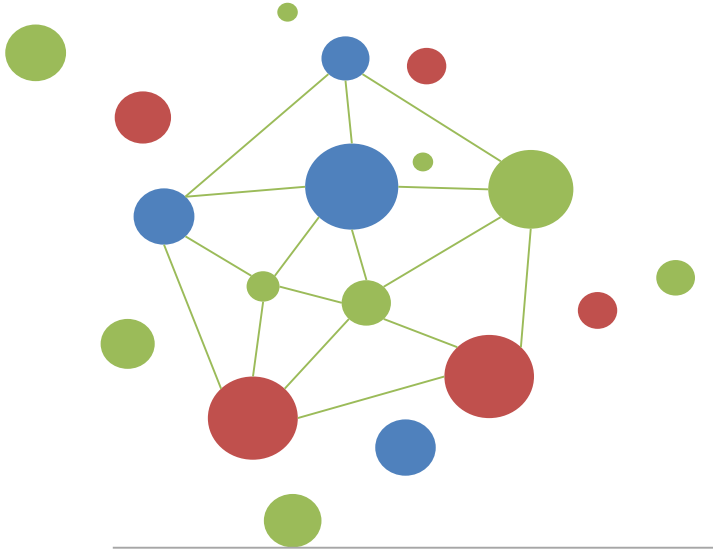
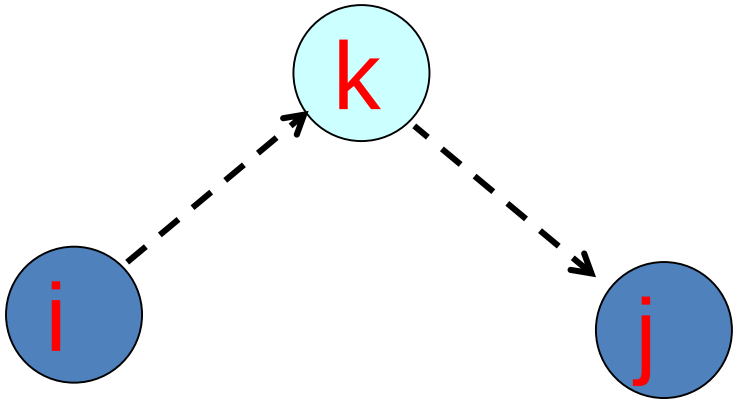


floyed算法

2019.04.13

floyd算法（计算每一对顶点间的最短路径）

最简单的最短路径算法，可以计算图中任意两点间i到j的最短路径**d[i][j]**。Floyed的时间复杂度是O (N^3)，适用于出现负边权的情况（无负劝回路）。




```
if  (d[i][k]+d[k][j]<d[i][j] ) d[i][j]=d[i][k]+d[k][j]
```

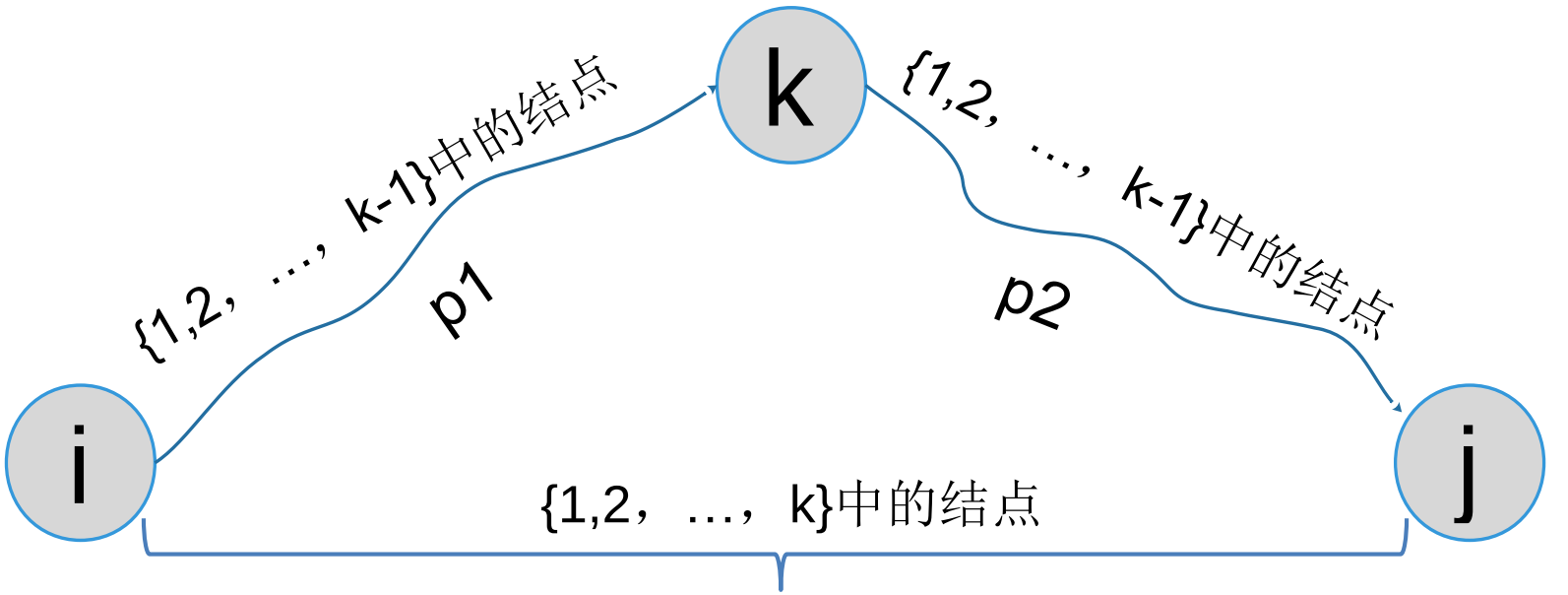
或写成：`d[i][j]=min(d[i][j],d[i][k]+d[k][j])`

