数学建模中的图论模型(B)

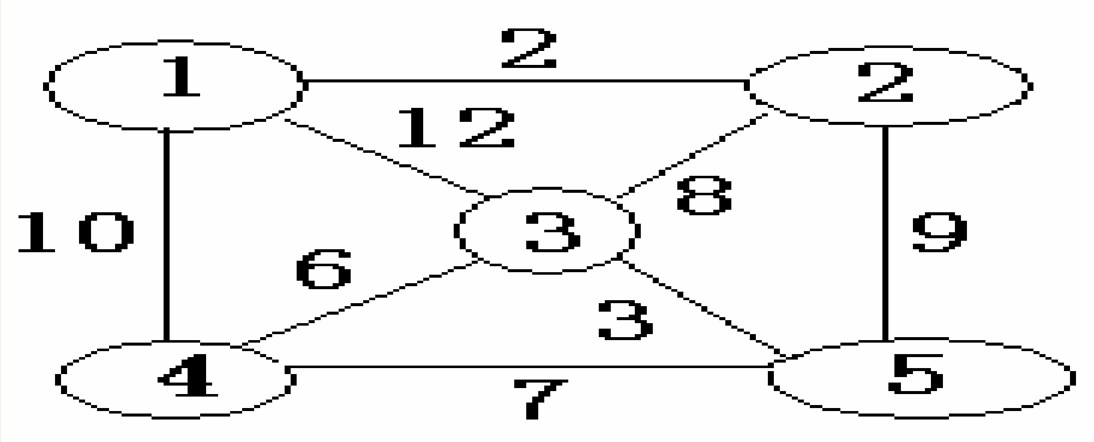
苏贵福
(gfs@mail.buct.edu.cn)

最小生成树算法

(Prim算法)

例1.最优工程造价

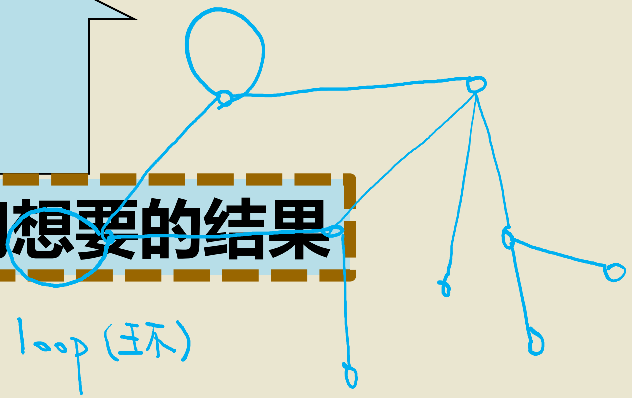
有一张城市地图，图中的顶点表示城市，无向边代表两个城市间的连通关系，边上的权表示在这两个城市之间修建高速公路的造价．研究发现该地图有一个特点：任一一对城市都是连通的．若要修建若干高速公路把所有城市联系起来，应如何设计可使工程的总造价最少．



1. 保证构建的图是连通的
2. 图中所有权值之和最小
3. 图中不能存在环路

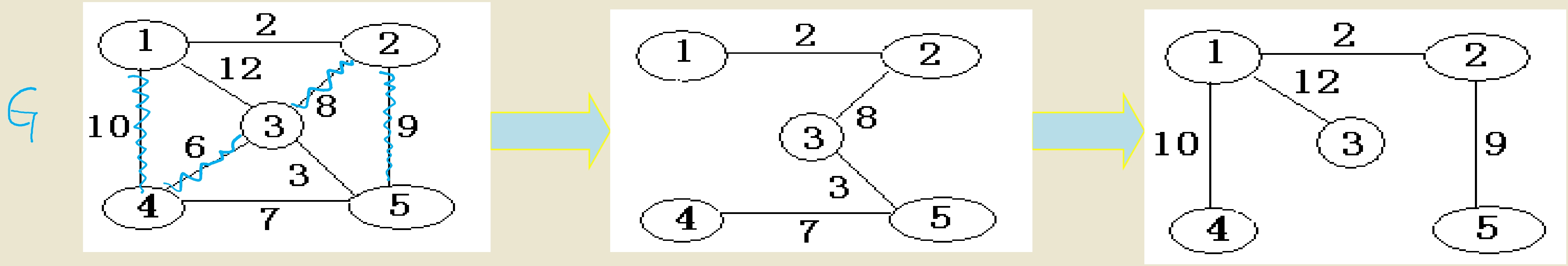
★这类问题可归纳为图的最小生成树问题!

这是我们想要的结果



1. 最小生成树

- 网络G=(V, E)是一个无向连通赋权图, E中每一条边(v, w)的权为c(v, w). 如果G的一个子图H是一棵包含G的所有顶点的树, 则称H为图G的**生成树**.
- 在G的所有生成树中, 各边权之和最小的树称为G的**最小生成树**.

