最短路&最小生成树

CUC - ACM Training 6

19 数媒技 杨雪婷

2022.03.12


PART I
最短路

最短路算法总览

INTRODUCTION

BELLMAN-FORD的队列优化
可以判断负环

SPFA

单源最短路
所有点对最短路

DIJKSTRA

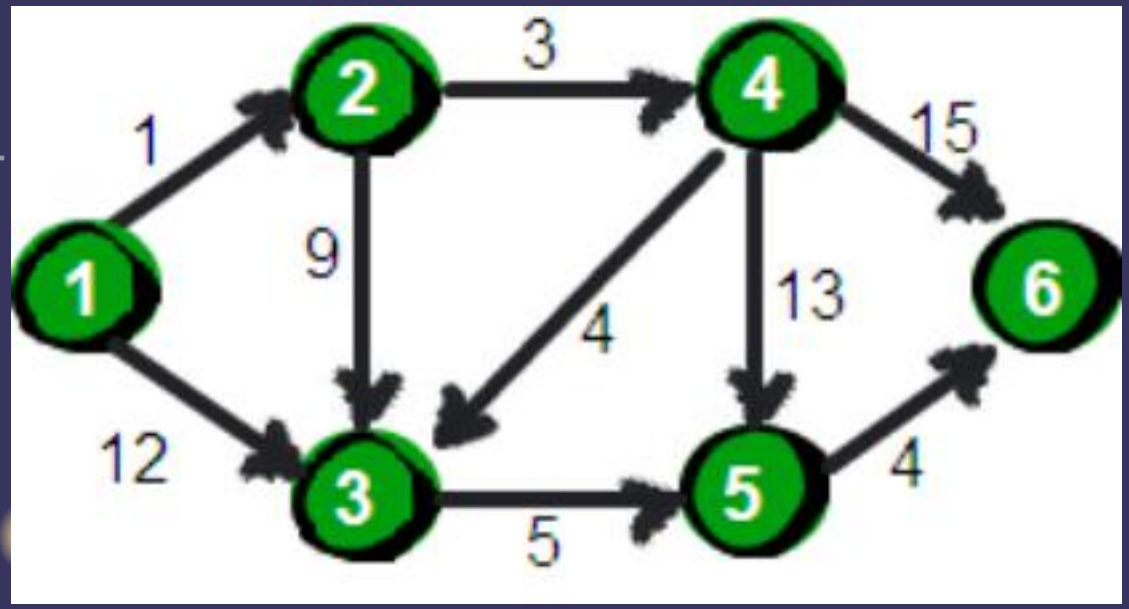
任意两点直接的最短路
本质上是动态规划

FLOYD

Dijkstra

问题:

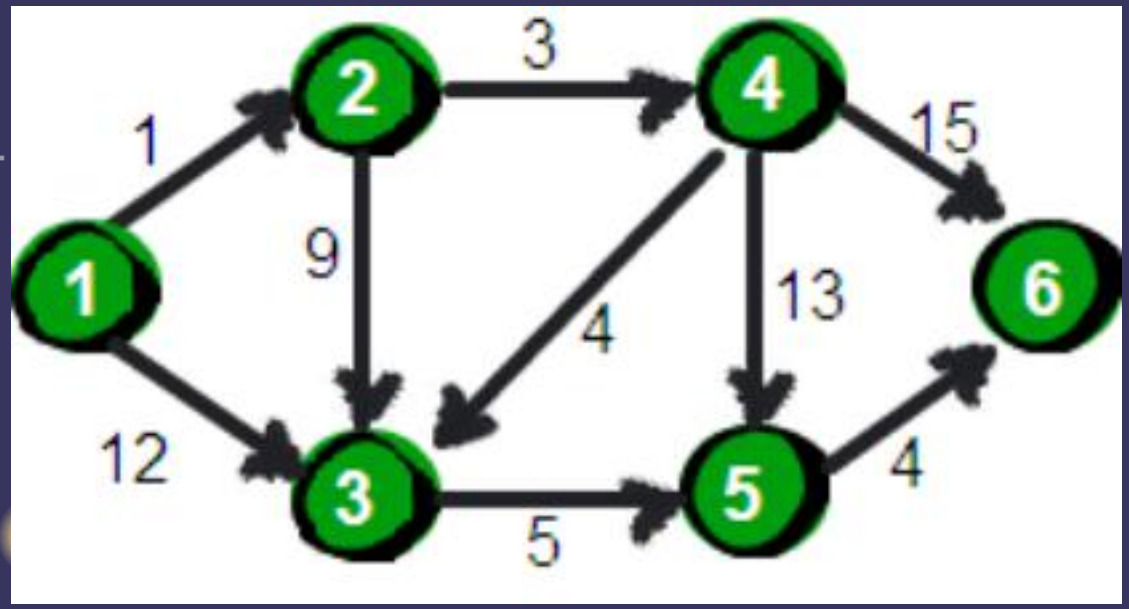
给定一个图，求从起点到终点，经过权值最小的路径



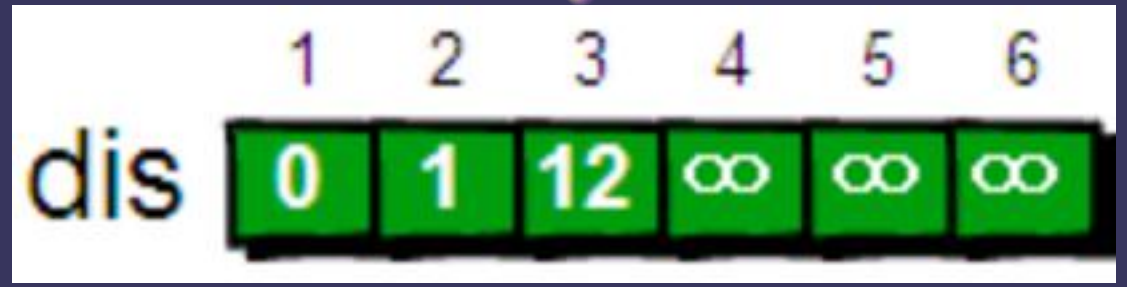
Dijkstra

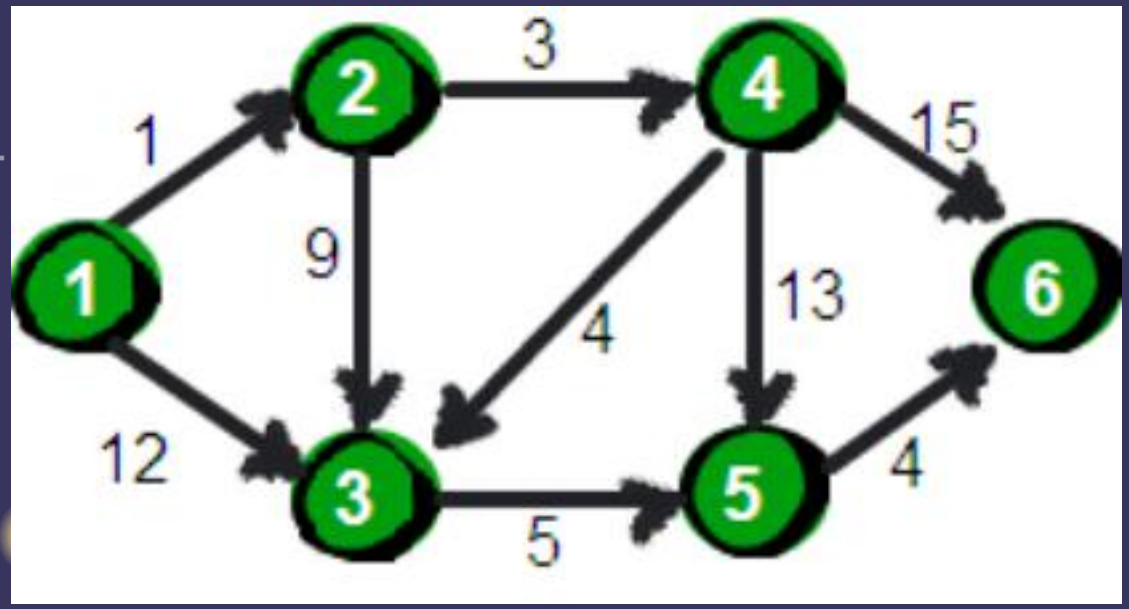
基本思路:

从起点开始，每次找到一个没有访问过的、距离原点最近的点，然后用这个点，去更新所有和它相连的点到原点的距离。

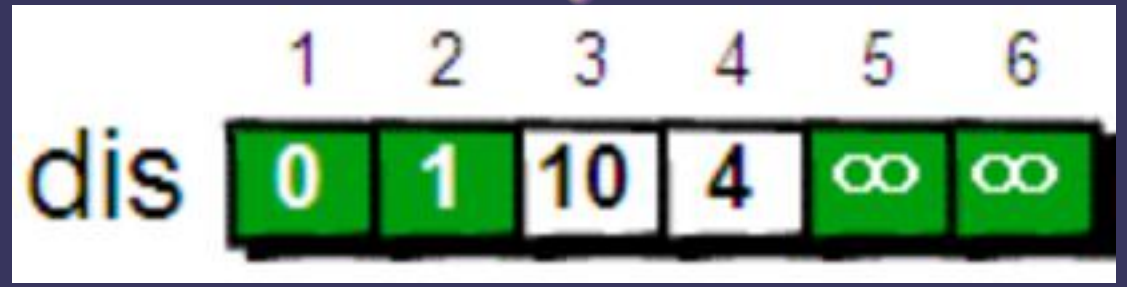


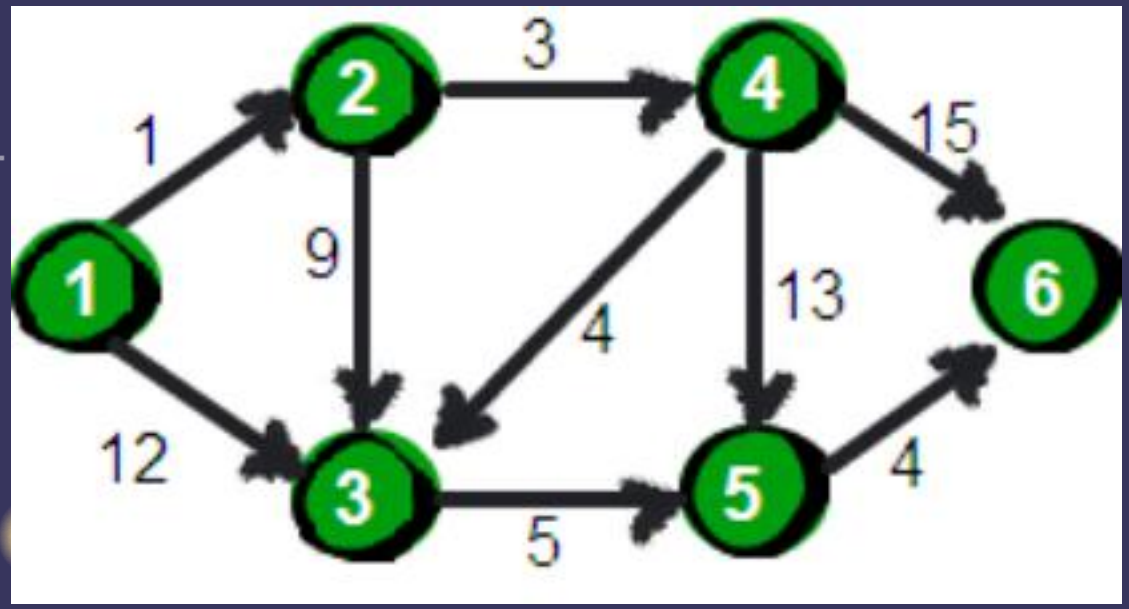
Dijkstra



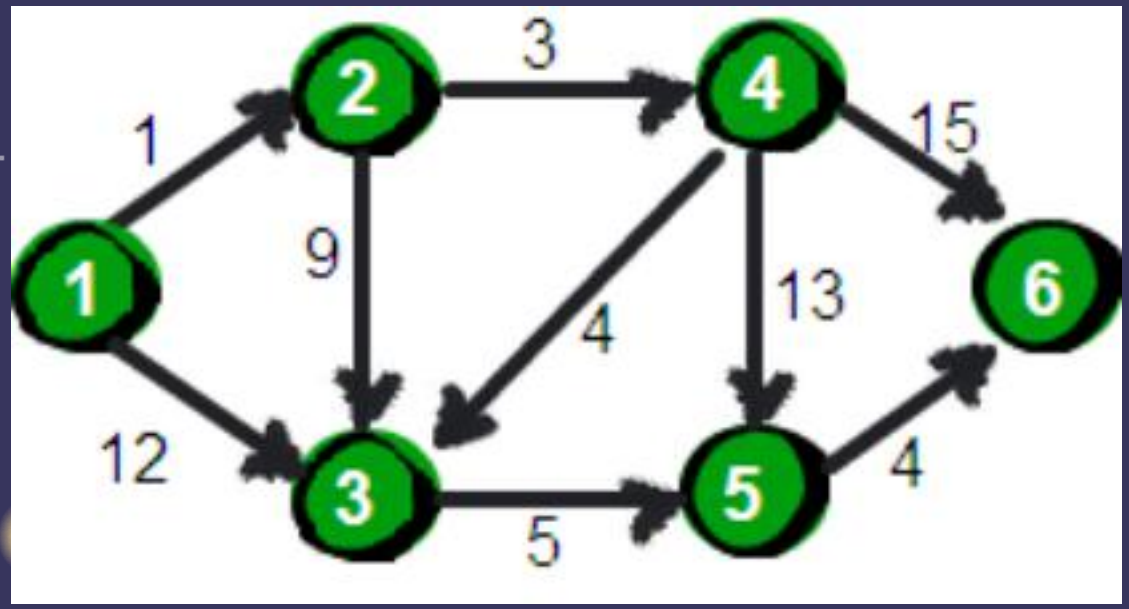


Dijkstra

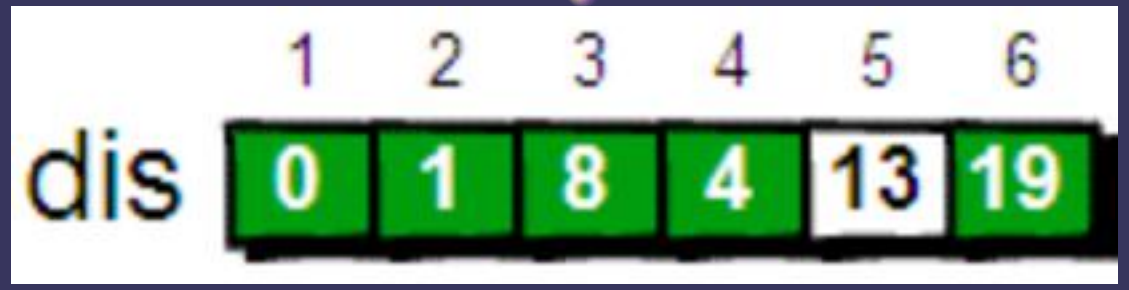


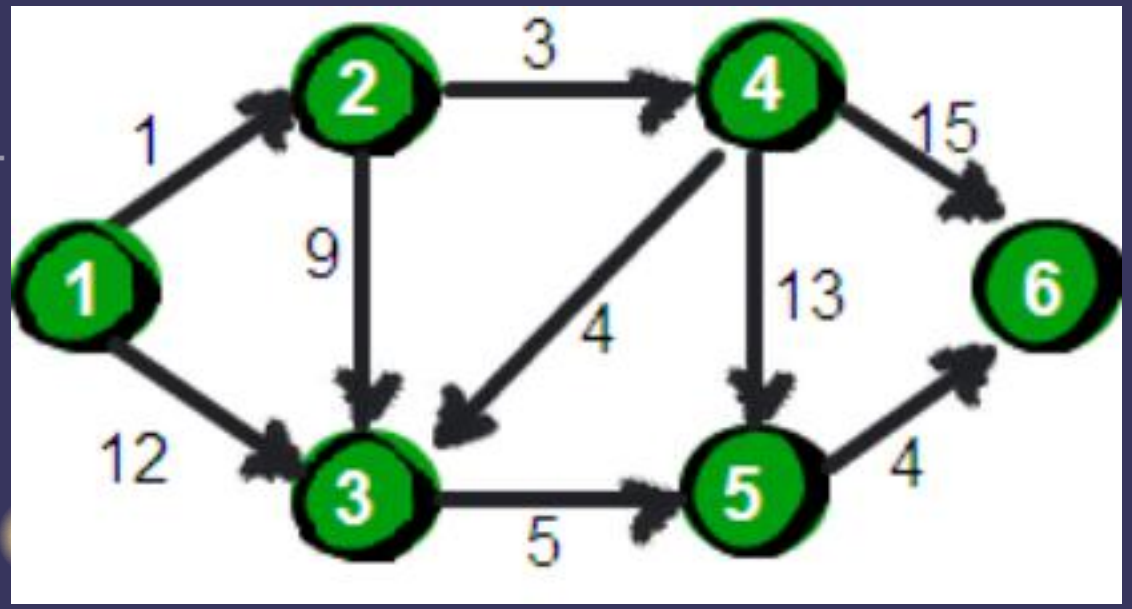


Dijkstra

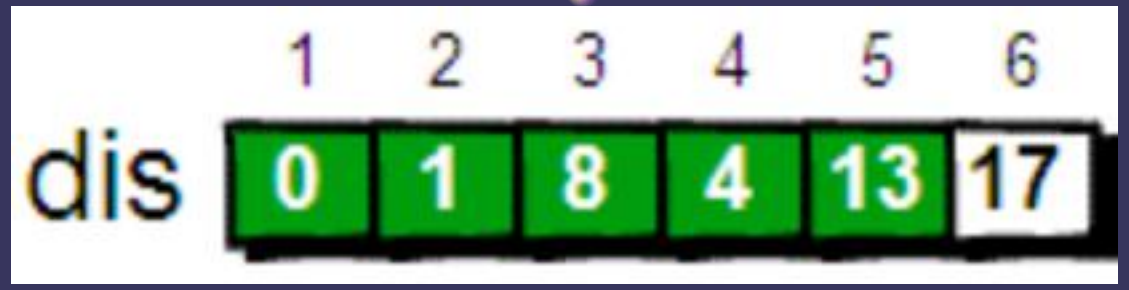


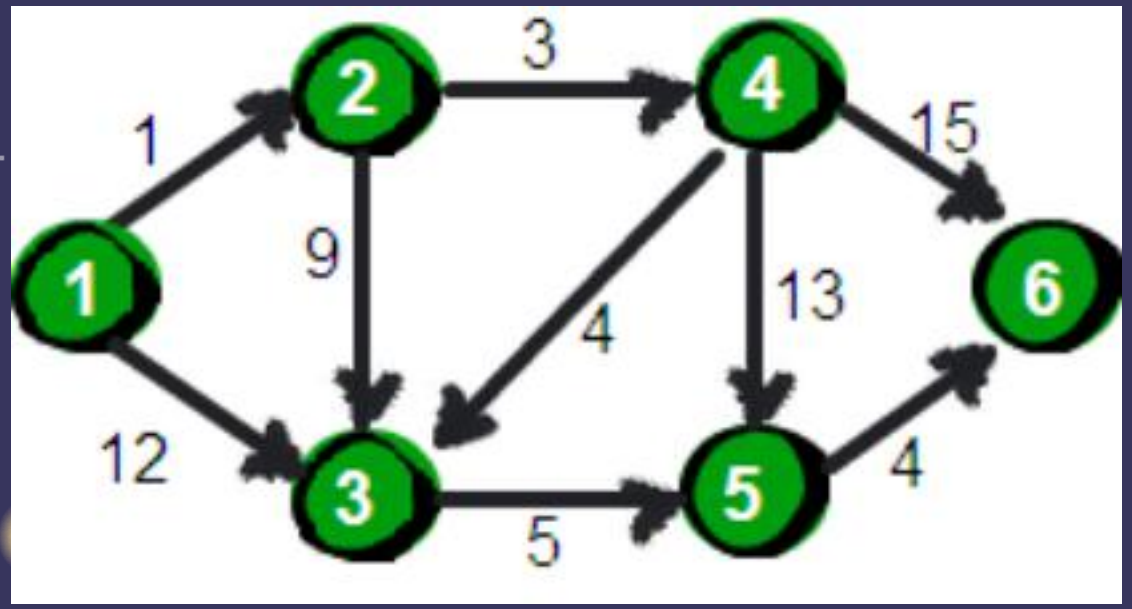
Dijkstra



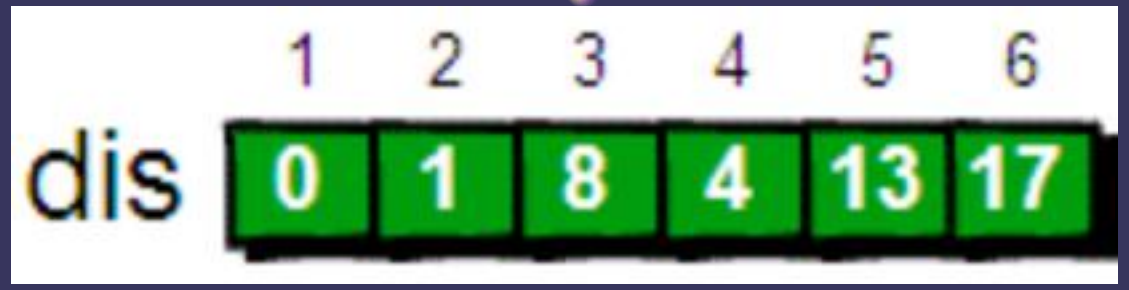


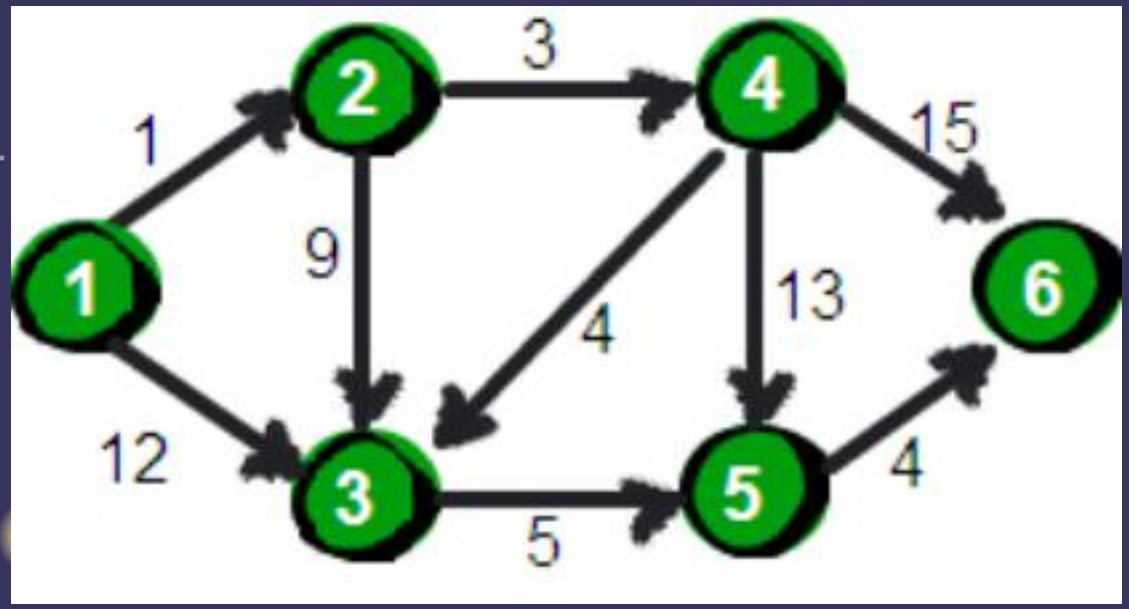
Dijkstra



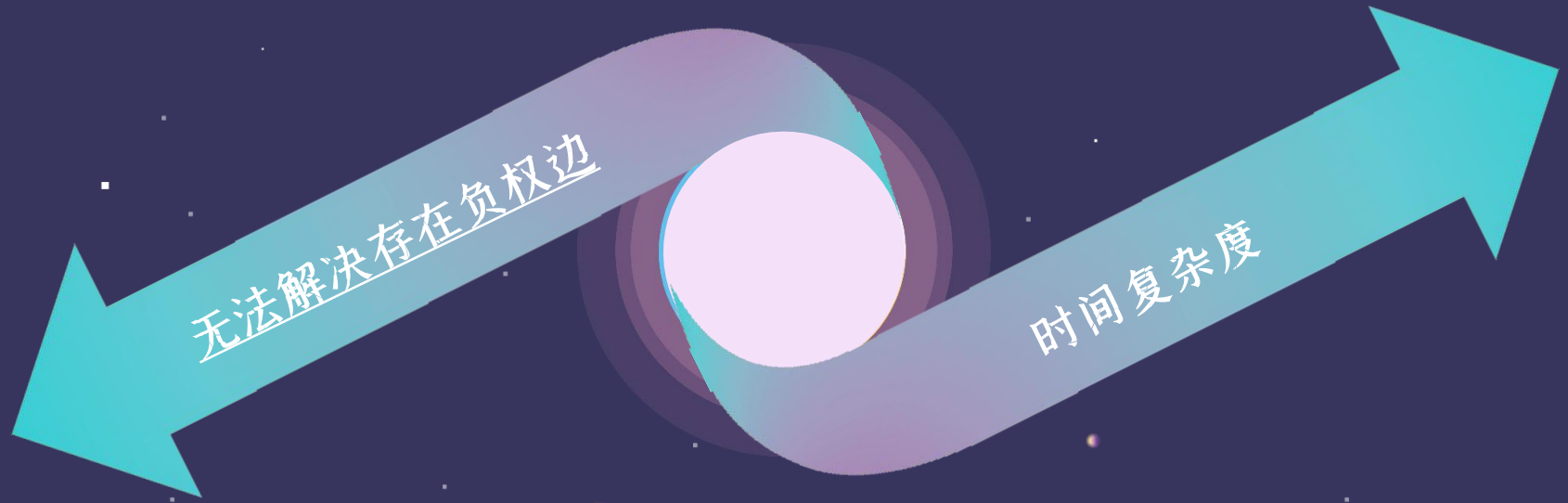