邻接矩阵初始化：对角线为0；无边的无穷大
使用之前初始化：
memset(d,0x3f,sizeof(d));
d[i][i]=0
d[i][j]=a[i][j]　i到j有边//可以直接用d存邻接矩阵

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

一遍floyed类似矩阵乘法：A*A

d[k][i][j]:表示“经过若干个编号不超过k的节点”从i到j的最短距离。
考虑i到j的路径p的中间结点均取自于{1,2，…，k}：
如果k不是路径p的中间结点，则p的所有中间结点都在集合{1,2，…，k-1}
如果k是路径p的中间结点，则p分解为：i→k→j
d[k][i][j]=a[i][j]　　(k=0)
d[k][i][j]=min(d[k-1][i][j],d[k-1][i][k]+d[k-1][k][j])　(k>=1)



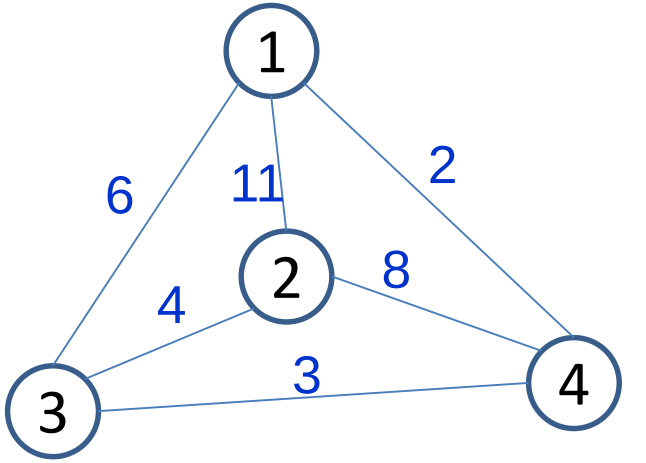
去掉一维k直接更新：d[i][j]=min(d[i][j],d[i][k]+d[k][j])

变形及其应用：

- 1.求s到e正好经过k条边的最短路(不是简单路径)
- 2.求最小环
- 3.传递闭包

1.求s到e正好经过k条边的最短路(不是简单路径)

