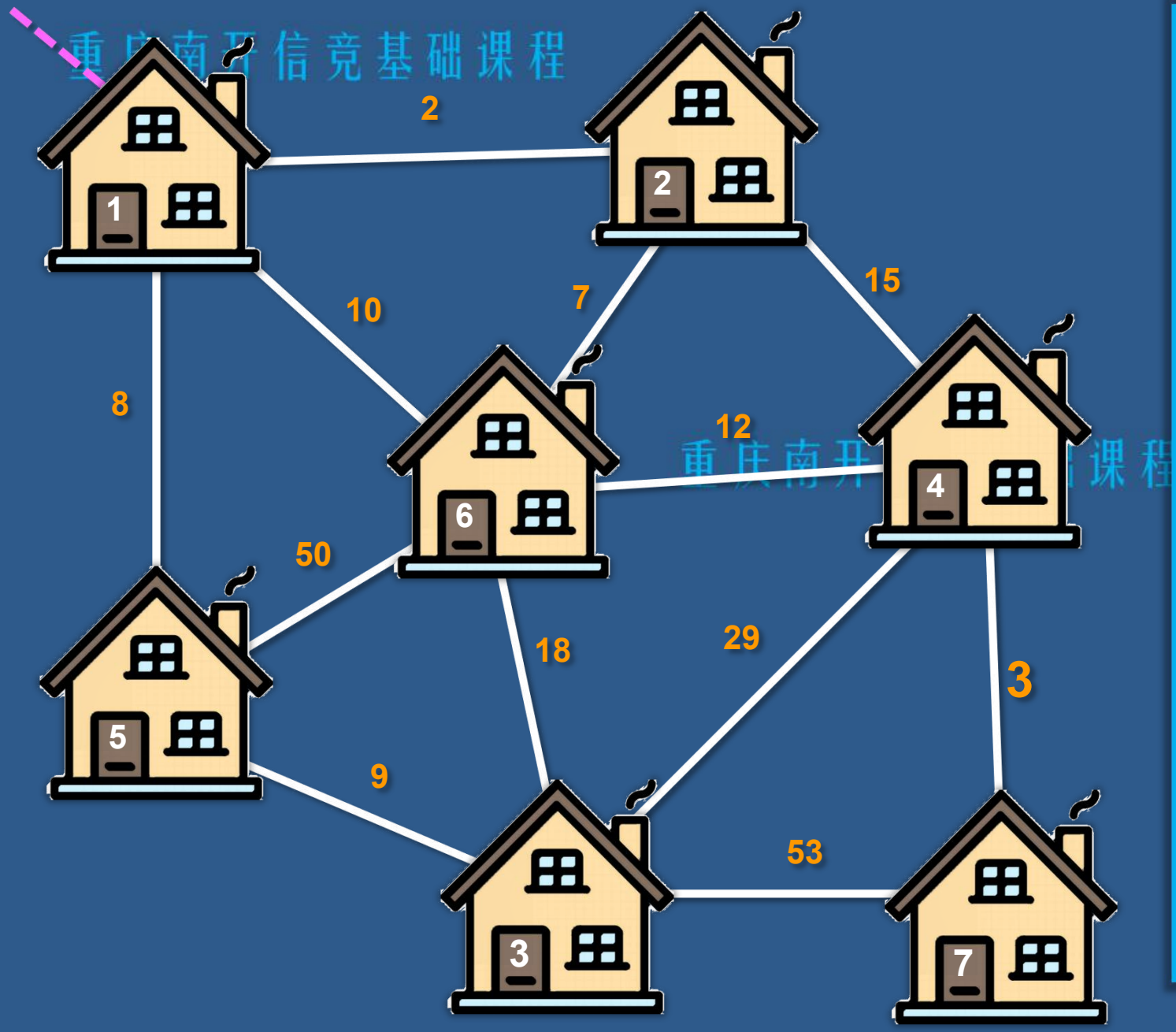
最小生成树

重庆南开信竞基础课程

Minimum Spanning Tree

Prim 和 Kruskal 算法

重庆南开信竞基础课程

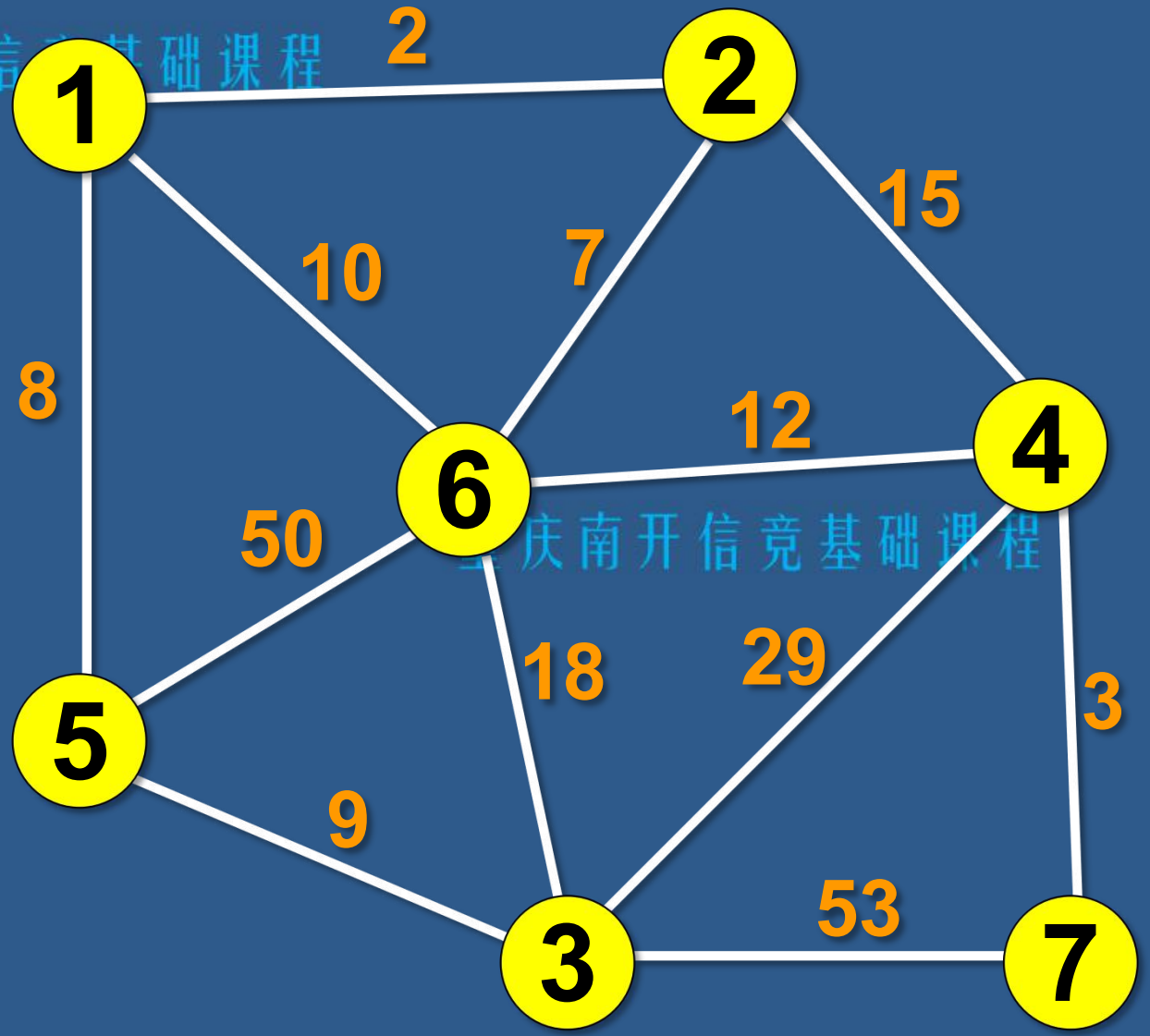


引例：村长的难题

