Question 1.

1. 假设对称旅行商问题的邻接矩阵如图 1 所示，试用优先队列式分枝限界算法给出最短环游。画出状态空间树的搜索图，并说明搜索过程。

$$\begin{pmatrix} \infty & 20 & 30 & 10 & 11 \\ & \infty & 16 & 4 & 2 \\ & & \infty & 6 & 7 \\ & & & \infty & 12 \\ & & & & \infty \end{pmatrix}$$

图 1 邻接矩阵

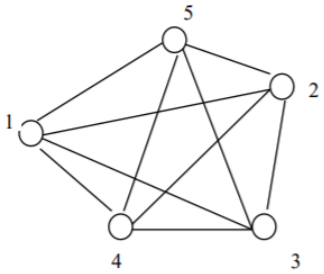
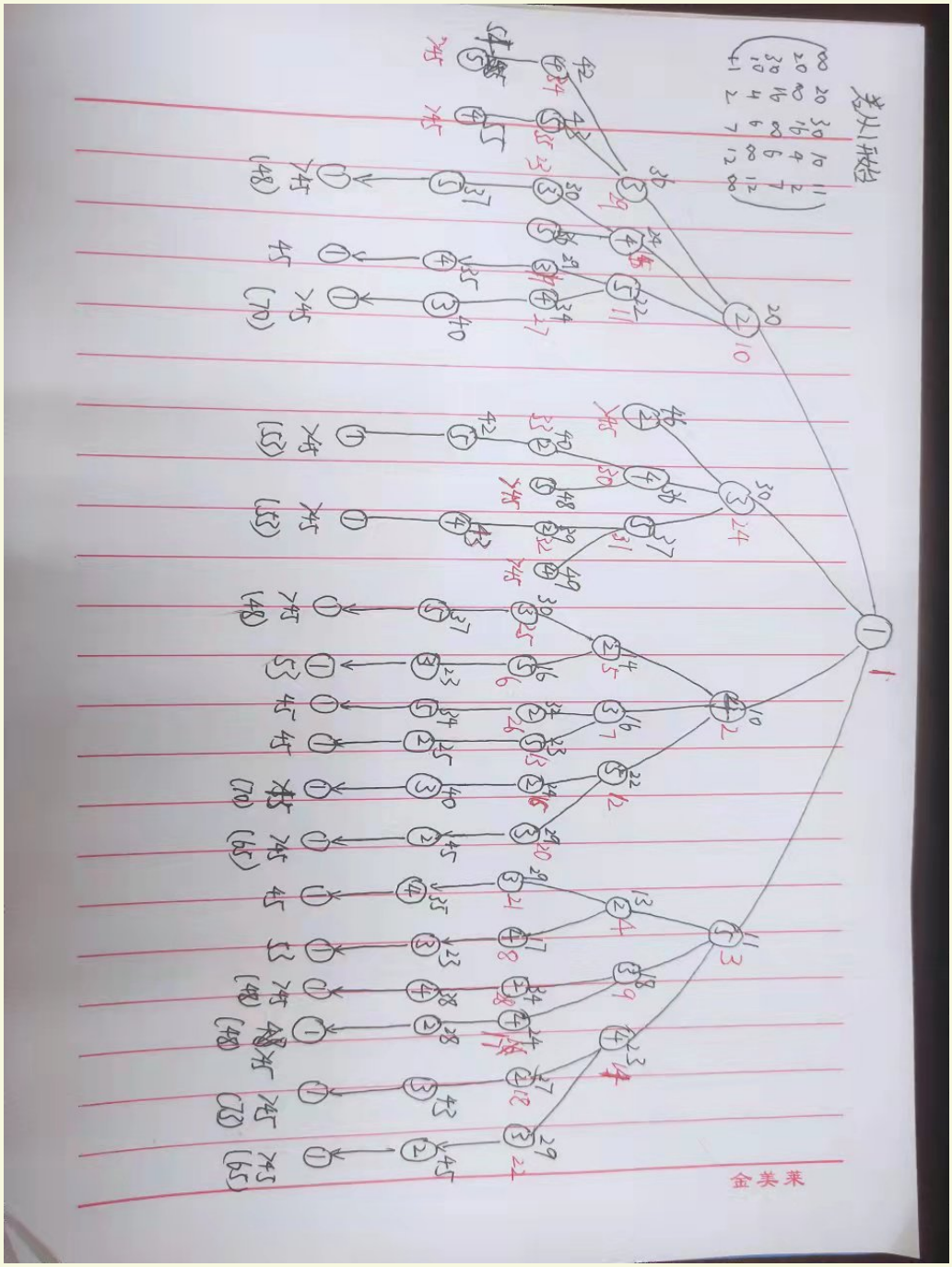


图 2 旅行商问题

My Answer 1. *1.* 搜索图如下所示：



图中红色数字为搜索序号，圆圈外的数字为已走路程，圆圈内数字为城市序号
由搜索图可知最短路长为 *45*，对应的解有：

$$1 \to 2 \to 5 \to 3 \to 4 \to 1$$

$$1 \to 4 \to 3 \to 2 \to 5 \to 1$$

$$1 \to 4 \to 3 \to 5 \to 2 \to 1$$

$$1 \to 5 \to 2 \to 3 \to 4 \to 1$$

Question 2.

