**题目1. 设有如图1所示的博弈树，其中叶结点对应的数字是给定的估值，请利用α-β剪枝技术剪去不必要的分支，并给出剪纸过程。**

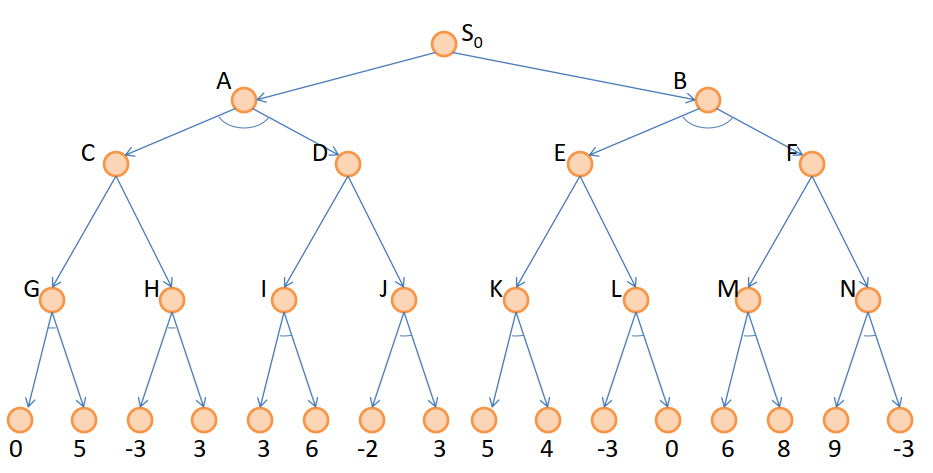
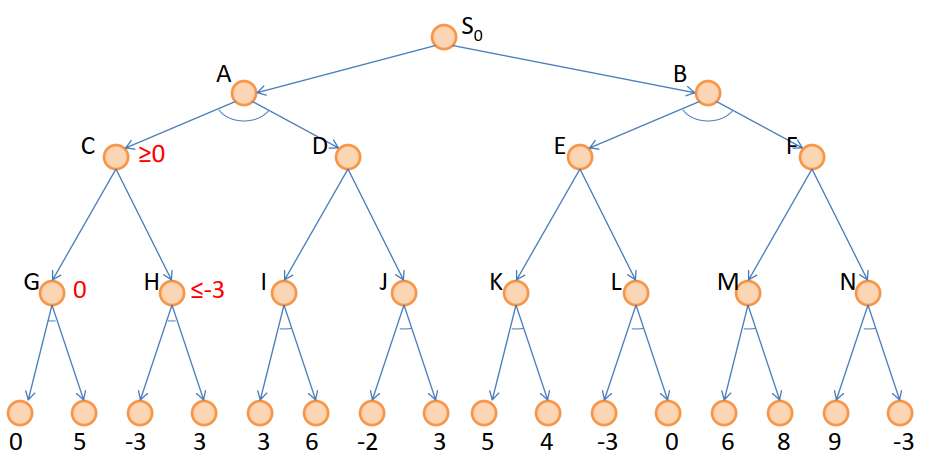
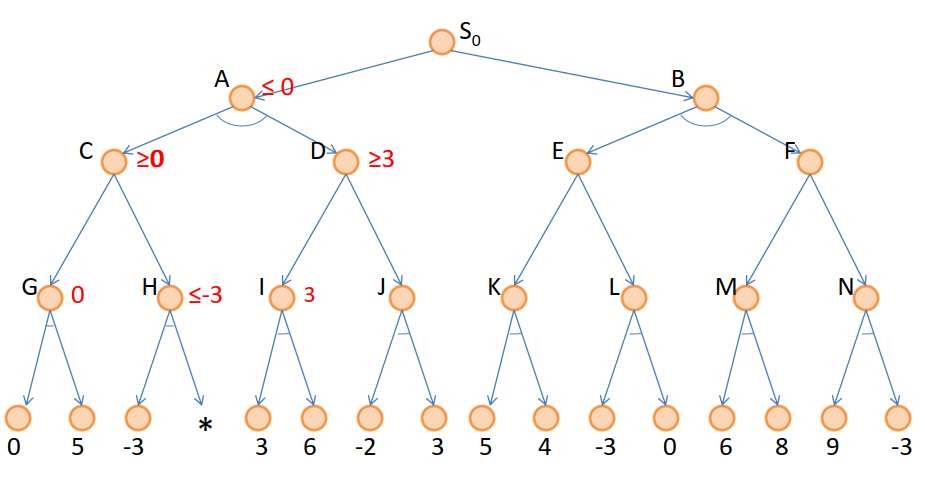