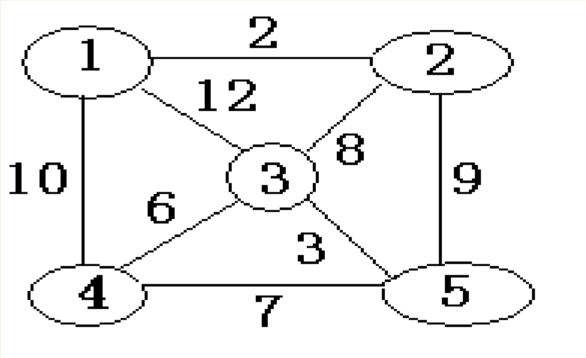


由此可以看出一个图的生成树是不唯一的.

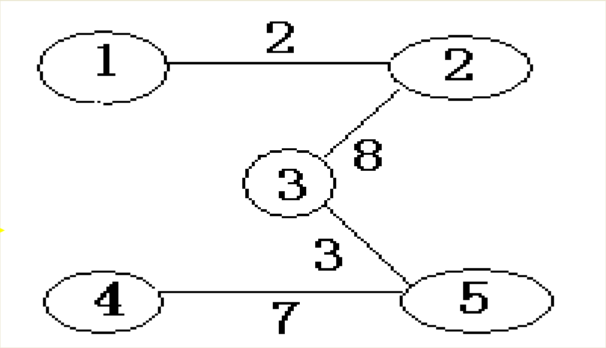
2. 如何求最小生成树

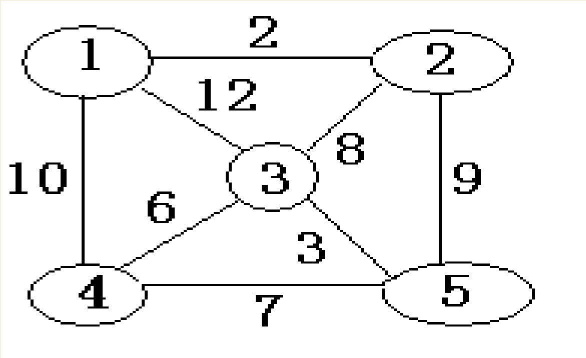
方法一:基于搜索的两种算法

- 深度优先遍历
- 广度优先遍历

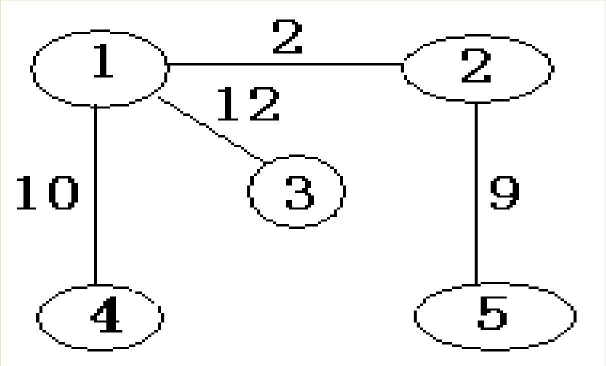


深度优先遍历

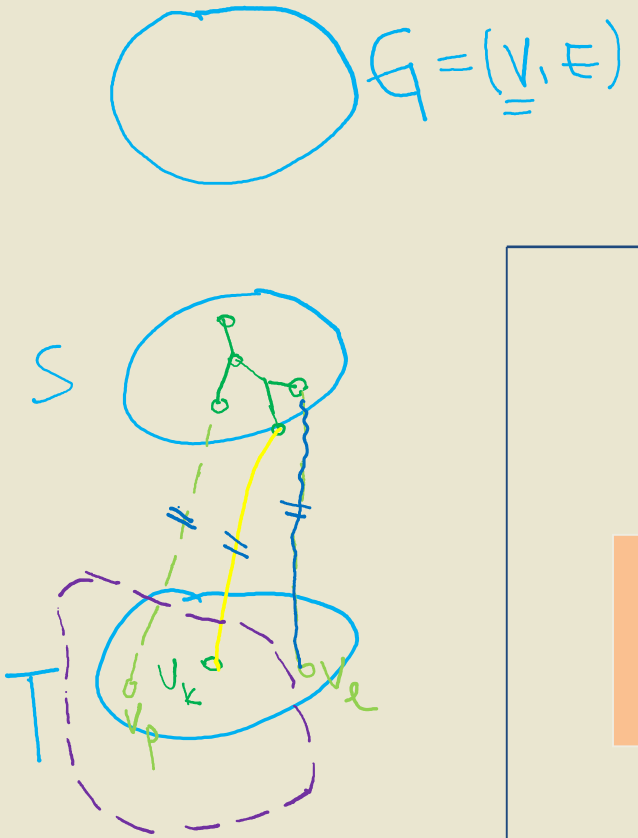


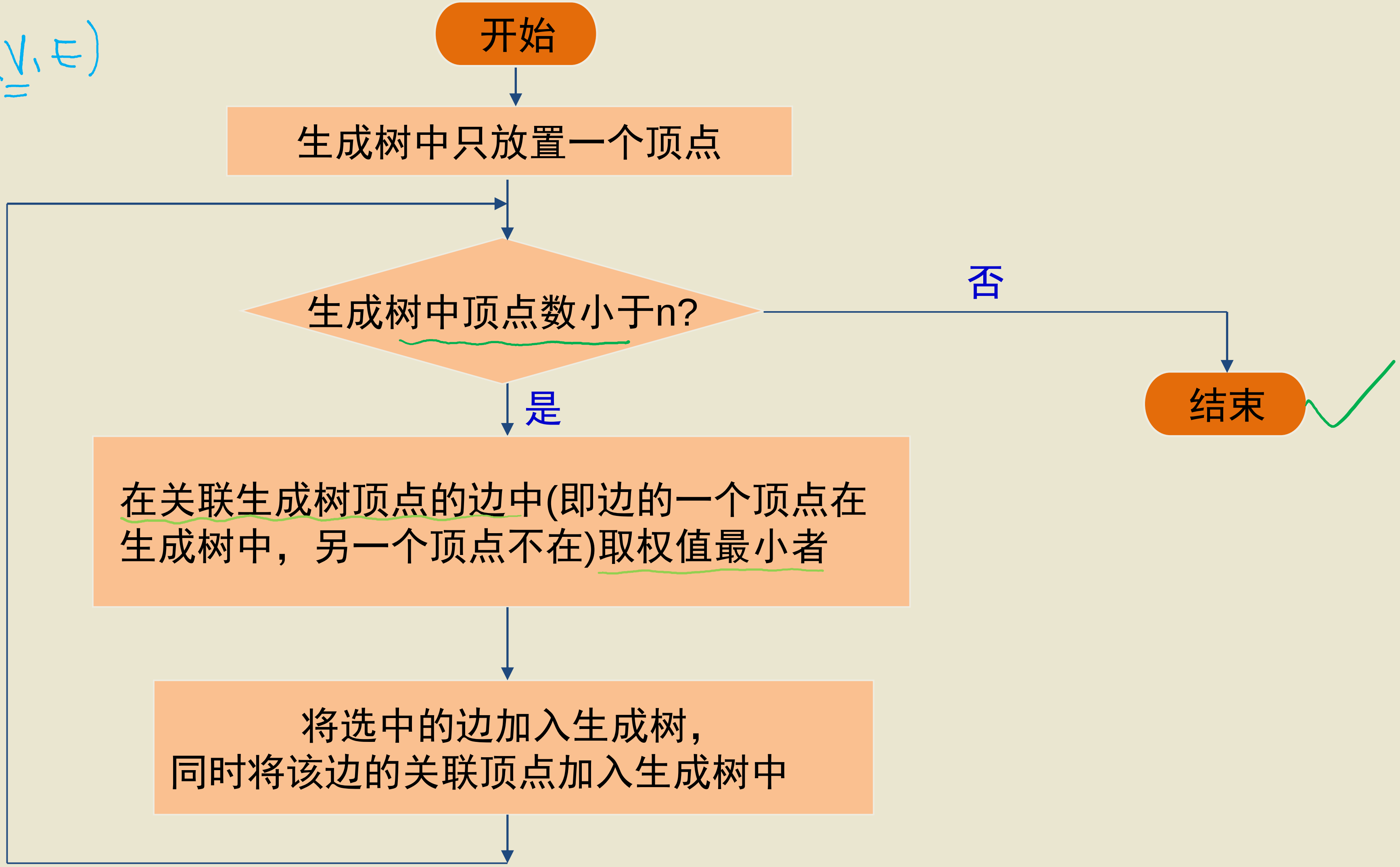


广度优先遍历



3. Prim算法框架图

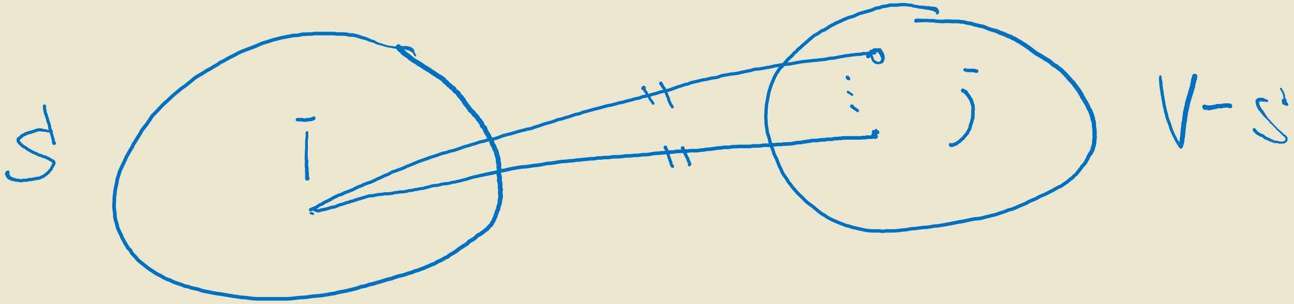

G=(V,E)
S
T


开始
生成树中只放置一个顶点
生成树中顶点数小于n?
否
结束
是
在关联生成树顶点的边中(即边的一个顶点在生成树中，另一个顶点不在)取权值最小者
将选中的边加入生成树，
同时将该边的关联顶点加入生成树中

4. Prim算法

