algorithm01 代码思想

- 读取task集合tasks，worker集合workers
- 生成距离倒数$1/r_{ij}$的列表$Edges$,这里直接就拉成一维的了
- 从大到小，排序，而且只排这一次就够了，时间复杂度$mn \times log(mn)$
- 选取$Edges$里第一个，也就是$1/r_{ij}$最大的，加入列表$Pairs$(表示匹配好的对)
- 更新列表$Edges$：获取刚刚匹配上的$worker\ id$和$task\ id$，将$Edges$与这两个id相关的边删掉
- 重复第四步,直到$Edges$里没有元素

```
class E():
    def __init__(self,t_id,w_id,dis) -> None:
        self.dis=dis
        self.t_id=t_id
        self.w_id=w_id

class Algorithm01:
    def __init__(self) -> None:
        self.Edges:E=[]
        self.Pairs:E=[]

    def run(self,tasks:Task,workers:Worker):
        self.__getEdges(tasks,workers)
        self.Edges.sort(key=lambda x:x.dis,reverse=False) # 从大到小排列
        while(len(self.Edges!=0)):
            self.Pairs.append(self.Edges[0])
            self.__updateEdges()

    def __getEdges(self,tasks,workers):
        n=len(tasks)
        for i in range(n):
            for j in range(n):
                t=tasks[i]
                w=workers[j]
                e=E(t.id,w.id,dist(t,w))
                self.Edges.append(e)
    def __updateEdges(self):
        newEdges=[]
        e:E=self.Edges[0]
        t_id=e.t_id
        w_id=e.w_id
        for i in self.Edges:
            if i.t_id==t_id or i.w_id==w_id :
                continue
            newEdges.append(i)
        self.Edges=copy.deepcopy(newEdges)
```