INT10 2W5 _贪 心 算 法

局部最优到全局最优，最小找零数，小子

定义：对于一个较为复杂的问题，分开来看并在每一部分选取最符合当下问题的解，最后由此得到全局的解

应用场景：在给定范围的情况下求取最小值或者最大值，比较经典的例子就是背包问题（这是反例，用来说明贪心算法的解并不一定最优但是一定是相对比较能够接受的）和硬币找零问题，然后贪心算法也在图中有所应用

优点：对于贪心算法来说，我们不需要在每一步上都有较多的努力，通常也可以用较短的时间来得到我们想要的解

缺点：我们得到的解并不一定是最优解

在图论中的应用：

求解最小生成树：

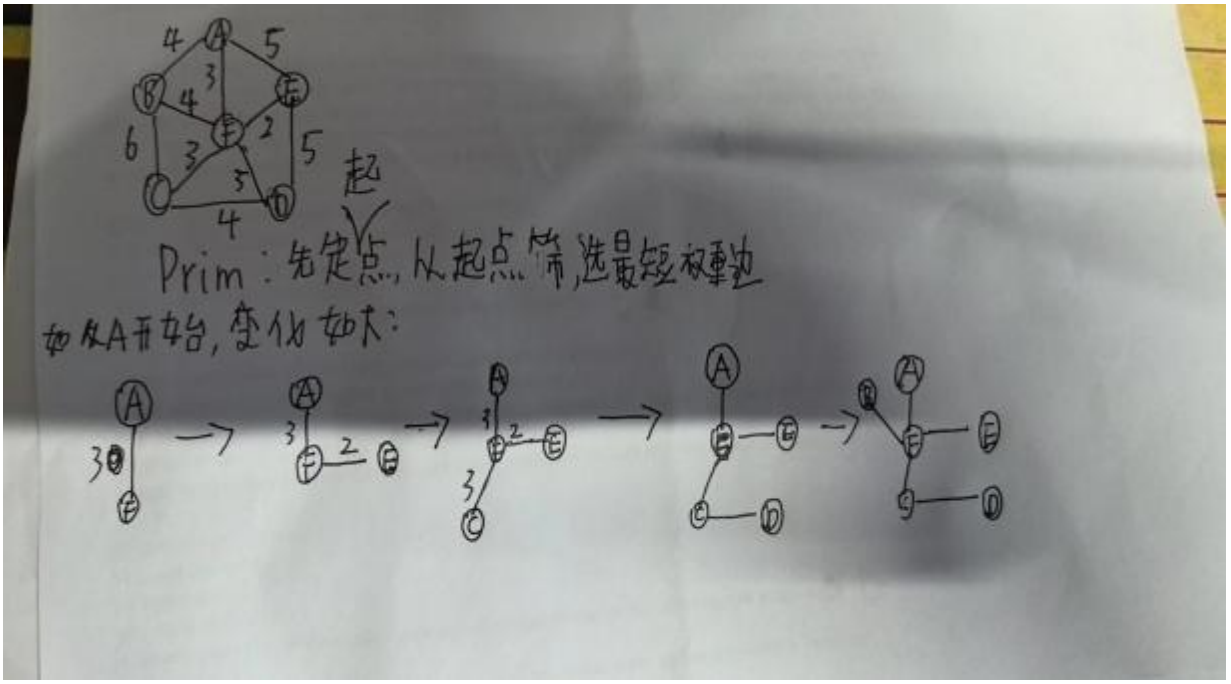
1、算法具体分类：普里姆算法（prim）、克鲁斯卡尔算法（kruskal）

2、输入类型：对每条边都有不同权重的无向图

3、输出数据类型：最小生成树

普里姆算法的详细内容和例子：

普里姆算法指的是从某点出发，根据距离点权重最低的边连接最终生成树的算法，他会从图中任意一点开始连接，并将连接后的点所相邻的边也纳入比较范围。具体情况如下图所示



伪代码则如下

pick a vertex V_0 in V

$V_T = \{V_0\}$

$E_T = \varnothing$

for i=1 to |V|-1 do

pick an edge e =(v*, u*) with minimum weight among all the edges (v, u) such that v is in V_T and u is in V- V_T

