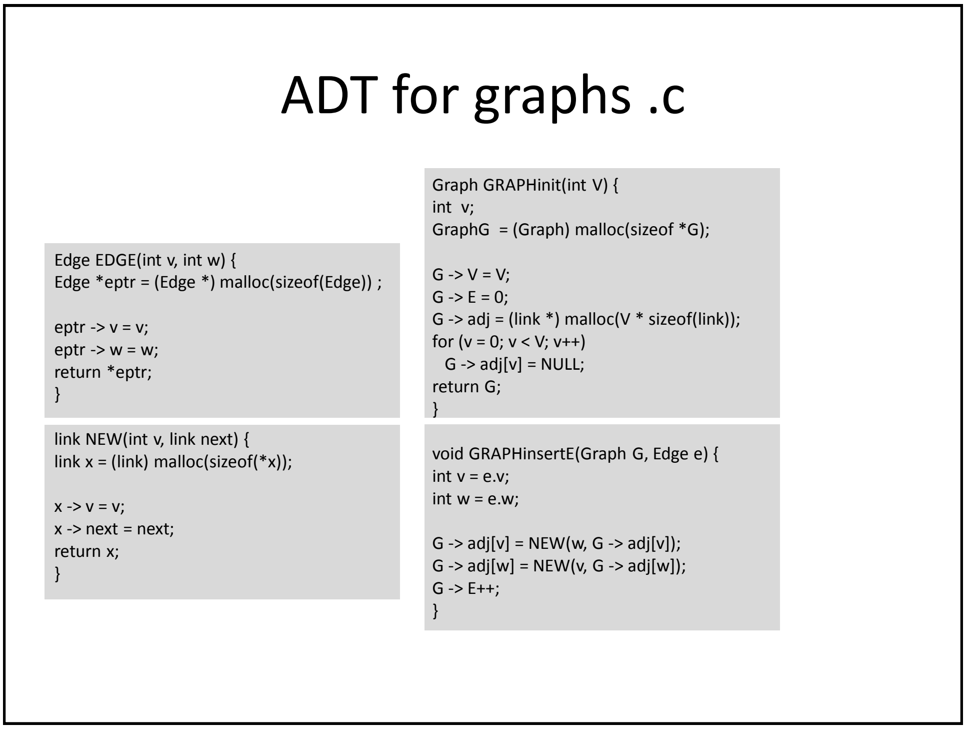
ADT for graphs .h

```
typedef struct {int v; int w;} Edge;
Edge EDGE(int, int);
```

```
typedef struct node *link;
struct node {int v; link next;};
link NEW(int v, link next);
```

```
/* Adjacency list representation of a graph */
typedef struct {int V; int E; link *adj;} *Graph;
```

```
Graph GRAPHinit(int);
void GRAPHinsertE(Graph, Edge);
void GRAPHshow(Graph);
void GRAPHplot(Graph, char *);
int randV(Graph);
Graph GRAPHrandE(int, int);
Graph GRAPHrandp(int, int);
```