- 设G=(V, E)是连通赋权图, V={1,2,….n}.

- **算法的基本思想是:**
- 首先置S={1}, 只要S是V的真子集, 则如下贪心选择:
- 遍历满足i∈S, j∈V-S的顶点, 找出c(i,j)最小的边, 将顶点j加入S中.
- 重复上述过程直到到S=V, 所选取的边构成的图即为G的最小生成树.



```
void prim(int n, Type **c)
{
   Typelowcost[MAX];
   int clost[max];
   bool s[max]; s[1]=true;
     for (i = 2; i <= n; i++)
   {
     lowcost[i] = c[1][i];
     clost[i] = 1;  s[i]=false;
   }
   for (i = 1; i <n; i++)
   {
     Type min =  inf;
     int j=1;
     for (int k= 2; k<= n; k++)
     {
       if (lowcost[k] < min && !s[k])
       {  min = lowcost[k]  j=k;  }
     }
     cout << "j" << ''<< closet[j] << endl;
     s[j]=true;
```

初始化

遍历邻近顶点，找出最短距离的边，输出顶点和边，并放入S

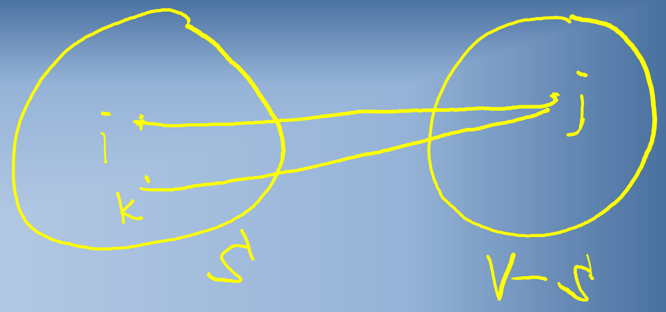
```
for (int k= 2; k<= n; k++)
      {
        if (c[j][k] < lowcost[j]) &&(!s[k])
        {
          lowcost[k] = c[j][k];
          closet[k]=j;
        }
      }

   }
```

对于新加入S的顶点通过比较最短距离，更新lowcost数组

4.1 Prim算法详细过程

- 设G=(V, E)是连通赋权图, V={1,2,….n}.

- 对于j∈V-S, Closet[j]表示S中与顶点j距离最短的顶点.



- Lowcost[j]=c(j, Closet[j]).

  V-S S

- 在算法的执行过程中, 先找出V-S中是Lowcost值最小的顶点, 选取边(j,Closet[j]), 将j添加到S中.