memset(a,0x3f,sizeof(a));
a[i][i]=0
a[i][j]=w i到j有边w

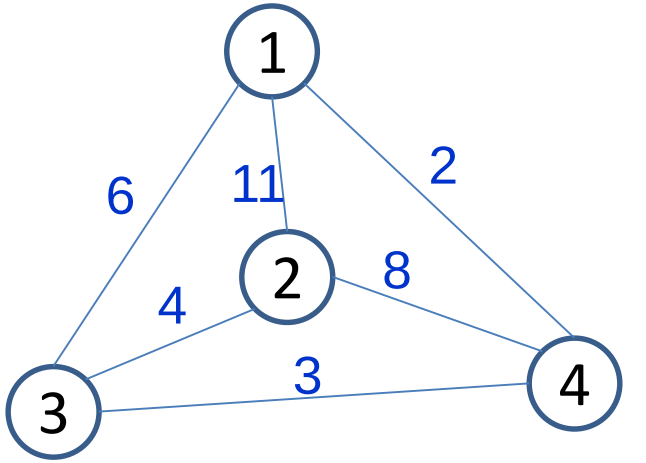


没跑floyed前临接矩阵A表示任意两点经过1条边的最短路。A

```
1.  for(int k=1;k<=n;k++)
2.     for(int i=1;i<=n;i++)
3.        for(int j=1;j<=n;j++)
4.           a[i][j]=min(a[i][j],a[i][k]+a[k][j]);
```

一遍floyed后：a[i][j]是i到j的最短距离：
中间至少1条边（连通图），最多n-1条边。

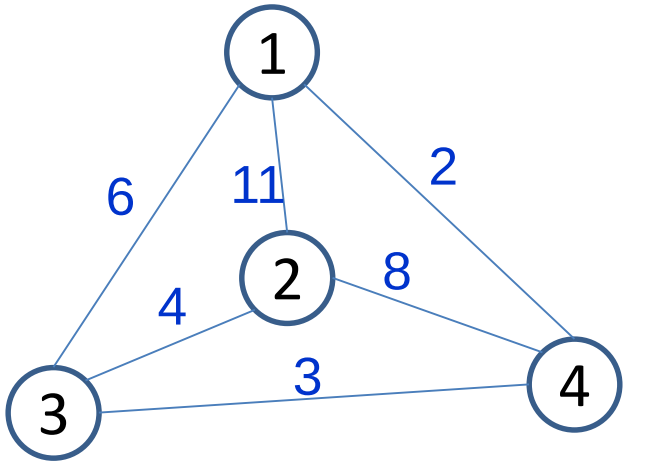
memset(c,0x3f,sizeof(c));




```
1.  for(int k=1;k<=n;k++)
2.     for(int i=1;i<=n;i++)
3.        for(int j=1;j<=n;j++)
4.           c[2][i][j]=min(c[2][i][j],a[i][k]+a[k][j]);
```

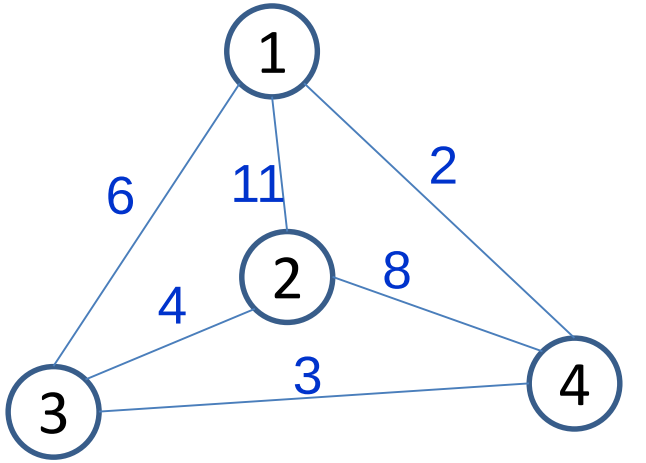
没跑floyed前临接矩阵A表示任意两点经过1条边的最短路。C[1]=A
1遍floyed后：c[2][i][j]是i到j的最短距离：

中间恰好经过2条边。C[2]=A*A



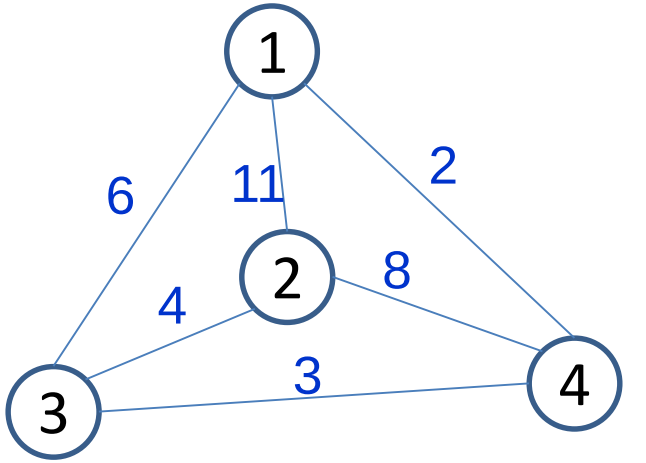

```
1.  for(int k=1;k<=n;k++)
2.     for(int i=1;i<=n;i++)
3.        for(int j=1;j<=n;j++)
4.           c[3][i][j]=min(c[3][i][j],c[2][i][k]+a[k][j]);
```