何老板是信竞村的村长，何老板打算给该村的所有人家都连上网。

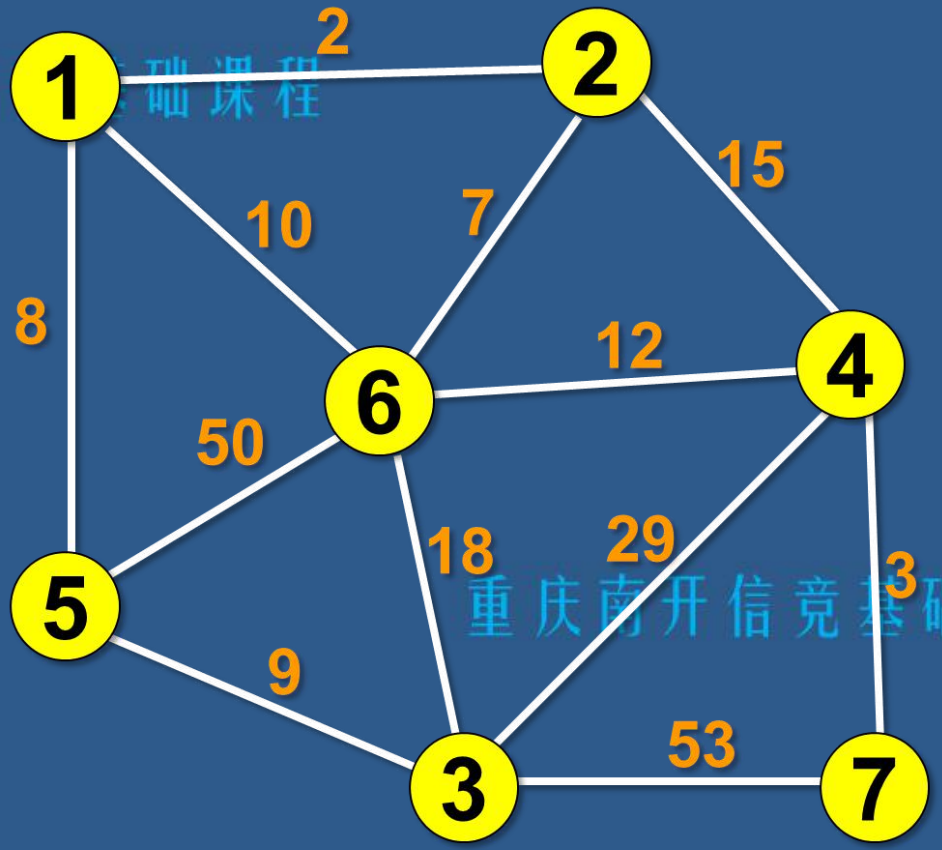
该村有n(1<=n<=1000)户人家，编号1到n。由于地形等原因，只有m(1<=m<=50000)对人家之间可以相互牵网线。在不同人家间牵线的长度不一定相同。比如在A与B之间牵线需要C米长的网线。