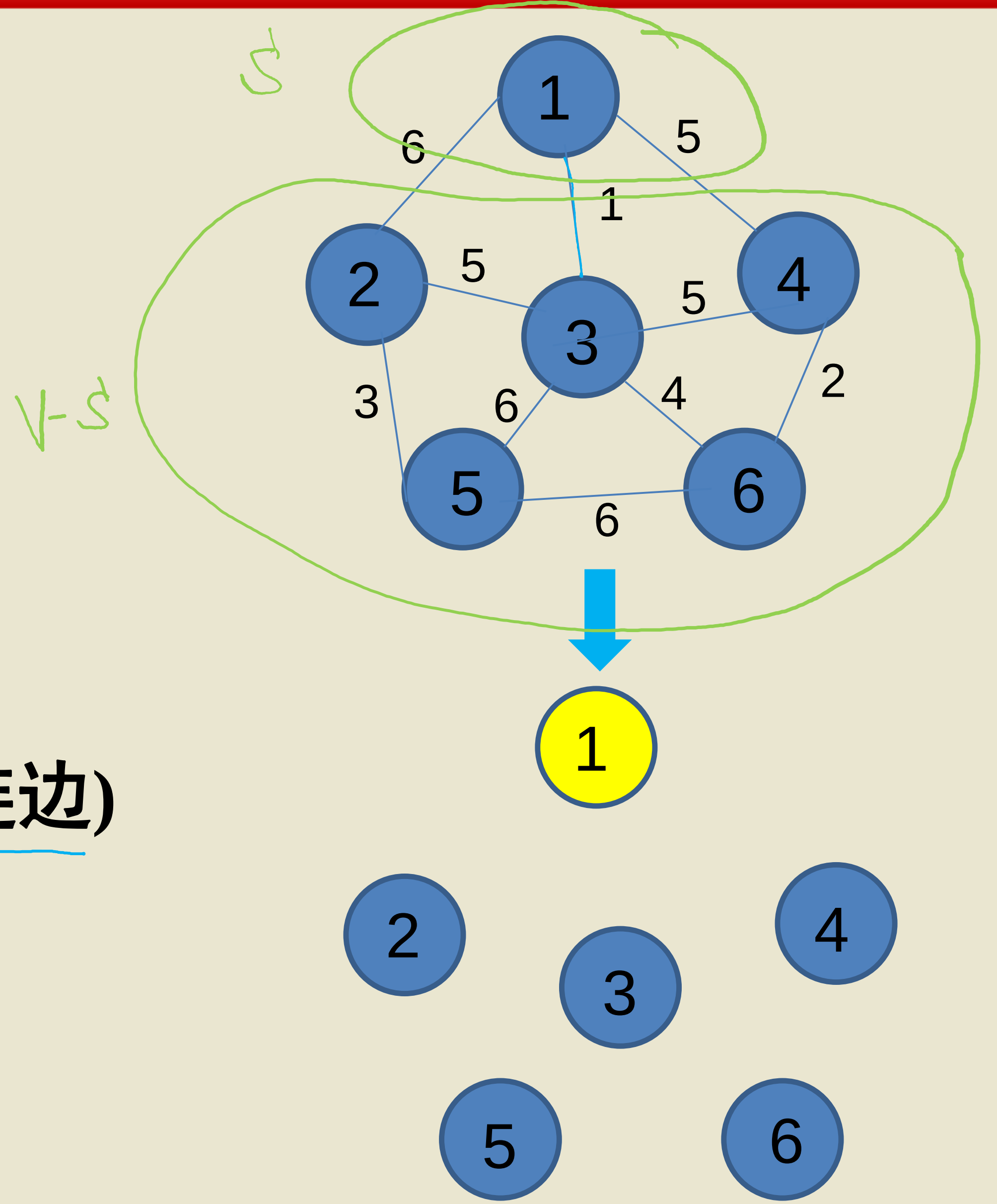
- 根据与新加入S的顶点的距离的比较, 更新Lowcost数组.

初始化

在Prim算法中最小生成树的起点设置为1. 此时S={1}, 由于S中只含有顶点1, 故 Lowcost[i]=c(1,i), Closet[i]=1:

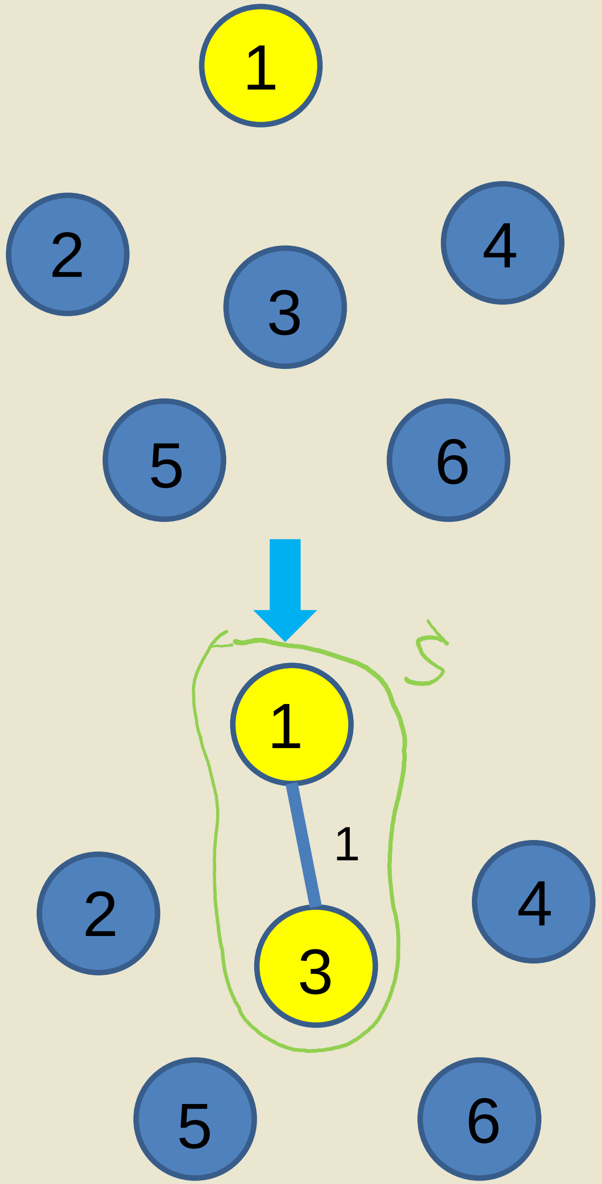
$$\begin{cases} Lowcost[1]=c(1,1)=0 \\ Lowcost[2]=c(1,2)=6 \\ Lowcost[3]=c(1,3)=1 \\ Lowcost[4]=c(1,4)=5 \\ Lowcost[5]=c(1,5)=max \\ Lowcost[6]=c(1,6)=max \end{cases}$$

