# 图的表示以及优劣

设图G的点数是V，边数是E

最好的结构是邻接表和链式前向星了。

## 邻接矩阵：

用一个正方形二维数组表示图，int map[V+1][V+1] 上限是图的点数+1，因为一般图论里图的点下标从一开始，当然也可以把数组直接开够了。

map[u][v]=val;表示点u到点v存在一条有向边，权值是val,

初始化map数组是-1，这个认为定义。map[u][v]=-1表示不存在

### 优点

* 优点是已知两个点编号u和v直接可以知道权值，访问单独的边十分容易

### 缺点

* 需要内存过大，储存不是特别稠密的图会由很多浪费，这点是致命的，题目中一旦给出V<=100000，就宣告不能使用这种结构。
* 遍历某个点到其他边时候，会由浪费，因为遍历数组某一行或者列，势必会访问到那些无用的不存在的边

## 邻接表：

struct Edge{

Int b,e,val;//b是边的起点，e是终点，val是权值

}

vector< Edge>adList[V+1]

adList[i]表示以i为起点的边所形成的边集，利用vector可变长度的性质，动态添加删除边，可以根据删除插入的频繁程度来选取用c++STL的list代替vector

访问以u为起点，v为终点的边，需沿着起点遍历adList[u][i],i代表边集第几个；

因此要迭代到adList[u][i].e=v

实现寻找反向边的功能需要在输入时候预处理，且Edge增加一个属性k,表示以b为终点的边是adList[e][k]

### 优点

* 不浪费内存有多少点存多少边，就算V是100000也可以用
* 遍历以u为起点的所有边，这个操作十分方便
* 如果人为vector效率低可以改用list甚至自己实现链表

### 缺点

* 访问以u为起点，v为终点的边，也就是单边访问较慢。解决这个问题可以用链式前向星

## 边集数组：

struct Edge{

Int b,e,val;//b是边的起点，e是终点，val是权值

}

edgeSet[E+1];

edgeSet[i]表示编号为i的边，

这个一点不好用，被邻接表完爆

## 前向星：

struct Edge{

Int b,e,val;//b是边的起点，e是终点，val是权值

}

edgeSet[E+1];head[V+1]

edgeSet[i]表示编号为i的边，

对edgeSet边集按照 先b后e 的顺序排序

head[u]表示点u为起点的第一条边的下标，这样能实现类似邻接表的效果

基本被邻接表完爆，它需要个E\*logE的排序

## 链式前向星：

struct Edge{

Int e,i ,val;// e是终点，下一条边所在edgeSet 数组中的位置，val是权值

}

edgeSet[E+1];head[V+1]

edgeSet[i]表示编号为i的边，

head[u]表示点u为起点的最后一条边的下标，

此外，如果有必要，对于一切偶数eveni, 链式前向星edgeSet[eveni]和edgeSet[eveni+1]表示两条方向相反的边；

* 有邻接表的全部功能，而且由于数组实现效率高于邻接表
* 能实现查找某的个点的反向节点的功能，而用邻接表实现要储存额外信息