整个村的网络入口在1号人家，何老板的问题是：是否能使得所有人家都连上网？使所有人家都连上网，最少需要多少米网线？



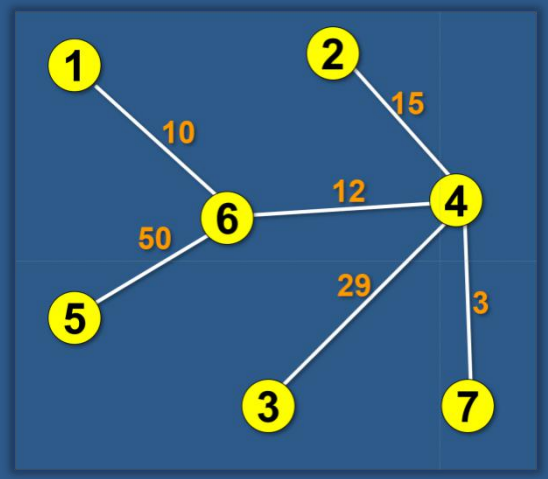
用网线连接n户人家，找出一种方案，使得总的长度最少。

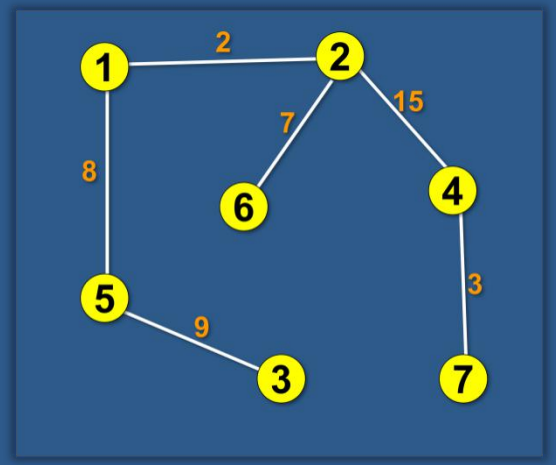