Dijkstra





Dijkstra的局限性



SPFA（Bellman-ford的优化算法）

基本思路:

队列不为空时，每次从队首取出一个点，更新其他能到达的点，然后将没在队列中的点放入队列。

如何判断负环:

某个点更新超过n-1次，则负环存在。

SPFA（Bellman-ford的优化算法）

基本步骤:

1. 先加入起点
2. 枚举当前队首点的所有出边
3. 判断出边对应的节点会不会被更新

ps: 在队列存的就是被更新的点

为什么要枚举:

因为被更短的路更新了，所以所有出边都需要枚举。

SPFA（Bellman-ford的优化算法）

```
void spfa(int u){
    q.push(u);
    vis[u]=1;
    while(!q.empty()){
        int x=q.front();
        q.pop();
        vis[x]=0;
        for(int i=head[x];i;i=nxt[i]){
            int y=to[i];
            if(dis[x]+w[i]<dis[y]){
                dis[y]=dis[x]+w[i];
                if(!vis[y]) vis[y]=1,q.push(y);
            }
        }
    }
}
```

Floyd算法

适用于求所有两两点的最短路径

算法复杂度：O(V^3)

适用范围：

无负权回路即可，边权可正可负，运行一次算法即可求得任意两点间最短路

Floyd算法

```
for(int k=0;k<n;++k)
    for(int i=0;i<n;++i)
        for(int j=0;j<n;++j)
            dist[i][j]=min(dist[i][j],dist[i][k]+dist[k][j])
```

动态规划的思想，找i和j之间通过编号不超过k（k从1到n）的节点的最短路。（当k=n时达到最终最短路径）

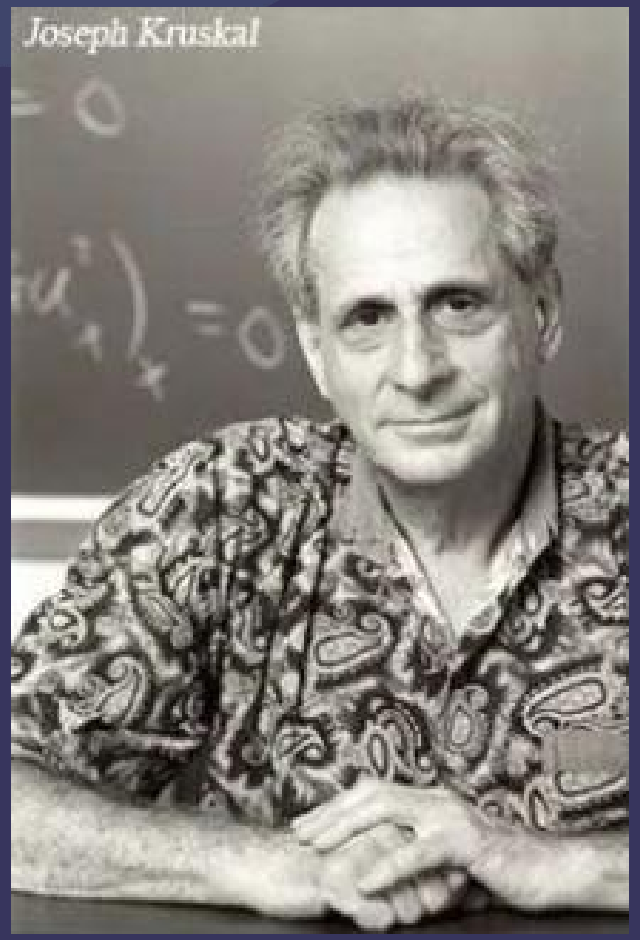
我们用f[k][i][j]表示i和j之间可以通过编号不超过k的节点的“最短路径”。对于k-1到k，只有两种可能：

