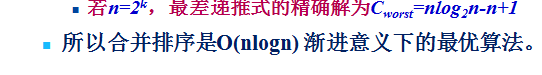
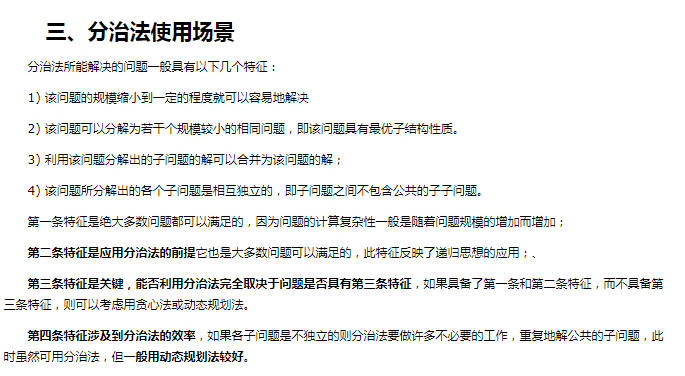
1. 算法就是一组有穷的规则。√
2. 概率算法中蒙特卡罗算法得到的解必是正确的。×
3. 程序和算法一样，都是某种程序设计语言的具体实现。×程序是算法用某种程序设计语言的具体实现
4. 合并排序算法是渐近最优算法。√



5. 递归定义必须是有确切含义是指必须一步比一步简单，最后是有终结的，决不能无限循环下去。√

6. 二分搜索方法在最坏的情况下用O(log n)时间完成搜索任务。√

7. 能否利用分治法完全取决于问题是否具有如下特征：利用该问题分解出的子问题的解可以合并为该问题的解。√



8. 分治法的基本思想是将一个规模较大的问题分解成若干个规模较小的子问题，这些子问题之间并不一定相互独立。×

9. 递归算法的效率往往很低，费时和费内存空间。√

10. 当一个问题具有最优子结构性质时只能用动态规划方法求解。×

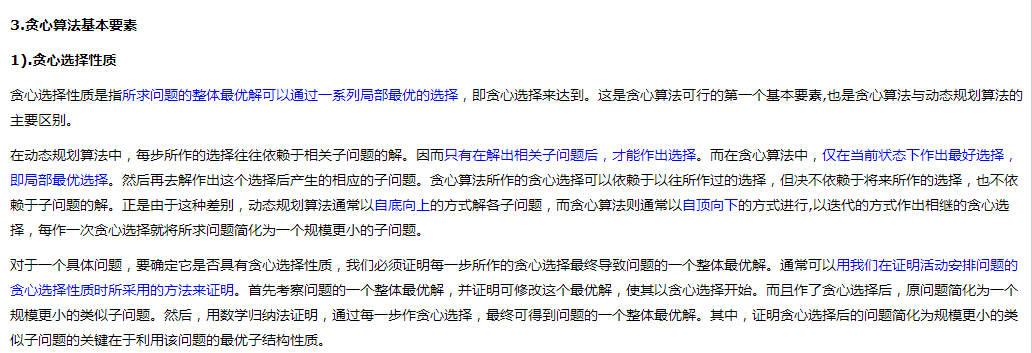
11. 如果一类活动过程一个阶段的决策确定以后，常影响到下一个阶段的决策，则称它为多阶段决策问题。√