（无向图）

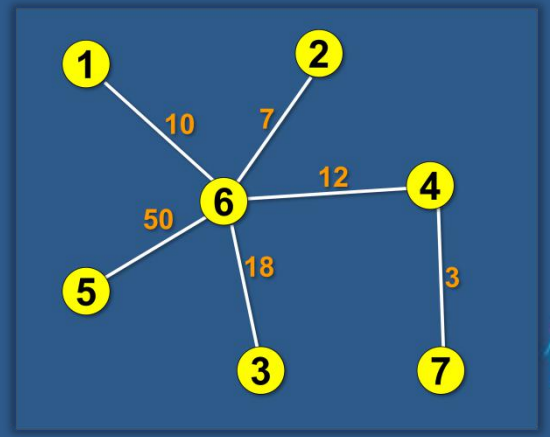


什么是生成树?

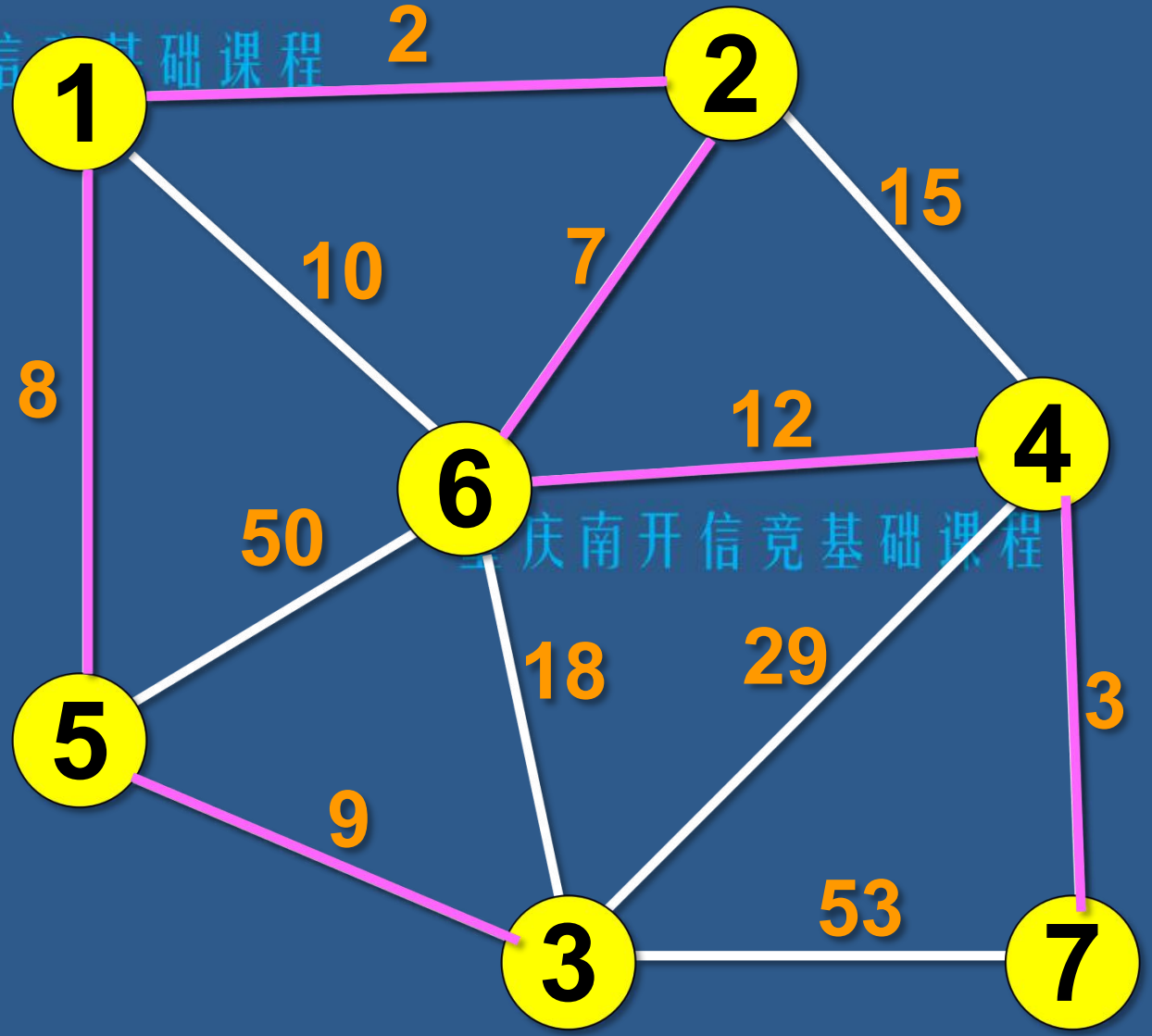
将无向连通图的一些边删掉，使得剩下的点和边构成一棵树，这样的树称为生成树。







.....