2. 试写出 0/1 背包问题的优先队列式分枝限界算法程序，并找一个物品个数至少是 16 的例子检验程序的运行情况。

My Answer 2. *2.* 由题可定义数据结构

```
class Node:
    def __init__(self,Parent=None,Level=None,Tag=None,CC=None,CV=None,CUB=None):
        self.Parent=Parent#指向父节点的指针
        self.Level=Level#节点在解空间中的深度
        self.Tag=Tag#该节点深度对应的物品那还是不拿1/0
        self.CC=CC#该节点下背包的剩余空间
        self.CV=CV#该节点下物品的价值
        self.CUB=CUB#该节点下可能达到的物品价值上界Puv,该上界≥真实上界
```

根据 *0/1* 背包问题算法写出 *python* 程序，实现效果如下：

```
价值数组为： [11. 17. 29. 17. 19. 10. 19. 22. 17. 10.  6.  8.  9.  7.  1.  2.]
质量数组为： [ 1.  2.  4. 15. 18. 12. 23. 30. 25. 23. 14. 20. 23. 18.  7. 19.]
背包容量为： 254.0
背包内剩余空间为 0.0
背包内总价值为 204.0
应该拿的物品为： [16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

分析说明：当背包容量大于等于待装物体总质量时，全部装下是最优解

```
价值数组为： [30. 29.  4. 28. 18. 22. 26.  7. 17.  3. 11.  7.  6.  5.  3.  2.]
质量数组为： [ 2.  7.  2. 19. 14. 22. 29.  8. 22.  6. 23. 17. 21. 26. 23. 25.]
背包容量为： 177.33333333333334
背包内剩余空间为 6.333333333333343
背包内总价值为 202.0
应该拿的物品为： [12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

分析说明，当背包容量小于待装物体总质量时，可以验证上图中解为最优解

代码说明：当需要自定义价值矩阵与重量矩阵时，可以打开 7_2.*py* 文件，修改 P,W 的值；若需要改变背包容量，可以修改文件中 M 的值；修改完成后命令行或用 *ide* 运行 7_2.*py* 即可

算法 1 0/1 背包问题

输入: P, W, M//价值矩阵、重量矩阵、背包容量

输出: 最大价值 cv 以及对应的解 ANS

```
function MAXVALUE(P, W, M)
    初始化
    while True do
        记录当前节点 E 的剩余容量、价值、所在树的深度 i
        if E 为叶子节点 then
            if E 的价值大于等于全局最优价值 prev then
                更新全局最优价值 prev 并更新最优解为 E
            end if
        else
            if W then[i+1] 可被放入背包
                开启 E 的左子节点，放入活节点表
            end if
            if 不拿 W[i] 的可能上界大于最好上界 then
                开启 E 的右子节点，放入活节点表
                更新 prev← max{prev, pvl − ε}
            end if
        end if
        if 活节点表为空 then
            break
        end if
        E ← 活节点表中 CUB 值最大的节点
        if E 可能达到的上界比已知能达到的最好上界 prev 小 then
            break//说明活节点表所有的解都不可能超越最优解了
        end if
    end while
    打印最优解
end function
```

Question 3.

3. 最佳调度问题:假设有 n 个任务要由 k 个可并行工作的机器来完成，完成任务 i 需要的时间为 t_i。试设计一个分枝限界算法，找出完成这 n 个任务的最佳调度，使得完成全部任务的时间（从机器开始加工任务到最后停机的时间）最短。

My Answer 3. 由题定义数据结构:

```
class Node:
    def __init__(self,Parent=None,Level=None,Solve=None,Time=None):
        self.Parent=Parent#父节点
        self.Level=Level#所在任务编号
        self.Solve=Solve#所在任务给哪台机器
```

由如下算法编写 *python* 代码，实现效果如下:

```
共有 4 台机器
任务时间数组为 [5, 12, 6, 4, 9, 6, 3, 15]
最优解为: [[1 2 3 1 3 1 2 4]]
最短时间为: 15
```

代码说明：若需要更改任务时间数组，请打开 7_3.*py*，修改 *t* 数组即可；修改完毕后，命令行或 *ide* 运行 7_3.*py* 即可

算法 2 多机调度问题

输入: t, K//任务时间数组, 机器数

输出: 最短时间 *bestTime* 以及对应的解 *soluSet*

```
function MINTIME(t, K)
    初始化
    while True do
        记录节点 E 当前所在深度 L
        if L==N then
            if 全局最短时间 bestT> 当前节点所花费时间 then
                更新全局最优价值 bestT 并更新最优解为 E
            end if
        else
            if E.Time<=bestT then
                for i do=1:K
                    生成第 i 个子节点并放入活节点表
                end for
            end if
        end if
        if 活节点表为空 then
            break
        end if
        E ← 活节点表中耗时最短的节点
    end while
    打印最优解集
end function
```