1. 经过编号为k的点，要么不能找到一条从i到j的更短路，此时有f[k][i][j] = f[k-1][i][j]
2. 要么能找到，那这个最短路径一定是d[i][k]+d[k][j]

综合以上两种情况，f[k][i][j] = min(f[k-1][i][j] , f[k-1][i][k]+f[k-1][k][j])。


PART III
最小生成树

Kruskal算法



前言：什么是最小生成树

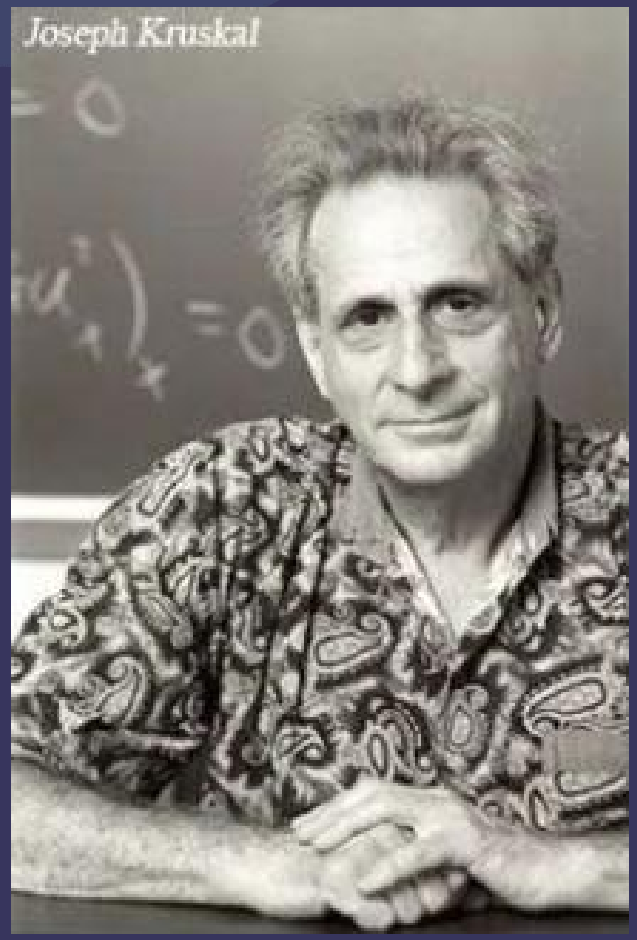
树：无环、N顶点（n-1边）

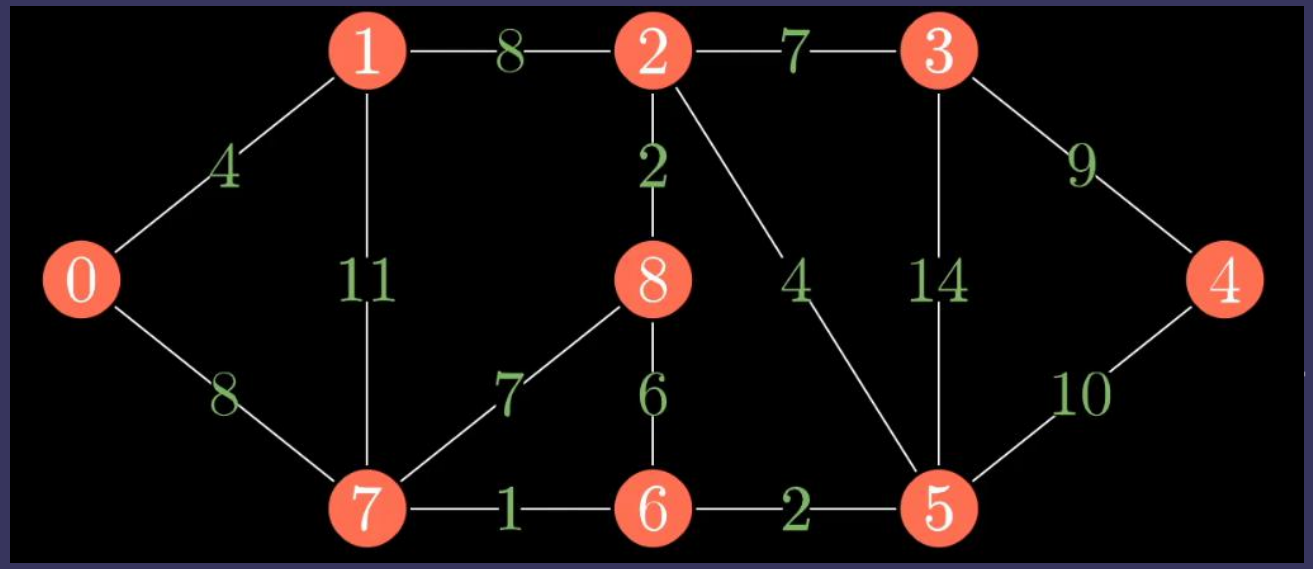
最小：不同生成树有不同权值和，最小生成树就是生成的权值和最小的树。

基本思路：

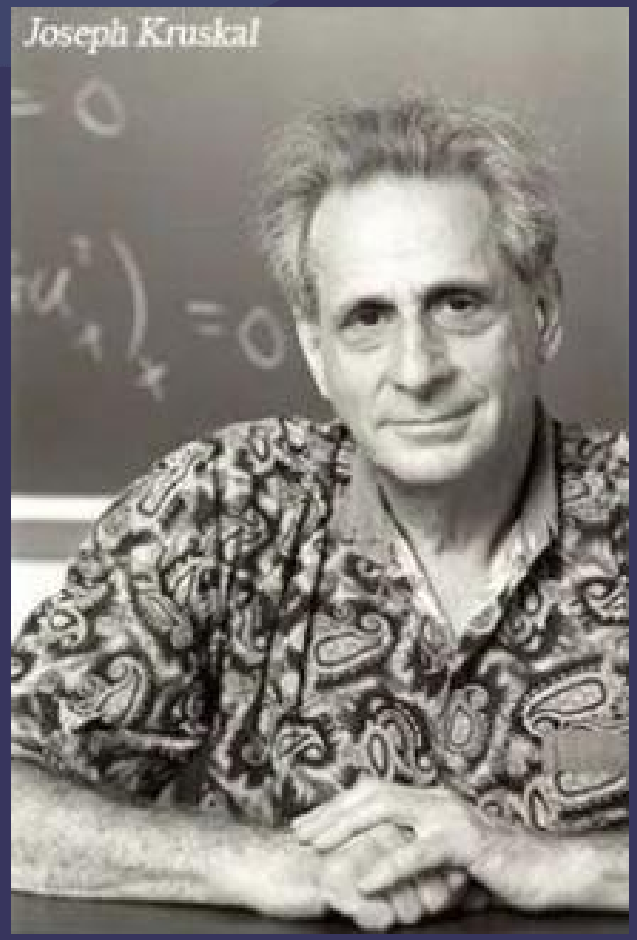
将边按权值从小到大排序，从小到大往树里塞边，若形成环，则舍弃这条边。（基于贪心算法）