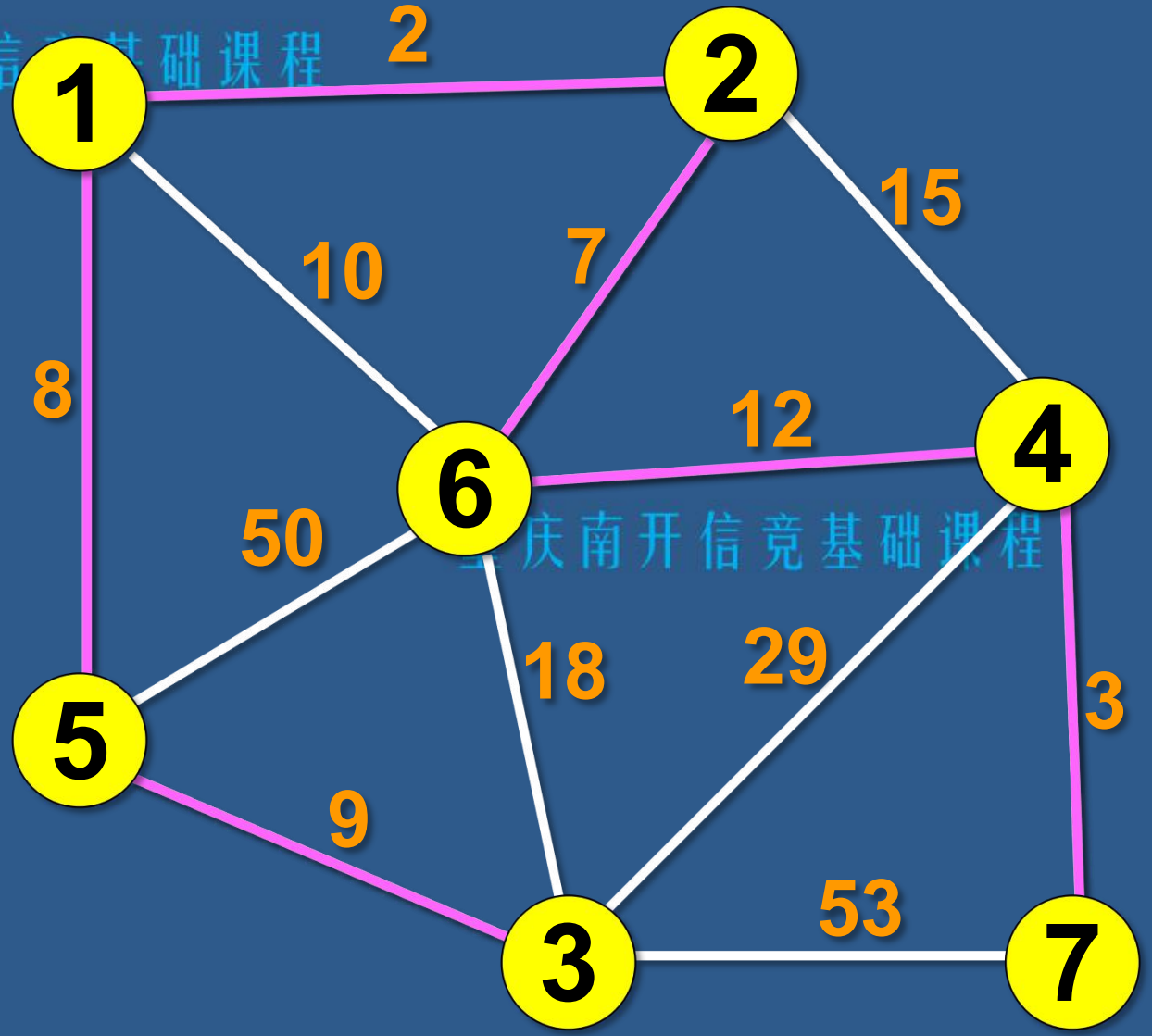


什么是最小生成树?