12. 反复应用分治手段，不能使子问题与原问题类型一致而其规模却不断缩小。×

13. 裴波那契数列的定义：f(n)=f(n-1)+f(n-2),f(0)=1,f(1)=2，其数据的定义形式不是按递归定义。×

14. 0/1背包问题与背包问题这两类问题都可以用贪心算法求解。×

15. 证明贪心选择后的问题简化为规模更小的类似子问题的关键在于利用该问题的最优子结构性质。√



16. 子问题之间不包含公共的子问题，这个条件涉及到分治法的效率。√

17. 概率算法允许在执行过程中随机地选择下一个计算步骤。√

18. 二分搜索法的二分查找只适用于顺序存储结构。 √

19. 要想在电脑上扩大所处理问题的规模，有效的途径是降低算法的计算复杂度。√

20. 用回溯法解题一个显著特征是在搜索过程中动态产生问题的解空间。√

21. 从分治法的一般设计模式可以看出，用它设计出的程序一般是一个递归过程。因此，分治法的计算效率通常可以用递归方程来进行分析。√

22. 多阶段决策问题中，每一个阶段可能有若干个决策可供选择。√

23. 拉斯维加斯算法不会得到不正确的解，但有时找不到解。 √

24. 在通往边界条件的递归调用过程中，系统用堆栈保存的每次调用的中间结果是局部变量和返回地址值。√

25. 要想在电脑上扩大所处理问题的规模，有效的途径是提高算法的计算复杂度。×

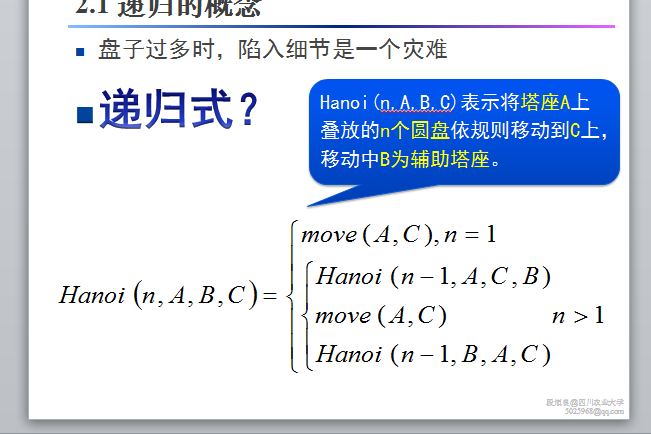
26. 程序必须满足算法具有数据输出的性质。√ 有零个或多个输入，至少1个或多个输出

27. 反复应用分治手段，可以使子问题与原问题类型一致而其规模却不断缩小。√

28. 一个算法产生一个或多个输出，它们是同输入有某种特定关系的量。√

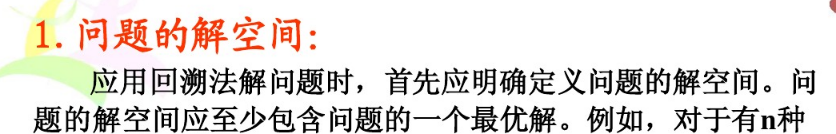
29. 最优子结构性质特征反映了递归思想的应用。√

30. 递归边界本身并不使用递归的定义。√



31. 用分治法求解一个问题，所需的时间是由子问题的个数、大小以及把这个问题分解为子问题所需的工作总量来确定的。√

32. 应用回溯法解问题时，首先应明确定义问题的解空间。问题的解空间应至少包含问题的一个（最优）解。 √



33. 好的约束函数能显著地减少所生成的结点数，但这样的约束函数往往计算量较大。因此，在选择约束函数时通常存在生成结点数与约束函数计算量之间的折衷。 √

34. 一个递归定义必须是有确切含义的，必须一步比一步简单，最后是有终结的，不能无限循环下去。√

35.最优子结构性质是应用分治法的前提。√



36.操作系统，它是一个在无限循环中执行的程序，因而不是一个算法。√

37.有些数据结构如二叉树等，由于其本身的递归特性、特别适合用递归的形式来描述。√

38.概率算法的一个基本特征是，对所求问题的同一个实例用同一个算法求解两次一定能得到完全相同的效果。 ×

39.问题可以分解为若干个规模较小的相同问题，即称该问题具有最优子结构性质。× 当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。