(max表示无直接连边)



初始化Lowcost第一次循环

遍历Lowcost数组, 找到最小值为Lowcost[3]=1. 将顶点3加入S中, 并选定边(1, 3), 则S={1, 3}.



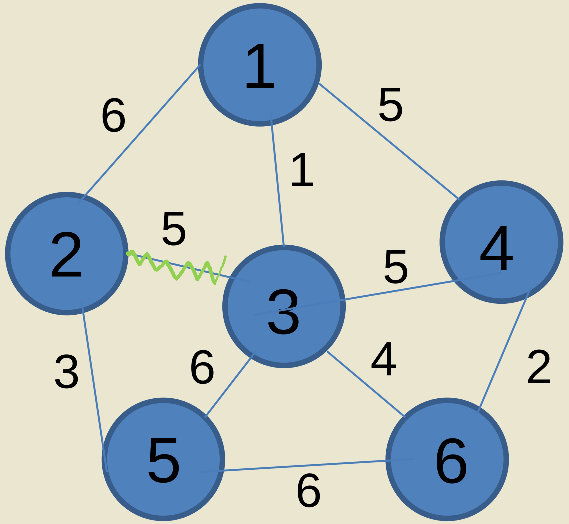
更新Lowcost数据

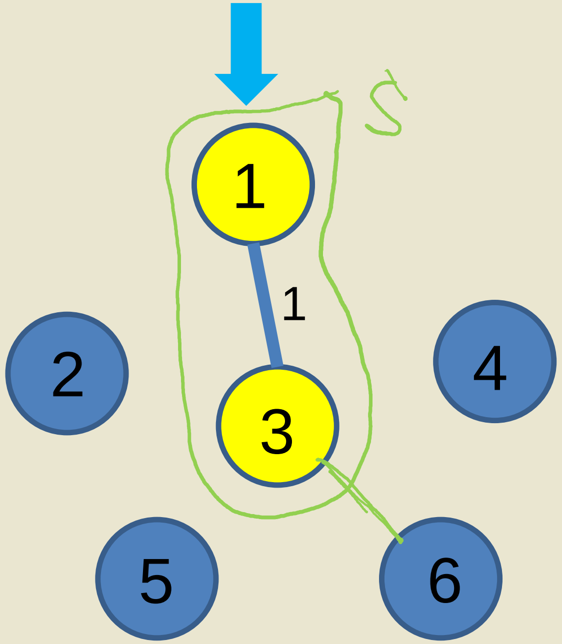
c(3,2)=5<c(1,2)=6 ⇒ Closet[2]=3

⇓

Lowcost[2]=c(2, Closet[2])=c(2,3)=5

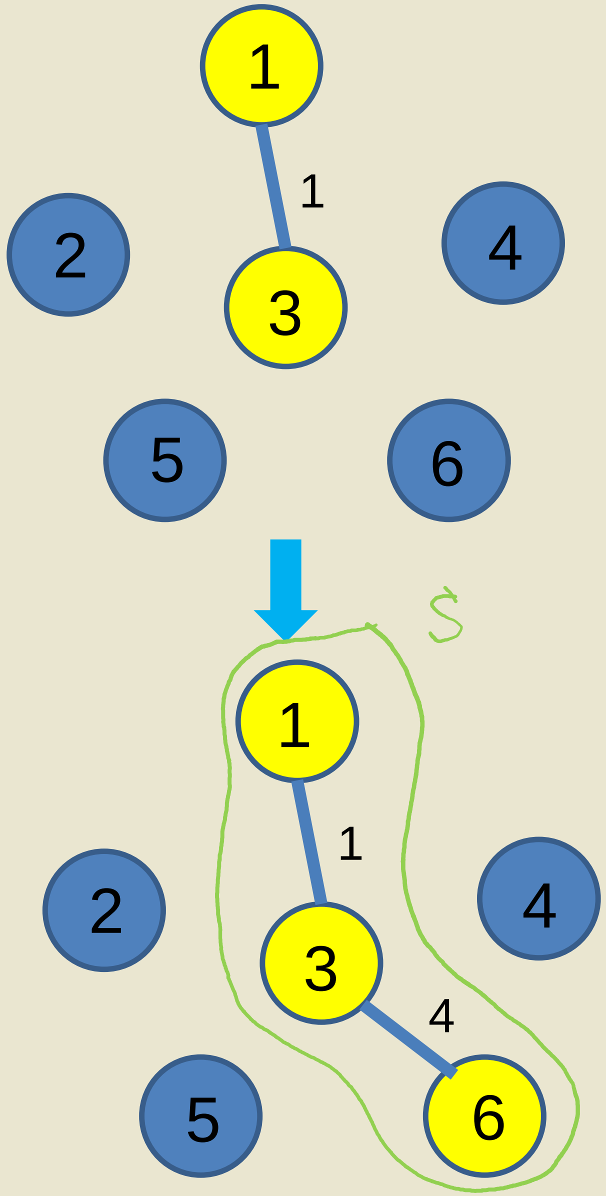
c(3, 2)=5<c(1, 2)=6, 更新Lowcost[2]=5
c(3, 3)=0,
c(3, 4)=5=c(1, 4), 不更新Lowcost[4] ✓
c(3, 5)=6<c(1, 5)=max, 更新Lowcost[5]=6 ✓
c(3, 6)=4<c(1, 6)=max, 更新Lowcost[6]=4