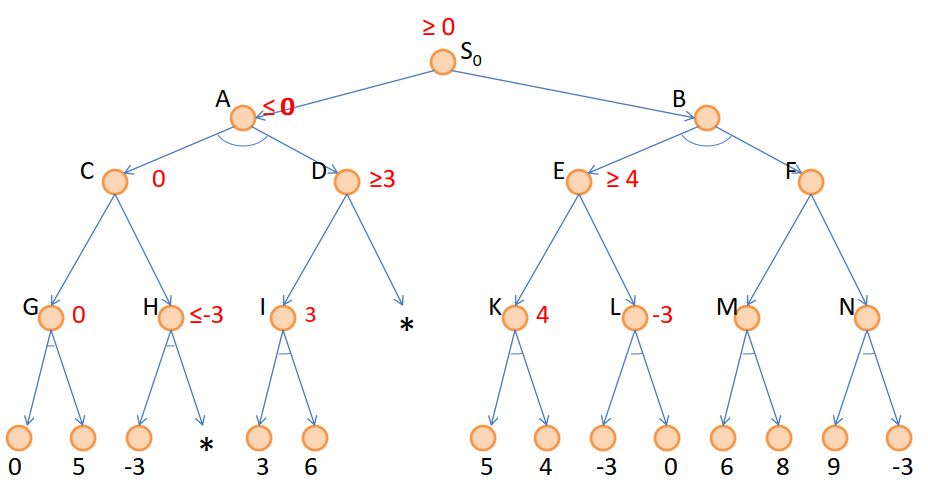
图1 博弈树

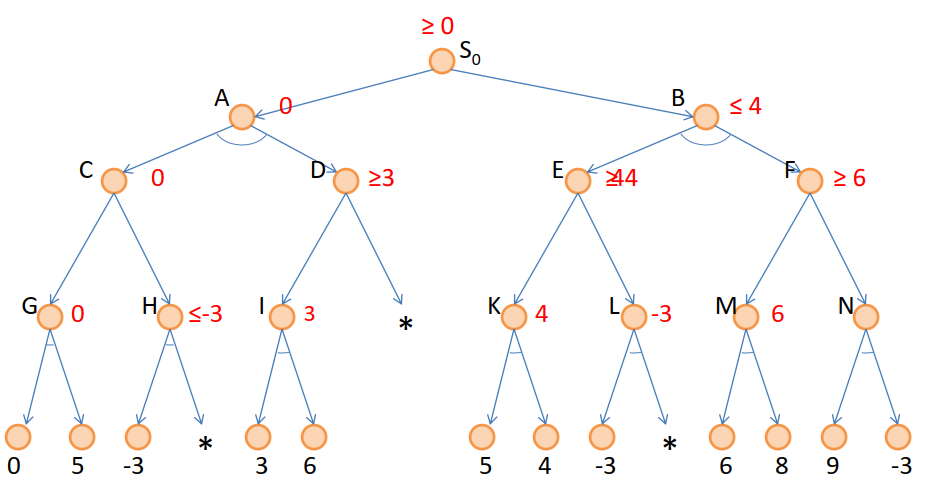
解答：

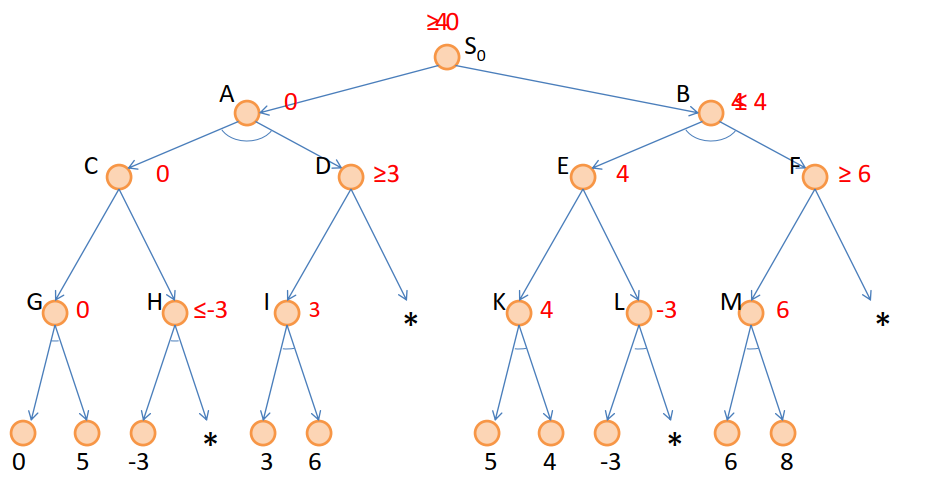
剪枝过程如下：











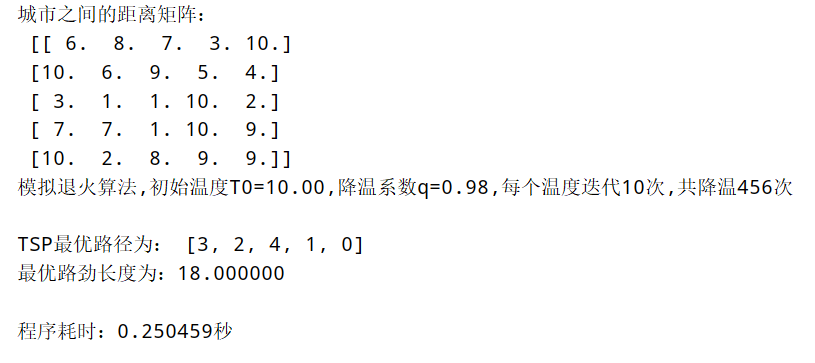
**题目2.** TSP问题（Traveling Salesman Problem，旅行商问题）问题描述如下：有若干个城市，任何两个城市之间的距离都是确定的，现要求一旅行商从某城市出发必须经过每一个城市且只在一个城市逗留一次，最后回到出发的城市。TSP问题求解的目标是，选择最短的线路保证其旅行的费用最少。

请使用模拟退火算法求解TSP最优路径和最优路径长度，给出Python实现代码。

要求：城市数量为5，使用random模块，随机生成每个城市的坐标，假设city a 到city b的距离和city b 到city a的距离一样。

解答：

|  |
| --- |
| **算法：模拟退火算法求解TSP问题**  **输出：TSP最优路径，最优路径长度** |
| '''  设有n个城市和距离矩阵D=[dij]，  其中dij表示城市i到城市j的距离，i，j=1，2 … n，  则问题是要找出遍访每个城市恰好一次的一条回路并使其路径长度为最短。  '''  from time import time  from copy import copy  from numpy import exp  import numpy as np  import random  T0 = 10.0 # 初始温度  T\_end = 0.001 # 最低温度  q = 0.98 # 退火系数  L = 10 # 每个温度时的迭代此时，即链长  N = 5 # 城市数量  city\_list = [i for i in range(N)] # 初始化一个解  city\_dis = np.floor(10 \* np.random.random((N, N)) + 1) # 城市之间的距离矩阵  # 计算路径长度  def path\_len(path\_list):  path = 0  for i in range(len(path\_list) - 1):  city1 = path\_list[i]  city2 = path\_list[i + 1]  dis = city\_dis[city1][city2]  