40.递推是从内边界条件出发，通过递推式达到边界条件。√

41.所有的递归函数都能找到对应的非递归定义。√

42.定义递归函数时可以没有初始值。×

43.动态规划算法的基本要素是最优子结构。√

44.最优子结构性质是指原问题的最优解包含其子问题的最优解。√

45.动态规划算法求解问题时，分解出来的子问题相互独立。×

46.满足贪心选择性质必满足最优子结构性质。×

47.回溯法中限界函数的目的是剪去得不到最优解的子树。√

48.分支限界法类似于回溯法，也是一种在问题的解空间树T上搜索问题解的算法，两者的搜索方式是相同的。×

49.任何递归算法都有递归出口。√

50.递归算法的执行效率比功能相同的非递归算法的执行效率高。×

51.递归算法不能转换成对应的非递归算法。×

52.数据元素是数据的最小单位。× 数据项 数据元素是基本单位

53.数据对象就是一组数据元素的集合。× 是性质相同的数据元素的集合。

54.任何数据结构都具备三个基本运算：插入、删除和查找。×

55.数据对象是由有限个类型相同的数据元素构成的。√

56.数据的逻辑结构与各数据元素在计算机中如何存储有关。× 物理结构

57.如果数据元素值发生改变，则数据的逻辑结构也随之改变。×

58.逻辑结构相同的数据，可以采用多种不同的存储方法。√

59.逻辑结构不相同的数据，必须采用不同的存储方法来存储。×

60.数据的逻辑结构是指数据元素的各数据项之间的逻辑关系。× 数据元素之间的逻辑关系

61.顺序存储方式只能用于存储线性结构。×

62.算法可以用不同的语言来描述，如果用C语言或Pascal语言等高级语言来描述，则算法就等同于程序。×

63. 数据的逻辑结构是指各数据元素之间的逻辑关系。√

64.数据结构、数据元素、数据项在计算机中的映像（或表示）分别称为存储结构、节点和数据域。√

65.数据的物理结构是指数据在计算机内的实际存储形式。√

66.分配给单链表的内存与地址必须是连续的。×

67.从长度为n的顺序表中删除任何一个元素，时间复杂度都是O(n)。× 最坏情况下

68.向顺序表中插入一个元素，平均要移动大约一半的元素。√

69.凡是为空的单链表都是不含任何节点的。×

70.如果单链表带有头节点，则插入操作永远不会改变头节点指针的值。√

71.在循环单链表中，任何一个节点的指针域都不可能为空。√

72.顺序存储方式的特点是存储密度大且插入、删除运算效率高。×

73.线性表的顺序存储结构优于链式存储结构。×

74.顺序存储结构属于静态结构而链式存储结构属于动态结构。√

75.由于顺序存储结构要求连续的存储区域，所以在存储管理上不够灵活。√

76.对于单链表来说，只有从头节点开始才能扫描表中全部节点。√

77．对于循环单链表来说，从表中任一节点出发都能扫描表中全部节点。√

78.双链表的特点是很容易找任一节点的前趋和后继。√

79.线性表有两种存储结构：一是顺序表，二是链表。√

80.如果有多个线性表同时共存，并且在处理过程中各表的长度会动态地发生变化，线性表的总数也会自动改变。在此情况下，应选用链式存储结构。√

81.若线性表的总数基本稳定且很少进行插入和删除操作，但要求以最快的速度存取线性表中的元素，那么应该选用顺序存储结构。√

82.对于单链表和双链表，均能从当前节点出发访问到任一节点。×

83.循环单链表和循环双链表从尾指针出发可以访问到链表中的任意节点。√

84.若频繁地对一个线性表进行插入和删除操作，该线性表宜采用链式存储结构。√

85.栈底元素是不能删除的元素。×

86.顺序栈中元素值的大小是有序的。×

87.栈顶元素和栈底元素有可能是同一个元素。√

88.若用s[1…m]表示顺序栈的存储空间，则对栈的进栈、出栈操作最多只能进行m次。×

89.栈是一种对进栈、出栈操作总次数作了限制的线性表。×

90.空栈没有栈顶指针。×

91.环形队列中有多少元素，可以根据队首指针和队尾指针的值来计算。√

92.无论是顺序队列，还是链式队列，插入、删除运算的时间复杂度都是O(1)。√

93.栈和队列都是插入和删除操作受限的线性表。√

94.栈和队列的存储方式既可以是顺序方式，也可以是链式方式。√

95.环形队列也存在空间溢出的问题。√

96.消除递归不一定需要使用栈。√ 也可以使用迭代

97.二分搜索算法是利用贪心法实现的算法。×

98.动态规划算法通常是以自底向上的方式求解最优解。√

99.贪心法不能解决0/1背包问题。√

100.深度优先不是分支限界法的搜索方式。√ 广度优先

101.二分搜索算法是利用分治策略实现的算法。√

102.背包问题不能使用贪心法解决。×

103.单源最短路径问题不能使用贪心法解决。×

104.时间复杂度低是衡量一个算法好坏的标准。√

105.归并排序不可以使用分治法求解。×

106.拉斯维加斯算法有时找不到问题的解。√