将无向连通图的一些边删掉，使得剩下的点和边构成一棵树，且这棵树的**边权总和最小**，这样的树称为**最小生成树。**

思考1，最小生成树是否唯一？
思考2，怎么求最小生成树？

用网线连接n户人家，找出一种方案，使得总的长度最少。



Kruskal 1956

注：在选择新的边加入时，该边的两个端点不能在同一棵已生成的树上！

O(mlogm)

每次选最短的边加入生成树

Kruskal

Kruskal算法基本思想：

每次选不属于同一生成树的且权值最小的边的顶点，将边加入生成树，并将所在的2个生成树合并，直到只剩一个生成树

排序使用sort

检查是否在同一生成树用并查集

总复杂度$O(m\log^m)$

$O(m\log^m)$ m表示边的数量

```
#define  maxn 1001
#define  maxe 10001
struct Line
{
        int a,b; //边的2个顶点
        int len; //边的长度
};
Line Edge[maxe];      //保存所有边的信息
int Father[maxn]      //Father存i的父亲节点
int n,m;              //n为顶点数，m为边数
```



```
bool cmp(Line a,Line b) //按边长由小到大排序
{
        return a.len<b.len;
}
```

```
void init()//初始化
{
   scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++)
         scanf("%d%d%d",&Edge[i].a,Edge[i].b,Edge[i].len); //读入图的信息
   for(int i=1;i<=n;i++)Father[i]=i; //初始化并查集
   sort(Edge+1,Edge+1+m,cmp);        //使用快速排序将边按权值从小到大排列
}
```

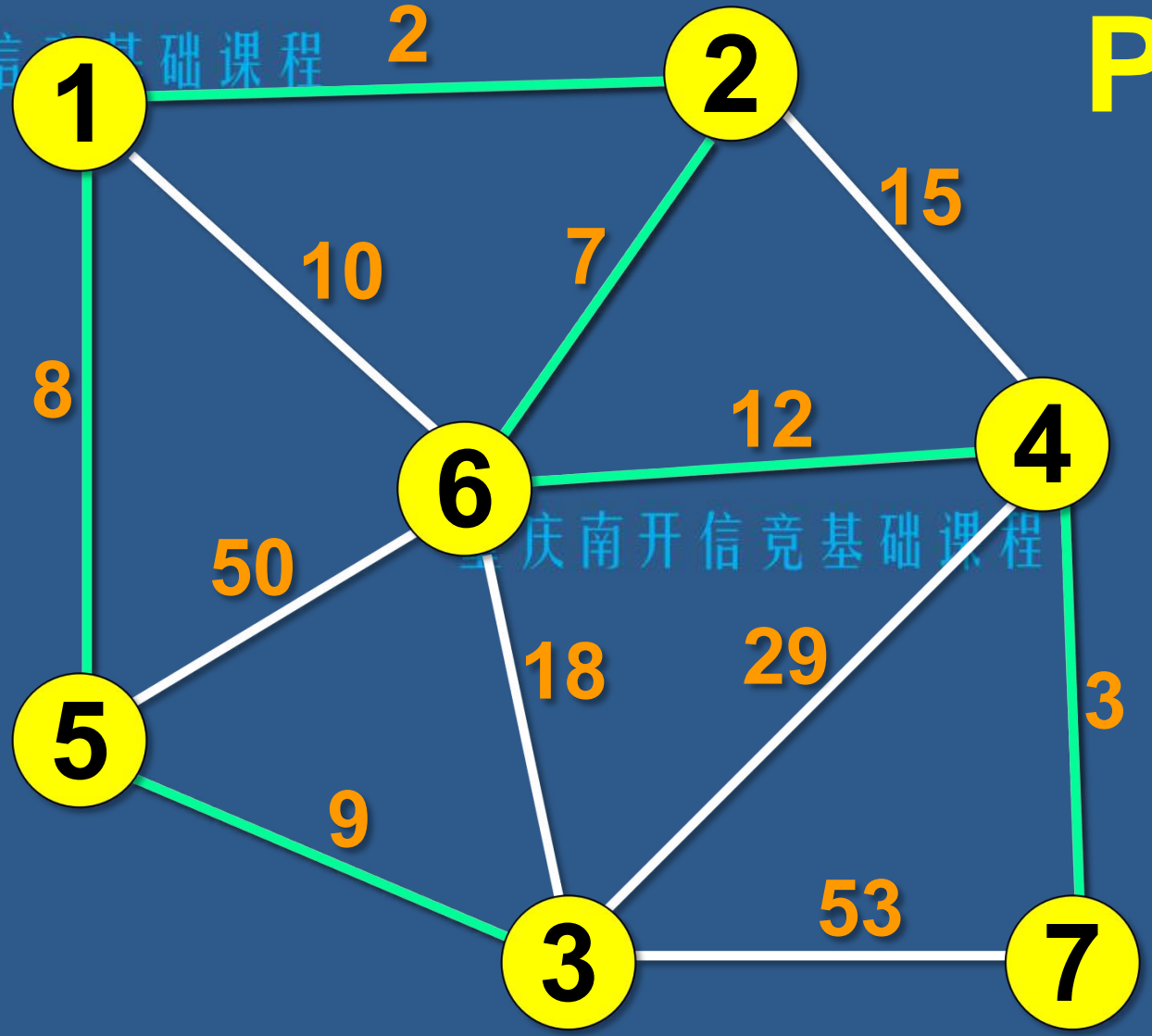
```
int getFather(int x)//并查集，用来判断2个顶点是否属于同一个生成树
{
   if(x!=Father[x])Father[x]=getFather(Father[x]);
   return Father[x];
}


void kruskal()
{
  int x,y,k,cnt,tot;          //k为当前边的编号,tot统计最小生成树的边权总和
                              //cnt统计进行了几次合并。n-1次合并后就得到最小生成树
  cnt=0;k=0; tot=0;
  while(cnt<n-1)              //n个点构成的生成树总共只有n-1条边
  {
        k++;
         x=getFather(Edge[k].a);
         y=getFather(Edge[k].b);
        if(x!=y)
        {
          Father[x]=y; //合并到一个生成树
          tot=tot+Edge[k].len;
          cnt++;
        }
  }
  printf("%d\n",tot);
}
```

```
int main()
{
   init();
   kruskal();
   return 0;
}
```