path += dis  last\_city = path\_list[-1]  first\_city = path\_list[0]  dis = city\_dis[last\_city][first\_city]  path += dis  return path    # 采用随机交换位置的方式产生新解  def create\_new():  pos1 = random.randint(0, N - 1) # randint闭区间  pos2 = random.randint(0, N - 1)  temp = city\_list[pos1]  city\_list[pos1] = city\_list[pos2]  city\_list[pos2] = temp    if \_\_name\_\_ == '\_\_main\_\_':  t1 = time()  count = 0 # 记录降温次数  T = T0  city\_list\_copy = [] # 保存原始解    while T > T\_end: # 当温度低于结束温度时，退火结束  for i in range(L):  city\_list\_copy = copy(city\_list) # 复制数组  create\_new() # 产生新解  f1 = path\_len(city\_list\_copy) # 初始解目标函数值  f2 = path\_len(city\_list) # 新解目标函数值  df = f2 - f1  # Metropolis 准则  if df >= 0:  print("df:", df)  print("exp:", exp(-df / T))  if exp(-df / T) <= random.random(): # 保留原来解  city\_list = copy(city\_list\_copy)  T \*= q # 降温  count += 1  t2 = time()  print("城市之间的距离矩阵：\n", city\_dis)  print("模拟退火算法,初始温度T0=%.2f,降温系数q=%.2f,每个温度迭代%d次,共降温%d次\n" % (T0, q, L, count))  print("TSP最优路径为：", city\_list)  print("最优路劲长度为：%1f\n" % (path\_len(city\_list)))  print("程序耗时：%1f秒\n" % (t2 - t1)) |

**运行结果：**