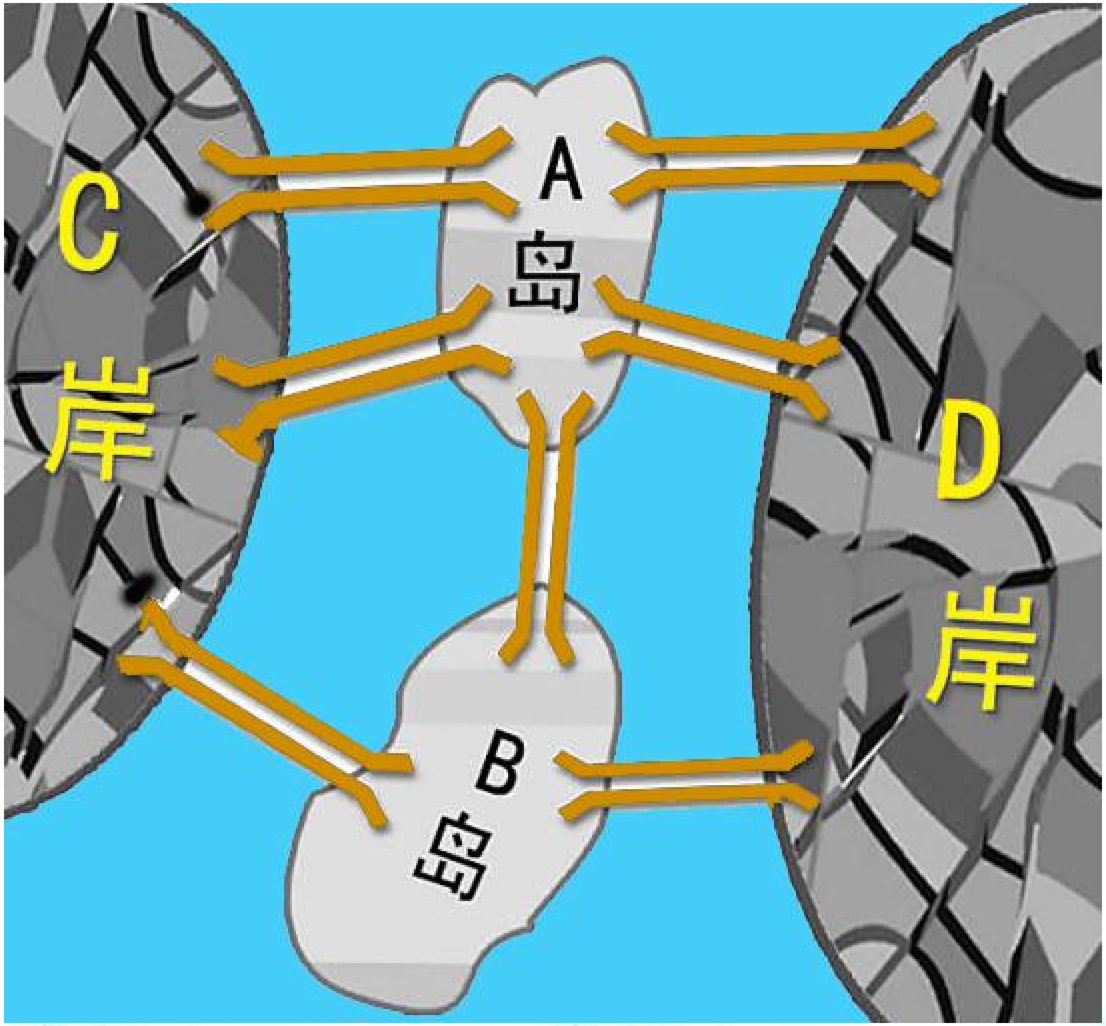
图论2

欧拉路与欧拉回路

七桥问题

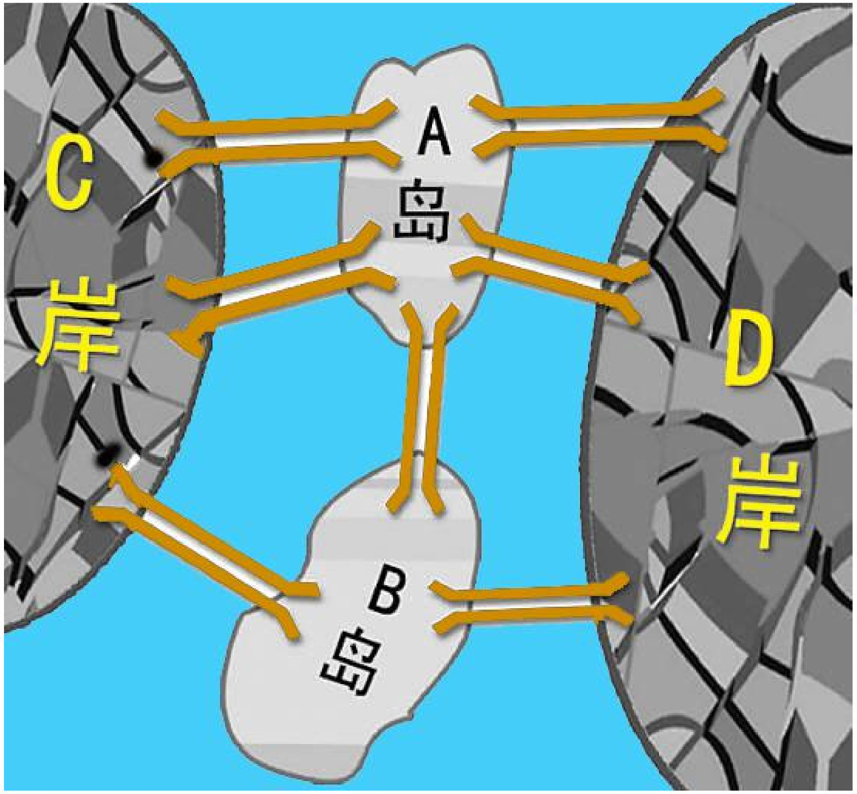


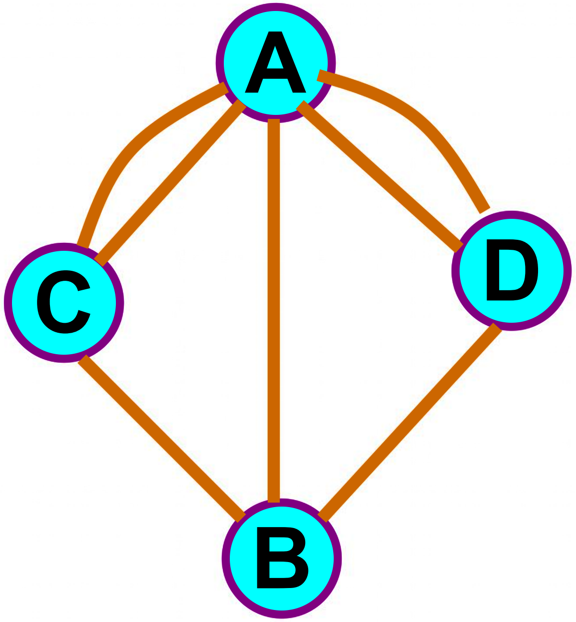
koenigsberg

1736年，欧拉:

是否可从某个地方出发，经过每座桥一次,回到原来出发的地方？

由顶点和边构成的集合，称为图

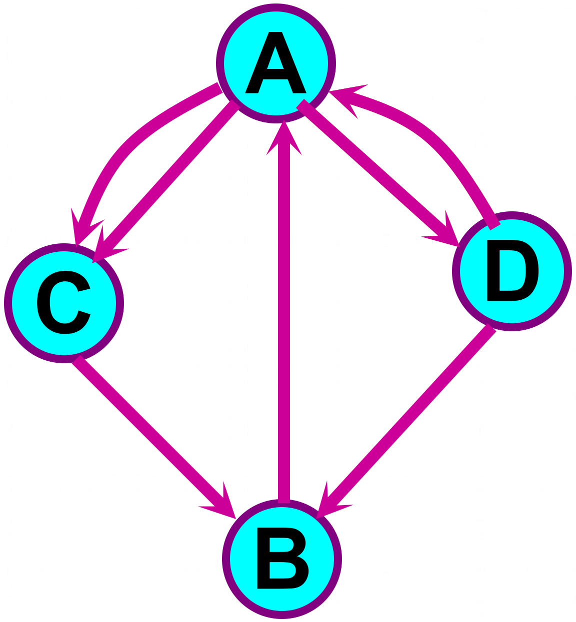