$V_T = V_T \cup \{v^*\}$

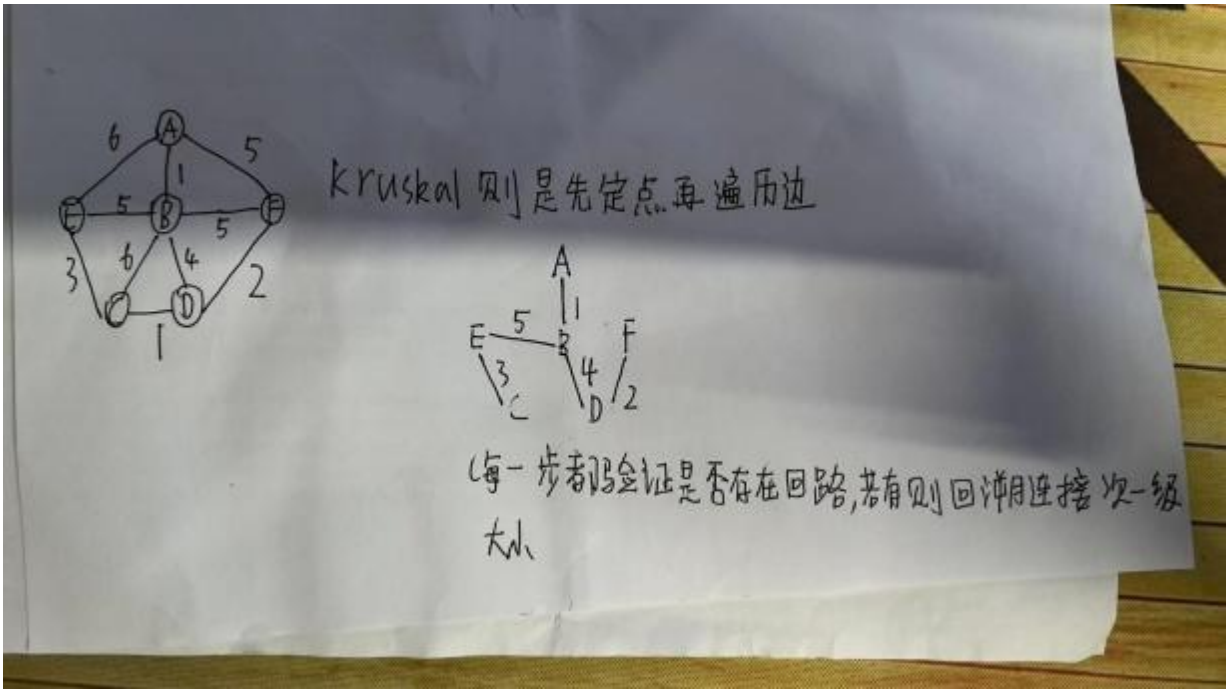
$E_T = E_T \cup \{e^*\}$

Return E_T

然后我们就可以根据伪代码来计算他的时间复杂度了，具体的来说他的时间复杂度取决于我们的图是如何储存的，我们首先让V代表节点数E代表边数，那么对应该算法我们可以知道他首先是要遍历一遍所有的点以此来找到对应权重最短的边，然后这个操作在每一次得到新的节点后都会重复一次，那么最终我们就可以得到他是O（v^2）,另外一种情况下我们是使用邻接表和二叉堆来计算，但是我也没太搞清楚堆的概念，结果是O（Elogv）