Kruskal算法

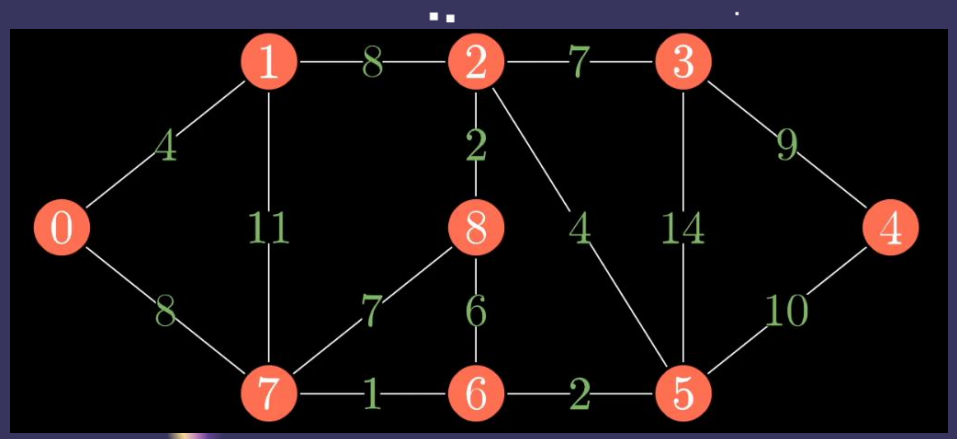




Kruskal算法

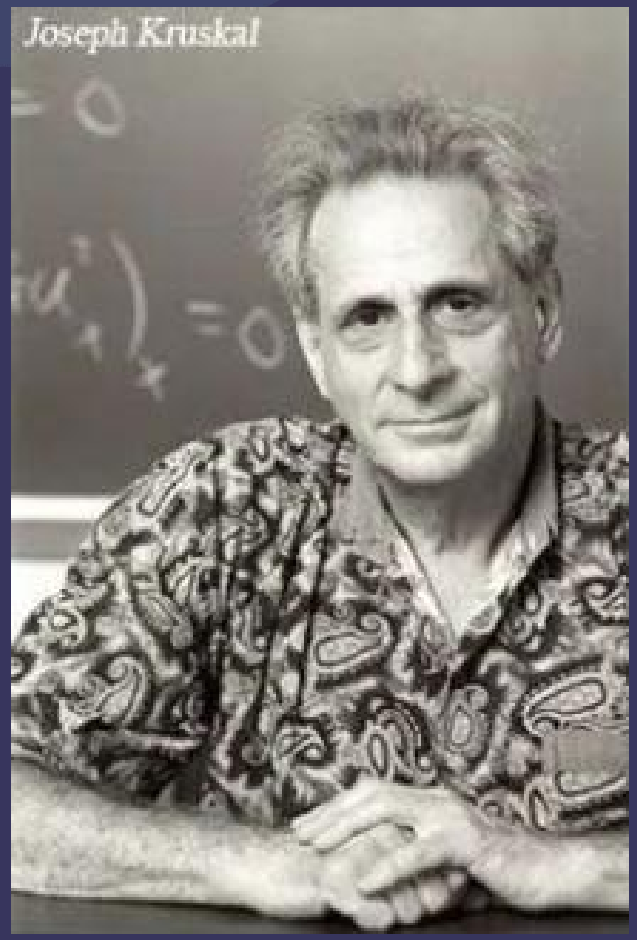


先以边权从小到大排序边:

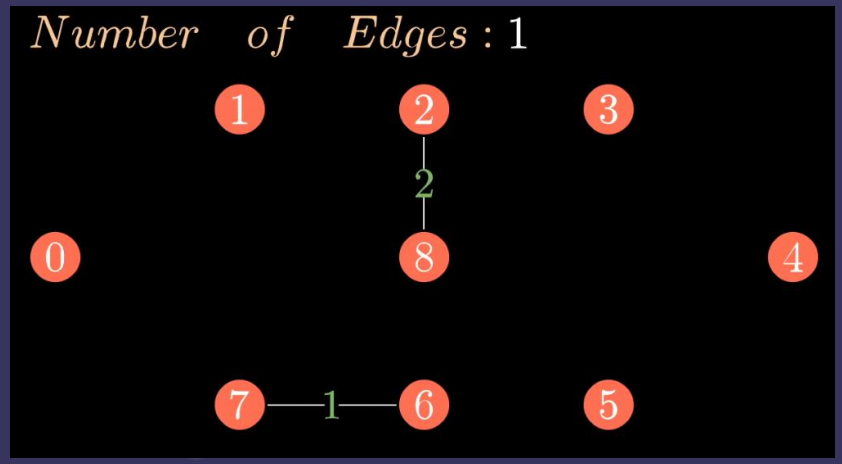


Edge	Weighted
6—7	1
2—8	2
5—6	2
0—1	4
2—5	4
6—8	6
7—8	7
2—3	7
0—7	8
1—2	8
3—4	9
4—5	10
1—7	11
3—5	14