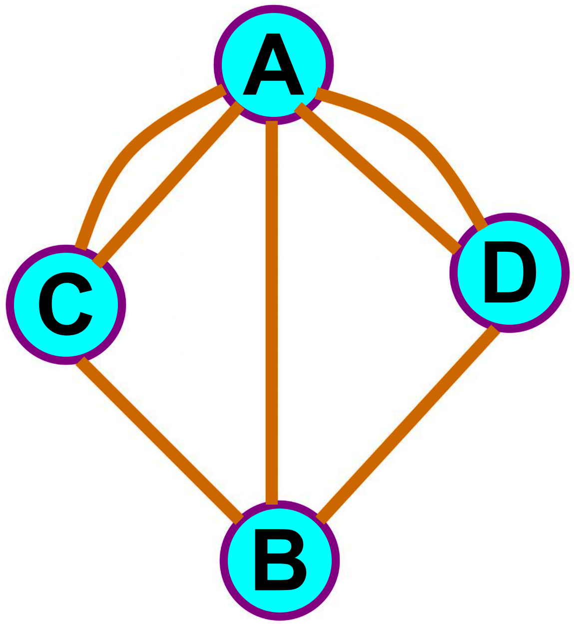


由顶点和边构成的集合，称为图

图中所有的边都是无向的，该图称为无向图

图中所有的边都是有向的，该图称为有向图





顶点的度，就是指和该顶点相关联的边数

出度，有向图中，从某顶点出发的边数

入度，有向图中在某顶点终止的边数

七桥问题与欧拉回路

若恰通过图中每条边一次**回到起点**,则称该回路为**欧拉(Euler)回路。**
具有**欧拉回路**的图称为**欧拉图。**
定理1：
一个无向图是欧拉图，当且仅当该图所有顶点度数都是**偶数。**