马上练一练nkoj2747

Prim 1957



1.任选一个点，加入生成树集合

2.在未加入生成树的点中，找出离生成树距离最近的一个点，将其加入生成树。

3.反复执行2，知道所有点都加入了生成树

```
void prim(int x)
{
      int   dis[101],path[101],i,j,k,Min;
      for(i=1;i<=n;i++)
      {   dis[i]=map[i][x];  path[i]=x;   }
      for(i=1;i<=n-1;i++)
    {
          Min=inf;
          for(j=1;j<=n;j++)
               if ((dis[j]!=0)&&(dis[j]<Min))
               {   Min=dis[j];    k=j;    };
          dis[k]=0;
          for(j=1;j<=n;j++)
               if(dis[j]>map[j][k])
               { dis[j]=map[j][k];
                 path[j]=k; }
    }
}
```

//开始时任选一点x加入生成树，故一开始树中只有一个点x

//dis记录各节点到生成树的最小距离

//path[i]记录生成树中与节点i最近的一个节点的编号，用于记录路径

//初始化，将每个节点到生成树的最小距离赋值为它到点x的距离,将生成树中与i最近的节点赋值为x(因为此时生成树中只有一个节点x)
//除x外，还有n-1个节点要讨论

//inf为自定义的一个表示无穷大的数。

//找出在未加入生成树的节点中，离当前生成树距离最近的一个节点

//将找出的离生成树距离最近的节点t到生成树的距离赋值为0，表示它已经加入到树中了

//k加入生成树后，可能有其他节点到生成树的最短距离发生变化，调整他们的path值

//讨论未加入到生成树的节点j与k的距离是否比j原来到生成树的最短距离dis[j]要短，如果是，则用新的更短的距离取代原来的距离，并将生成树离j最近的节点改成t

输出最短路径总长度

```
for(i=1;i<=n;i++)
    if(path[i]!=i)
        total=total+map[i][path[i]];
cout<<total;
```


prim时间复杂度$O(n^2)$

堆优化O(nlogn)

马上练一练nkoj2747