Lowcost数据第二次循环

因上一步Lowcost数组更新，故需第二次循环，找出最小值. 其中最小值为Lowcost[6]=4，将顶点6添加到S中，则S={1, 3, 6}.



更新Lowcost数据

$c(3, 2)=5<c(1, 2)=6<c(6, 2)=max$

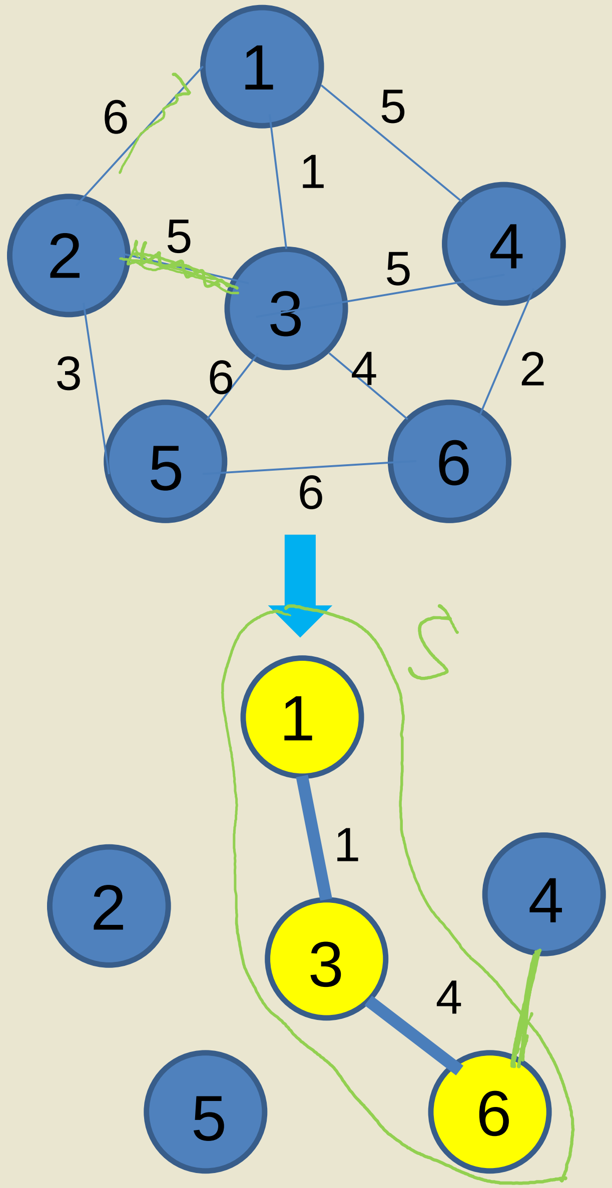
↓

Closet[2]=3

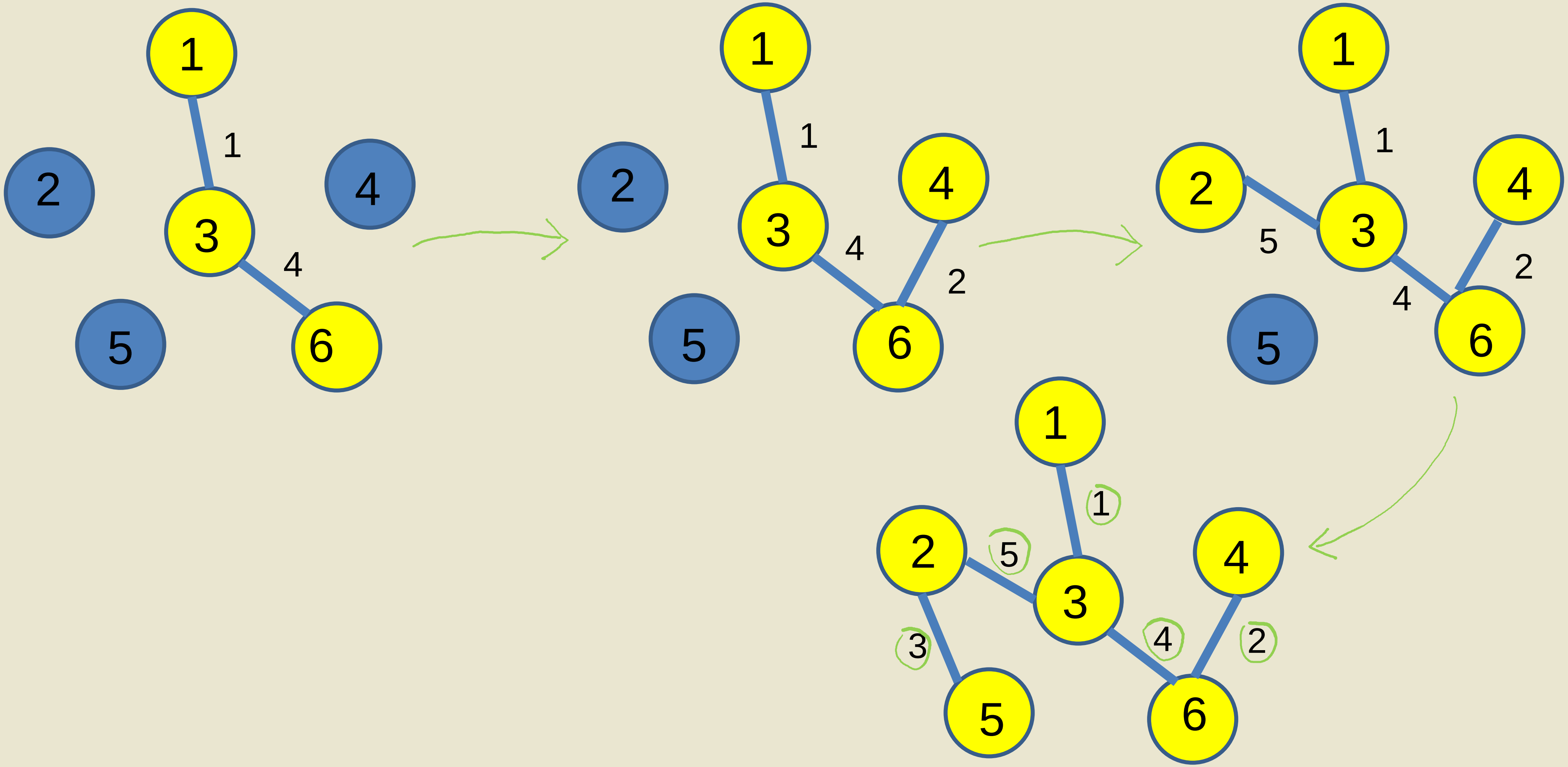
↓

Lowcost[2]=c(2, Closet[2])=c(2,3)=5