一个有向图是欧拉图，当且仅当该图所有顶点度数都是**0**(入度与出度之和)。

若从起点到终点的路径恰通过图中每条边一次（**起点与终点是不同的点**），则该路径称为**欧拉路。**
定理2：
存在欧拉路的条件：图是连通的，且存在**2个奇点。**
如果存在2个奇点，则欧拉路一定是从一个奇点出发，以另一个奇点结束。

定理3：存在欧拉回路的条件：图是连通的，且不存在奇点。

怎样找出欧拉路或欧拉回路？Hierholzer 算法

1.**判断**：先判断该图包含欧拉路还是欧拉回路
2.**找起点**：若是欧拉路则以两个奇数度点中任意一个为起点。若是欧拉回路则以任意点为起点。
3.**遍历**：从起点开始遍历，将已走过的边删除，直到所有边都被删除。

```
//相当于深搜，当搜索完一次后，实际上所有边已被删除，回溯回来只是记录路径
void euler(int x)        //遍历，当前讨论到x号点
{
   int i;
   for(i=1;i<=n;i++)  //枚举跟x相连的点
      if(m[x][i]!=inf)       //如果x,i间存在边，则将其删除，相当于已走过
      {
         m[x][i]=inf;    //这里是无向图，将来回两条边删除
         m[i][x]=inf;
        euler(i);
      }
   step++;           //记录路径步数
   path[step]=x;     //记录当前这一步的节点编号
}
```

怎样判断该图是否连通？

怎样判断是否连通情况？

```
//相当于深搜，当搜索完一次后，实际上所有边已被删除，回溯回来只是记录路径
void euler(int x)        //遍历，当前讨论到x号点
{
   int i;
   for(i=1;i<=n;i++)  //枚举跟x相连的点
      if(m[x][i]!=inf)        //如果x,i间存在边，则将其删除，相当于已走过
      {
          m[x][i]=inf;    //这里是无向图，将来回两条边删除
          m[i][x]=inf;
          Cnt++;
         euler(i);
      }
   step++;              //记录路径步数
   path[step]=x;       //记录当前这一步的节点编号
}
```

```
if(Cnt!=m)cout<<"不连通！";//m为图中边的总数
```

哈密顿图：在连通图中，若存在一条路，经过图中每个**点**一次且仅一次，则称此图为**哈密顿**图；
哈密顿回路：一条沿图的n条边环行的路线，它经过每一个点一次且仅一次并且回到他的开始位置。

```
//找哈密顿路
void dfs(int x,int step)//当前讨论x号点，step记录走到当前点的步数
{
   int i;
   if(step>n){打印路径......}
   for(i=1;i<=n;i++)
         if ((m[x][i]!=inf&&(visit[i]==false))
         {
           path[step]=i; visit[i]=true;
           dfs(i,step+1);
           visit[i]=false;
         }
}
```

练习：1104,1106

旅行商问题（Traveling Saleman Problem）

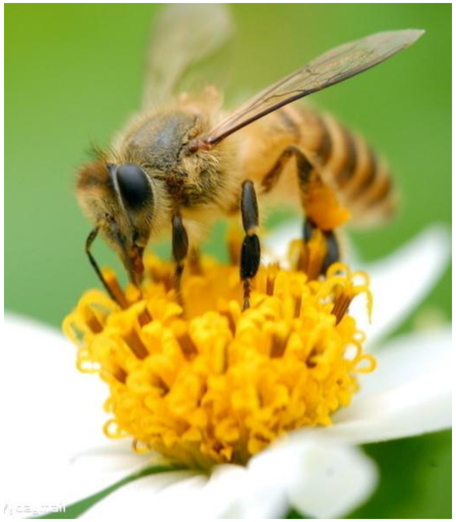
一个推销员要去n个城市推销产品。他要从其中某一个城市出发，经过每个城市一次，再回到他出发的城市，求最短的路线。也即求一个最短的哈密顿回路。

最明显的算法就是穷举法，即寻找一切组合并取其最短。这种算法的排列数为n!(其中n为节点个数)。

用动态规划技术，我们可以在$O(n^2*2^n)$时间内解决此问题。虽然这仍然是指数级的，但要比O(n!)快得多。

英国伦敦大学皇家霍洛韦学院等机构研究人员报告说，小蜜蜂显示出了轻而易举破解这个问题的能力。

他们利用人工控制的假花进行了实验，结果显示，不管怎样改变花的位置，蜜蜂在稍加探索后，很快就可以找到在不同花朵间飞行的最短路径。这是首次发现能解决这个问题的动物，研究报告发表在《The American Naturalist》杂志上。



奈杰尔·雷恩博士说，蜜蜂每天都要在蜂巢和花朵间飞来飞去，为了采蜜而在不同花朵间飞行是一件很耗精力的事情，因此实际上蜜蜂每天都在解决“旅行商问题”。