```
8
1 2
1 7
2 5
5 7
5 3
3 1
7 6
6 4
4 1
5 1
5 6
6 1
8 5
```

```
Dimensione: 8
1 --> 7 --> 2
2 --> 5
3 --> 1
4 --> 1
5 --> 6 --> 1 --> 3 --> 7
6 --> 1 --> 4
7 --> 6
8 --> 5
```

```
struct adj_node {
  int node; // ID
  float weight;

  struct adj_node* next;
};

typedef adj_node* adj_list;
```

```
graph new_graph(int n){
    graph G;
    G.dim = n;
    G.nodes = new struct adj_node * [n]; // [n] pointers of type struct adj_node.
    for (int i=0; i<n; i++) {
        G.nodes[i] = NULL;  //indicazione di lista vuota.   // (*(G.nodes)+i)->next
    }
    return G;
}
```

```
typedef struct {
  adj_node** nodes; //adj_list* nodes;
  int dim;
} graph;
```

```
G.nodes = new struct adj_node * [n];
```

```
struct adj_node {
  int node; // ID
  float weight;

  struct adj_node* next;
};

typedef adj_node* adj_list;
```

```
typedef struct {
  adj_node** nodes; //adj_list* nodes;
  int dim;
} graph;
```

G.nodes[0] --> 7 --> 2
G.nodes[1] --> 5
G.nodes[2] --> 1
G.nodes[3] --> 1
G.nodes[4] --> 6 --> 1 --> 3 --> 7
G.nodes[5] --> 1 --> 4
G.nodes[6] --> 6
G.nodes[7] --> 5

Attenzione G.nodes[0] non ritorna struct a cui puoi accedere e cambiare campi!
Ma G.nodes[0] ti da indirizio di node id 7!!
Di per se G.nodes[] e' solo puntatore!!!!!!!

~~G.nodes[0] Dimensione: 8~~
~~G.nodes[1] 1 --> 7 --> 2~~
~~G.nodes[2] 2 --> 5~~
~~G.nodes[3] 3 --> 1~~
~~G.nodes[4] 4 --> 1~~
~~G.nodes[5] 5 --> 6 --> 1 --> 3 --> 7~~
~~G.nodes[6] 6 --> 1 --> 4~~
~~G.nodes[7] 7 --> 6~~
~~8 --> 5~~

NON COSI!

```
struct adj_node {
    int node;//ID
    float weight;
    struct adj_node *next;

    struct node* data;
};
typedef adj_node *adj_list;

typedef struct {
    adj_list *nodes;
    int dim;
} graph;
```

```
struct node{
    char cont[81];
    char tipo;
};
```

puntatori — struct

G.nodes[0] --> 7 --> 2
G.nodes[1] --> 5
G.nodes[2] --> 1
G.nodes[3] --> 1
G.nodes[4] --> 6 --> 1 --> 3 --> 7
G.nodes[5] --> 1 --> 4
G.nodes[6] --> 6
G.nodes[7] --> 5