

链式前向星

在理解了邻接表的写法之后，我们可以对其少加改动，就能够实现一种全新的存储：链式前向星。

根据现有的通用命名规则：

我们将原有的first数组更名为head数组，其键为始发顶点编号，值为所链接的边的编号。

我们将原本的next数组，改为Edge结构体数组，并将原来的目标边和权值存入其中。

现有结构如下：

//边结构体

```
struct Edge {
    int next;    //与当前顶点共顶点的下一条边的编号
    int to;      //当前顶点将要到达的目标顶点号
    int w;       // 若果有的话
}
```

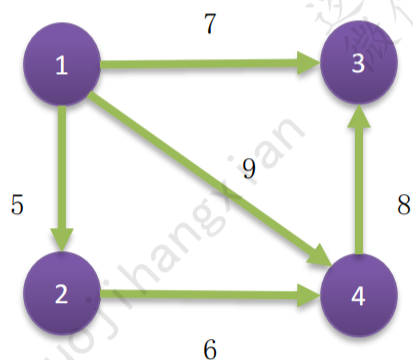
head

1	0
2	0
3	0
4	0
5	0

edges

	next	w	to
1			
2			
3			
4			

我们还是以邻接表中的图像为例，原始图像如下：



4 5
1 4 9
4 3 8
1 2 5
2 4 6
1 3 7

如同邻接表一样，我们开始填入数据。首先是第一条：1 4 9

edges

	next	w	to
1	0	9	4
2			
3			
4			
5			

head

1	0	1
2		
3		
4		

继续填入第二条数据： 4 3 8

edges

	next	w	to
1	0	9	4
2	0	8	3
3			
4			
5			

head

1	1	1
2		
3		
4	0	2

继续填入第三条数据： 1 2 5

edges

	next	w	to
1	0	9	4
2	0	8	3
3	1	5	2
4			
5			

head

1	1	3
2		
3		
4	2	2

此时，edges的第3条边的next变成了head[1]的值“1”，说明1号边和3号边共顶点1，因此它们两个将被关联起来。

继续填入第四条数据： 2 4 6

edges

	next	w	to
1	0	9	4
2	0	8	3
3	1	5	2
4	0	6	4
5			

head

1	3	3
2	0	4
3		
4	2	2

最后填入第五条数据： 1 3 7

edges

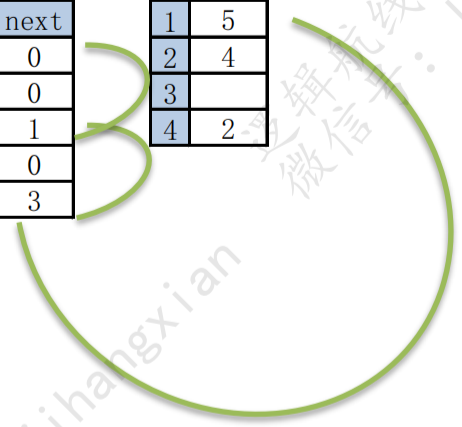
	next	w	to
1	0	9	4
2	0	8	3
3	1	5	2
4	0	6	4
5	3	7	3

head

1	3	5
2	0	4
3		
4	2	2

最终结果如下：

edges				head	
	w	to	next		
1	9	4	0	1	5
2	8	3	0	2	4
3	5	2	1	3	
4	6	4	0	4	2
5	7	3	3		



1号顶点连接第5条边，第5条边连接第3条边，第3条边连接第1条边。

编码

```
#include <bits/stdc++.h>

using namespace std;

#define N 10005

//定义边的结构体
struct Edge {
    int next; //下一条边的编号
    int to;   //当前顶点链接的目标顶点
    int w;    //权重
} edges[N]; //边数组

int head[N]; //始发定点数组
int cnt = 0; //边索引

//添加边的函数
//起始顶点，目标顶点，权重
void AddEdge(int u, int v, int w) {
    //边索引增加
    cnt++;
    //记录基本信息
    edges[cnt].to = v;
    edges[cnt].w = w;
    //记录共顶点链接信息
    edges[cnt].next = head[u];
    //更新顶点链接的边索引
    head[u] = cnt;
}
```

```

//测试代码n个顶点，m条边
int n, m;
int u, v, w;

int main() {
    memset(head, 0, sizeof(int));
    scanf("%d%d", &n, &m);
    read(n), read(m);
    for (int i = 1; i <= m; i++) {
        scanf("%d%d%d", &u, &v, &w);
        AddEdge(u, v, w);
    }
    for (int i = 1; i <= n; i++) {
        if (head[i]) {
            for (int j = head[i]; j != 0; j = edges[j].next) {
                printf("%d %d %d\n", i, edges[j].to, edges[j].c);
            }
        } else {
            printf("\n");
        }
    }
    return 0;
}

```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。

