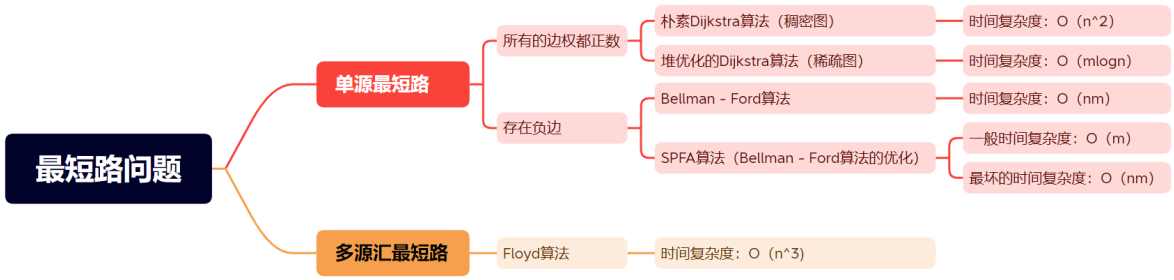
算法基础（十三）： 图 - 最短路问题3 - Floyd算法

复习一下这个图：



基本思想：

`floyd`算法过程超级简单，他是用来求多源汇最短路的，也就是说他是求任意的两个点之间的最短路，过程如下：

首先我们规定一个二维数组`d[i][j]`来存储所有的边

对于重边，我们只存最小权值的边

对于自环，题目中出现自环只可能是正自环，我们不保存正环

对于负环，由于`floyd`不能计算带负环的图，题目中使用`floyd`的时候不会出现负环的数据

然后经过三重循环后得到的`d[i][j]`就表示源点`i`到`j`的最短距离

伪码：

```
for(int k = 1; k <= n; k ++)
    for(int i = 1; i <= n; i ++)
        for(int j = 1; j <= n; j ++)
            d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
```

他的思想来自于动态规划，想知道为什么可以自行查看证明

代码实现：

```
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;

const int N = 210, INF = 1e9;
```

```cpp

int n, m, Q;

int d[N][N];

int floyd()
{
    //三重循环
    for(int  k = 1; k <= n; k ++)
        for(int i = 1; i <= n; i ++)
            for(int j = 1; j <= n; j ++)
                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
}


int main(){
    scanf("%d%d%d", &n, &m,&Q);
    //初始化
    for(int i = 1; i <= n ; i ++){
        for(int j = 1; j <= n; j ++){
            //初始化自己到自己的距离是0，从而处理所有的自环
            if(i == j) d[i][j] = 0;
            //自己到别的点的距离是无穷
            else d[i][j] = INF;
        }
    }

    while(m --){
        //读入所有的边
        int a, b, w;
        scanf("%d%d%d", &a, &b, &w);

        //处理重边，只保留最小的边
        //同样这里来处理自环的问题，这里再前面自己到自己的距离统一是0
        //而且题目保证不出现负环，所以自环统一不读入了
        d[a][b] = min(d[a][b], w);
    }

    floyd();

    while(Q --)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        //跟之前的算法判断相似，当两个点不可达的时候可能距离小于正无穷
        if(d[a][b] > INF / 2) puts("impossible");
        else printf("%d\n", d[a][b]);
    }
    return 0;
}

```

实现复杂度 `O(n^3)`