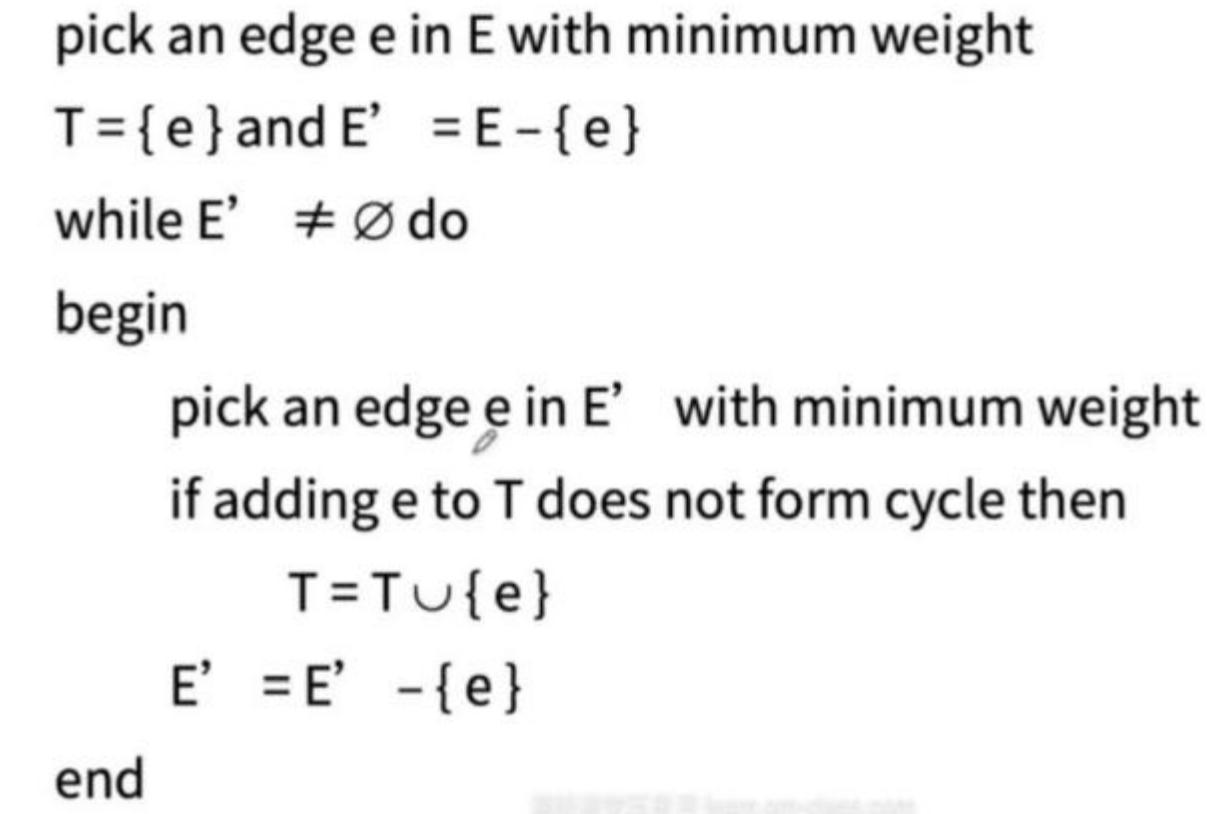
克鲁斯算法的详细内容和例子：

克鲁斯算法则是先固定出所有的点，然后再按照所有边中权重最低的开始连接并确保不生成回路（值得注意的是他不是生成再回溯而是直接避免生成）。同样我用一张手绘的图来表示内容



伪代码如下

生成树 ：



而kurskal的时间复杂度则是O（ElogE），因为对于这个算法来说我们首先是进行边的权值排序，然后就只用经过一次遍历，即对E使用排序的思想然后遍历连接，那么他就会先经过一个（ElogE）的时间复杂度然后一个O（n），那么就可以直接视为O（ElogE）

综上，我们可以大体把他们总结为下表

生成最小生成树的算法	普里姆算法	克鲁斯卡尔算法
算法思想	选择点	选择边
时间复杂度	$O(V^2)/O(Elog_2V)$	$O(Elog_2E)$
适用范围	稠密图	稀疏图

由该表我们可以看出在E>V^2的时候我们尽可能会选择prim算法，反之则是Kruskal算法。

定义：我们对一个无向图进行遍历过程中经过的点和边连通后可以看做为一个数（也就是从森林到树的过程）

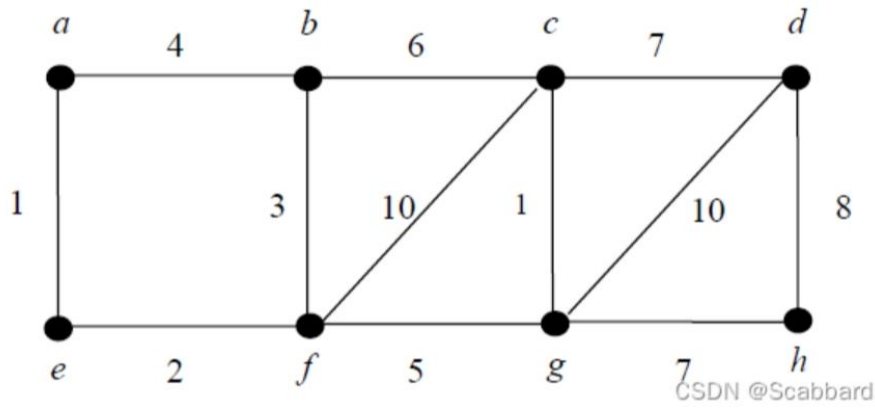
性质：生成树的权重是树中所有边权重之和，用上图为例子的话就是权重等于（1+3+5+2+4 = 15），同时，对应同一个图他的树的数量是不唯一的（这是为什么我们之前求得是最小生成树而不是生成树），树的顶点数量是和图的顶点数量相同的，同时生成树不存在回路一说。同时，对于最小生成树来说，边的数量始终是顶点数量-1