2遍floyed后：c[3][i][j]是i到j的最短距离：
中间恰好经过3条边。A*A*A　C[3]=C[2]*A




```
1.  for(int k=1;k<=n;k++)
2.     for(int i=1;i<=n;i++)
3.        for(int j=1;j<=n;j++)
4.           c[4][i][j]=min(c[4][i][j],c[3][i][k]+a[k][j]);
```

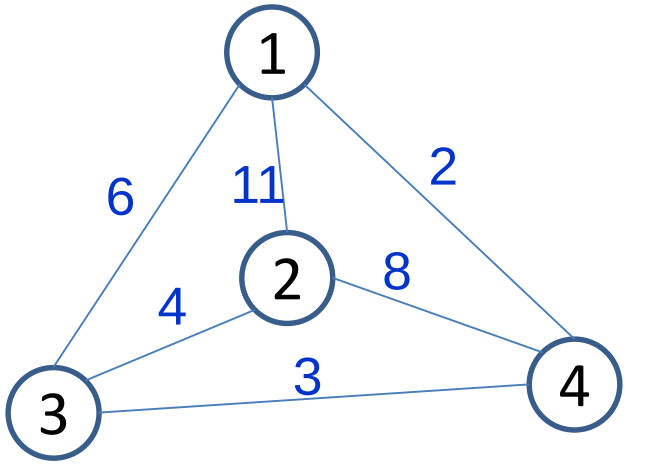
3遍floyed后：c[4][i][j]是i到j的最短距离：
中间恰好经过4条边。A*A*A*A
C[4]=C[3]*A




```
1.  for(int k=1;k<=n;k++)
2.     for(int i=1;i<=n;i++)
3.        for(int j=1;j<=n;j++)
4.           c[k][i][j]=min(c[k][i][j],c[k-1][i][k]+a[k][j]);
```

K-1遍floyed后：c[k][i][j]是i到j的最短距离：
中间恰好经过k条边。
$A*A*A*A*..*A=A^K$
C[k]=C[k-1]*A

```
1.  for(int k=1;k<=n;k++)
2.     for(int i=1;i<=n;i++)
3.        for(int j=1;j<=n;j++)
4.           c[r+p][i][j]=min(c[r+p][i][j],c[r][i][k]+c[p][i][k]);
```



r+p-1遍floyed后：c[r+p][i][j]是i到j的最短距离：
中间恰好经过r+p条边。
$C[r+p]=A^r*A^p=A^{r+p}$

◆所以：

- 0遍floyed： C[1]=A
- 1遍floyed： C[2]=A^2
- 2遍floyed： C[3]=A^3
- ..
- K-1遍floyed： C[k]=A^k
- 最终C[k]是恰好经过k条边的最短路（就是k-1遍floyed而已）。
- 如果K很大， A^k利用快速幂求。

重载*运算符：一次floyed

```
struct Matrix{
    int a[N][N];
};

Matrix operator *(Matrix A,Matrix B){
    Matrix C;
    memset(C.a,0x3f,sizeof(C.a));
    for(int k=1;k<=n;k++)
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
                C.a[i][j]=min(C.a[i][j],A.a[i][k]+B.a[k][j]);
    return C;
}
```

朴素算法：时间复杂度=$O(k*n^3)$

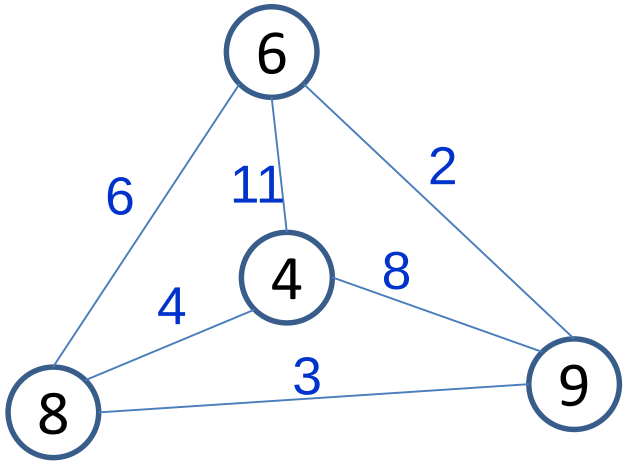
```
Matrix S=A;
for(int i=1;i<k;i++) S=S*A;
```