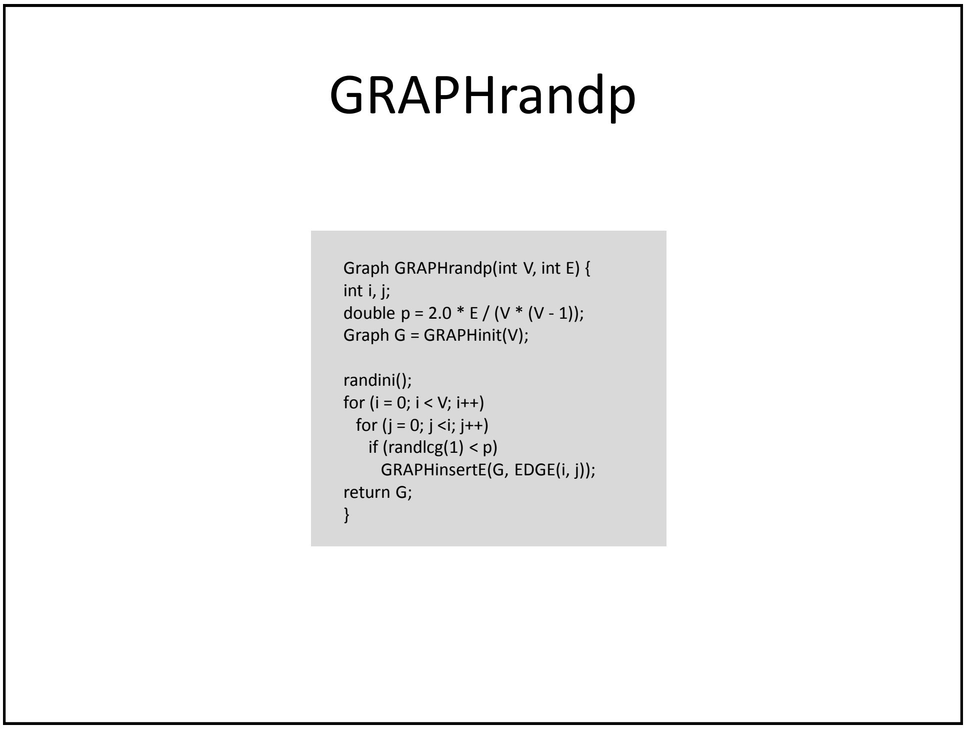
ADT for graphs .c

```
Edge EDGE(int v, int w) {
Edge *eptr = (Edge *) malloc(sizeof(Edge)) ;

eptr -> v = v;
eptr -> w = w;
return *eptr;
}
```

```
link NEW(int v, link next) {
link x = (link) malloc(sizeof(*x));

x -> v = v;
x -> next = next;
return x;
}
```

```
Graph GRAPHinit(int V) {
int  v;
GraphG  = (Graph) malloc(sizeof *G);

G -> V = V;
G -> E = 0;
G -> adj = (link *) malloc(V * sizeof(link));
for (v = 0; v < V; v++)
   G -> adj[v] = NULL;
return G;
}
```

```
void GRAPHinsertE(Graph G, Edge e) {
int v = e.v;
int w = e.w;

G -> adj[v] = NEW(w, G -> adj[v]);
G -> adj[w] = NEW(v, G -> adj[w]);
G -> E++;
}
```



GRAPHrandp

```
Graph GRAPHrandp(int V, int E) {
int i, j;
double p = 2.0 * E / (V * (V - 1));
Graph G = GRAPHinit(V);

randini();
for (i = 0; i < V; i++)
   for (j = 0; j <i; j++)
      if (randlcg(1) < p)
         GRAPHinsertE(G, EDGE(i, j));
return G;
}
```

Output - I

```
void GRAPHshow(Graph G) {
int v;
linkt ;

printf("%d vertices, %d edges\n", G -> V, G -> E);
for (v = 0; v < G -> V; v++) {
  printf("%2d:", v);
  for (t = G -> adj[v]; t != NULL; t = t -> next)
    printf(" %2d", t -> v);
  printf("\n");
  }
}
```

Output - II

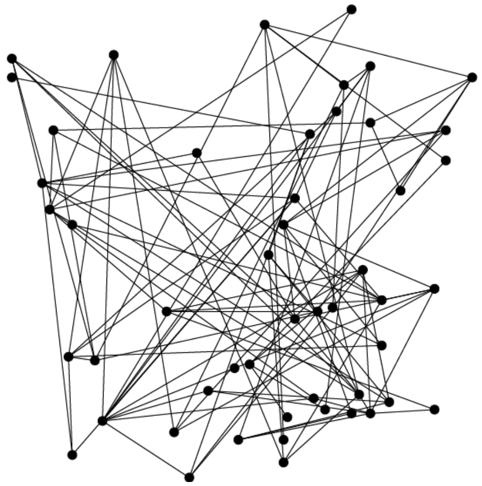
```
void GRAPHplot(Graph G, char *filename) {
FILE*ofp;
int v;
link t;

ofp = fopen(filename, "w");
fprintf(ofp, "%s", "size 12 12\n");
randini();
for (v = 0; v < G -> V; v++) {
  fprintf(ofp, "%s%6.1f%6.1f\n", "amove ", rand_unif(2,13), rand_unif(3,13));
  fprintf(ofp, "%s\n", "circle 0.12 fill black");
  fprintf(ofp, "%s%d\n", "save v", v);
}
for (v = 0; v < G -> V; v++)
  for (t = G -> adj[v]; t != NULL; t = t -> next)
    fprintf(ofp, "%s%d%s%d%s\n", "join v", v, ".cc - v", t -> v, ".cc");
fclose(ofp);
}
```

GLE file

GLE file

```
size 12 12
amove    7.7   7.1
circle 0.12 fill black
save v0
amove    5.7   2.7
circle 0.12 fill black
save v1

          •
          •
          •

join v0.cc - v49.cc
join v0.cc - v46.cc
join v0.cc - v24.cc
join v0.cc - v21.cc
join v0.cc - v17.cc
```

Graph drawing



More information

- “Algorithms in C: Part 5 Graph Algorithms,” Robert Sedgewick, 2002
- http://glx.sourceforge.net/ (GLE)