最小生成树：最小生成树就是我们在生成树的基础上要求他的权重最小，同时，最小生成树的数量也不是唯一的

附录：如何用表来体现这俩算法的计算过程，我直接用例子来表示

Question 3

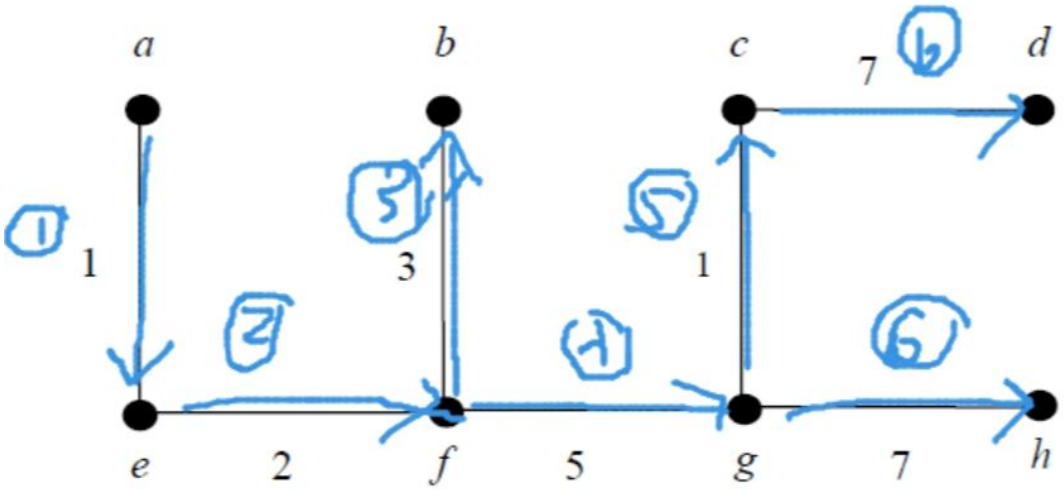
Consider the following graph G. The label of an edge is the cost of the edge.



CSDN @Scabbards_

1. Using *Prim's* algorithm, draw a *minimum spanning tree* (MST) of the graph. Also write down the change of the priority queue step by step and the order in which the vertices are selected. Is the MST drawn unique? (i.e., is it the one and only MST for the graph?) **[5 marks]**

Order Selected	a(0,-)	b(-,∞)	c(-,∞)	d(-,∞)	e(-,∞)	f(-,∞)	g(-,∞)	h(-,∞)
a(0,-)		b(a,4)	c(-,∞)	d(-,∞)	e(a,1)	f(-,∞)	g(-,∞)	h(-,∞)
e(a,1)		b(a,4)	c(-,∞)	d(-,∞)		f(-,2)	g(-,∞)	h(-,∞)
f(e,2)		b(f,3)	c(f,10)	d(-,∞)			g(f,5)	h(-,∞)
b(f,3)			c(b,6)	d(-,∞)			g(f,5)	h(-,∞)
g(f,5)			c(g,1)	d(g,10)				h(g,7)
c(g,1)				d(c,7)				h(g,7)
d(c,7)								h(g,7)
h(g,7)								



以上为prim算法的表的表示方法，如下为Kruskal算法的

2. Using *Kruskal's* algorithm, draw a *minimum spanning tree* (MST) of the graph G. Write down the order in which the edges are selected.
Is the MST drawn unique? (i.e., is it the one and only MST for the graph?)
(5 marks)

(a, e)	1/
(c, g)	1/
(e, f)	2/
(b, f)	3/
~~(a, b)~~	~~4~~
(f, g)	5/
~~(b, c)~~	~~6~~
(c, d)	7/
(g, h)	7/
~~(d, h)~~	~~8~~
~~(c, f)~~	~~10~~
~~(d, g)~~	~~10~~