S.a[s][e]:s到e经过k条边的最短路。

矩阵快速幂：时间复杂度=O(log(k)*n^3)

```
Matrix Power(Matrix A,int k){
    Matrix S=A;
    k--;
    while(k>0){
        if(k&1)S=S*A;
        A=A*A;
        k=k>>1;
    }
    return S;
}
```

题目：Cow Relays p2886 poj3613



给出一张无向连通图，求S到E经过n条边的最短路。

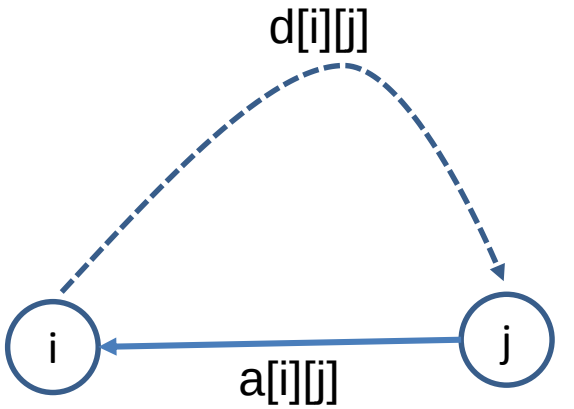
边T：2 ≤ T ≤ 100：

顶点编号：1 ≤ I1i ≤ 1,000; 1 ≤ I2i ≤ 1,000:需要求顶点个数

图的顶点tot最多2T个，最多200个顶点，可以用map离散化

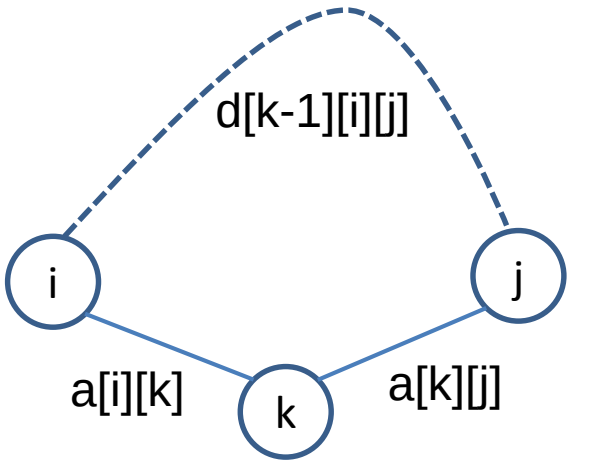
2.求最小环

- 有向图的最小环：
- 简单，跑完一遍floyed，然后枚举每条边即可：
- ans=min{d[i][j]+a[j][i]}



无向图（无重边）的最小环：至少有3个点

- 在floyed的第k层开始前， d[i][j]是保存的是“经过编号不超过k-1”的i到j最短路径长度。即还没有经过k点。
- 所以：
- min{d[i][j]+a[j][k]+a[k][i]}
- 1<=i<j<k
- 保存的是：
- d[k-1][i][j]编号不超过节点k
- 但是一定含有k的路径长度。



```
for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)d[i][j]=a[i][j];
int ans=INF;
for(int k=1;k<=n;k++){
    for(int i=1;i<k;i++)
        for(int j=i+1;j<k;j++)
            if((long long)d[i][j]+a[j][k]+a[k][i]<ans){
                ans=d[i][j]+a[j][k]+a[k][i];
            }
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            if(d[i][k]+d[k][j]<d[i][j]){
                d[i][j]=d[i][k]+d[k][j];
            }
}
```

题目：Sightseeing Trip poj1734

3.传递闭包

- 若干个元素以及二元关系，并且关系具有**传递性**。
- 通过传递性能推导出尽量多的元素直接的关系，这类问题称为传递闭包。
- 如：整数集合上的>:a>b,b>c,则能得出a>c,>具有传递性。
- d[i][j]=1：i和j有关系
- d[i][j]=0:i和j没关系。
- d[i][i]:根据情况而定，如=，>就不同

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

题目：Sorting It All Out p1347 poj1094

训练题目：

- **1. P1037 产生数**
- **2.光纤网络poj2570**
- 3. Sorting It All Out p1347 poj1094