链式前向星

这段代码是 **链式前向星（邻接表）** 的核心部分之一，用于**添加一条有向边**。它常用于图论题中构建稀疏图，尤其在竞赛中因其效率和空间优势被广泛使用。

我们先贴出这段代码，然后逐行讲解含义：

```
void add(int a, int b, int c){
    w[idx] = c;          // 1. 当前边的权值为 c
    e[idx] = b;          // 2. 当前边指向节点 b
    ne[idx] = h[a];      // 3. 当前边的“下一条边”指向 h[a]（即之前从 a 出发的第一条边）
    h[a] = idx++;        // 4. 将 h[a] 更新为当前边的编号 idx（即最新一条边），然后 idx++
}
```

🔧 使用前的变量说明（通常定义如下）：

```
const int N = 1e5 + 10;      // 点数最大值
const int M = 2e5 + 10;      // 边数最大值（无向图一般是 2 倍）
int h[N], e[M], ne[M], w[M], idx;
```

名称	类型	作用
`h[a]`	`int` 数组	点 `a` 的**第一条边的编号**（若为 `-1` 则说明没有边）
`e[i]`	`int` 数组	第 `i` 条边的**终点**
`w[i]`	`int` 数组	第 `i` 条边的**权值**
`ne[i]`	`int` 数组	第 `i` 条边的**下一条边编号**（同起点的其他边）
`idx`	`int`	表示当前可用的边的编号（相当于数组指针）

初始化时通常：

```
memset(h, -1, sizeof h);
idx = 0;
```

🧠 add 函数做了什么？

我们用图形解释一下：

添加一条边 `a -> b`，权值为 `c`，操作如下：

1. 把第 `idx` 条边的信息设置好：
   - `w[idx] = c`：权值
   - `e[idx] = b`：终点为 `b`
2. `ne[idx] = h[a]`：这一步是关键，**把这条新边链接到之前已有的以 `a` 为起点的边表上**
3. `h[a] = idx`：将 `a` 的第一条边更新为这条新的边
4. `idx++`：边的编号加 1，为下一次添加做准备

这等价于将边插入到了链表头部（即**头插法**）。

📌 举个例子：

假设我们按以下顺序添加三条边：

```
add(1, 3, 10); // 边0: 1 -> 3
add(1, 2, 5);  // 边1: 1 -> 2
add(1, 4, 8);  // 边2: 1 -> 4
```

此时我们查看点 1 的边表：

- `h[1] = 2`（最后一条插入的边）
- `e[2] = 4`, `ne[2] = 1`
- `e[1] = 2`, `ne[1] = 0`
- `e[0] = 3`, `ne[0] = -1`

遍历从点 `1` 出发的所有边：

```
for(int i = h[1]; i != -1; i = ne[i]){
    cout << "to " << e[i] << ", weight = " << w[i] << endl;
}
```

输出顺序是：

```
to 4, weight = 8
to 2, weight = 5
to 3, weight = 10
```

✅ 总结：链式前向星的优点

- **存储紧凑**：只用一维数组模拟邻接表
- **遍历效率高**：可以 `O(出边数)` 遍历一个点的所有出边
- **插入快**：头插法，`O(1)` 插入
- **控制力强**：更接近底层指针操作，非常适合手动优化图论算法（比如 SPFA、Dijkstra、Tarjan）