Kruskal算法

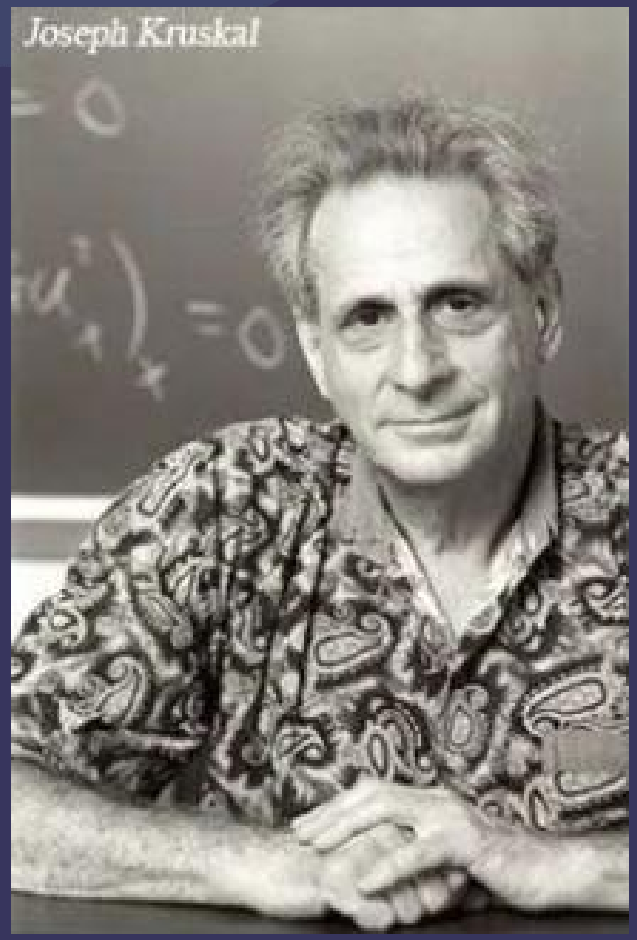


从小到大添加树的边：

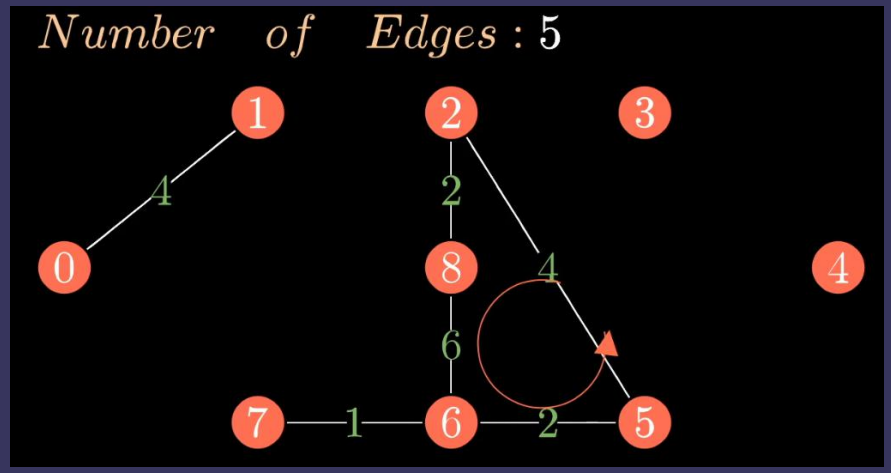


Edge	Weighted
6—7	1
2—8	2
5—6	2
0—1	4
2—5	4
6—8	6
7—8	7
2—3	7
0—7	8
1—2	8
3—4	9
4—5	10
1—7	11
3—5	14

Kruskal算法



如果添加的边使当前产生环，则不添加这条边

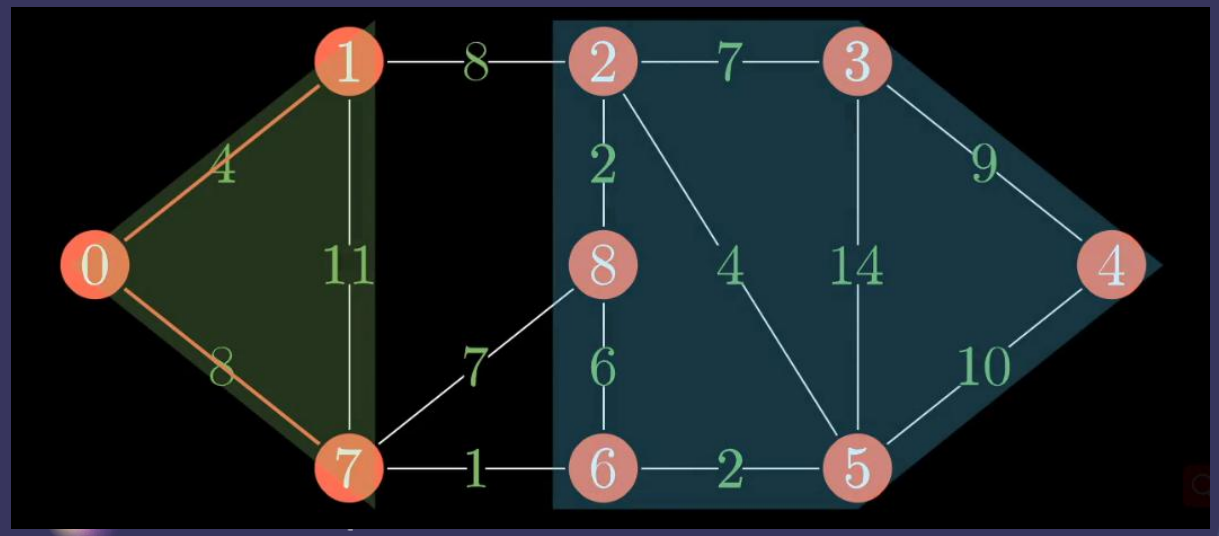


Edge	Weighted
6—7	1
2—8	2
5—6	2
0—1	4
2—5	4
6—8	6
7—8	7
2—3	7
0—7	8
1—2	8
3—4	9
4—5	10
1—7	11
3—5	14

Prim算法

基本思路:

也是优先选择权值最小的边，但是是从顶点出发，优先选择连接两个顶点集合的最小的边




THANKS