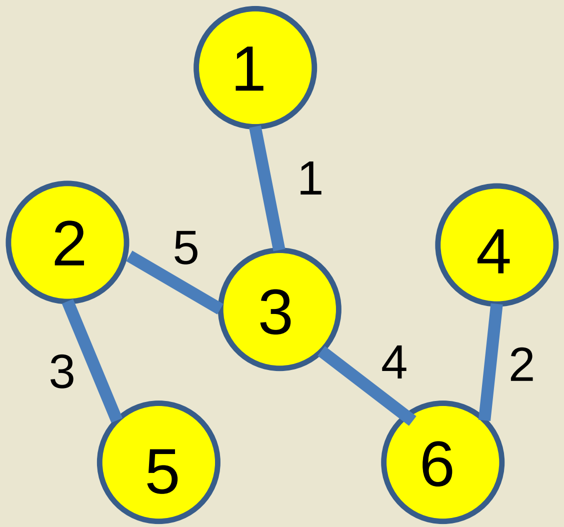
- Lowcost[2]=5, 不更新
- $c(6, 4)=2<c(3, 4)=5=c(1, 4)$, 更新Lowcost[4]=2
- $c(3, 5)=c(6, 5)=6<c(1, 5)=max$, Lowcost[5]不更新



最小生成树衍生过程



故所求的最小生成树为 S={1, 3, 6, 4, 2, 5}.



由于算法中利用了n*n的for循环, 易知所需的时间为O(n^2).

END