107.舍伍德算法有时候找不到问题的解。×舍伍德算法总能求得问题的一个解，且求得的解一定是正确的。

108.NP问题都是不可能解决的问题。×

109.P类问题包含在NP类问题中。√

110.NP类问题包含在P类问题中。×

111.NP完全问题是P类问题的子集。×

112.蒙特卡罗算法是概率算法的一种。√

113.蒙特卡罗算法是贪心算法的一种。× 概率算法

114.蒙特卡罗算法是回溯算法的一种。×

115.动态规划算法不是随机化算法。√

116.最优子结构性质是贪心算法与动态规划算法的共同点。√

117.矩阵连乘问题的算法可由动态规划算法来设计实现。√

118.Strassen 矩阵乘法是利用分治策略实现的算法。√

119.Strassen 矩阵乘法是利用贪心法实现的算法。×

120.贪心选择性质是贪心算法的基本要素。√

121.以深度优先方式系统搜索问题解的算法称为回溯算法。√

122.算法分析的两个主要方面是时间复杂度和空间复杂度分析。√

123.实现最大子段和利用的算法是动态规划法。√

124.实现最大子段和利用的算法是贪心法。×

125.实现最大子段和利用的算法是回溯法。×

126.广度优先是分支限界算法的一种搜索方式。√

127.广度优先是回溯算法的一种搜索方式。×

128.广度优先是贪心算法的一种搜索方式。×

129.舍伍德算法是概率算法的一种。√

129.舍伍德算法是贪心算法的一种。×

130.舍伍德算法是回溯算法的一种。×

132.实现最长公共子序列利用的算法是动态规划法。√

133.计算机算法指的是解决问题的方法和过程。√

134.根据排序元素所在位置的不同，排序分内排序和外排序。√

135.根据排序元素所在位置的不同，排序分首排序和尾排序。×

136.算法必须具备输入、输出和有穷性、确定性和可行性等5个特性。√

137.算法必须具备输入、输出和易读性、稳定性和安全性等 5个特性。×

138.与分治法不同的是，适合于用动态规划求解的问题经分解得到的子问题往往不是相互独立的。√

139.与分治法不同的是，适合于用动态规划求解的问题往往是相互独立的。×

140．二分搜索算法的基本思想是将n个元素分成个数大致相同的两半，取a[n/2]与x进行比较：如果x<a[n/2]，则只要在数组a的左半部继续搜索x。√

141.二分搜索算法的基本思想是将n个元素分成个数大致相同的两半，取a[n/2]与x进行比较：如果x>a[n/2]，则只要在数组a的左半部继续搜索x。×

142.算法必须具备输入、输出和可执行性、可移植性和可扩充性等5个特性。×

143.适用动态规划的问题必须满足最优化原理和无后效性。√

无后效性是指如果在某个阶段上过程的状态已知，则从此阶段以后过程的发展变化仅与此阶段的状态有关，而与过程在此阶段以前的阶段所经历过的状态无关。

144.适用动态规划的问题必须满足最优化原理和后效性。×

145.二分查找可适用于链式存储结构。×

146.二分查找只适用于链式存储结构。×

147.应用分治法的两个前提是问题的可分性和解的可归并性。√

148.应用分治法的两个前提是问题的可分性和解的复杂性。×

149.对于n个元素的排序问题。n＝2时只要作1次比较即可排好序。√

150.对于n个元素的排序问题。n＝2时要作2次比较即可排好序。×

151.分治法所能解决的问题应具有的最关键特征是利用该问题分解出的子问题的解可以合并为该问题的解。√

152.分治法所能解决的问题应具有的最关键特征是该问题的规模缩小到一定的程度就可以容易地解决。×

153.直接或间接的调用自身的算法称为递归算法。√

154．直接或间接的调用自身的算法称为动态规划算法。×

155.当上下限表示相等时我们使用Θ表示法来描述算法代价。√

156.当上下限表示相等时我们使用大O表示法来描述算法代价。×

157.递归通常用栈来实现。√ 会消耗更多的栈空间

158.递归通常用队列来实现。×

159.分治法的设计思想是将一个难以直接解决的大问题分割成规模较小的子问题分别解决子问题最后将子问题的解组合起来形成原问题的解。这要求原问题和子问题的问题规模不同，问题性质相同。√

160.0/1背包问题不能用贪心算法求解。√

161.可以由多项式时间算法求解的问题是易处理的。√

时间复杂度是O(p(n))的算法称为多项式时间算法，这里p(n)是关于n的多项式。不能够这样限制时间复杂度的算法被称为指数时间算法。

162.可以由多项式时间算法求解的问题是难处理的。×

163.需要超过多项式时间算法求解的问题是不能处理的。×

164.递归通常用数组来实现。×

165.哈密尔顿回路问题是典型的NP完全问题。√

166.排序问题是典型的NP完全问题。×

167.算法分析需要对算法需要多少计算时间和存储空间作定量分析。√

168.用数量级形式表示算法的执行时间称为算法的时间复杂度。√

169.用数量级形式表示算法的执行时间称为算法的空间复杂度。×

170.最坏情况下，顺序查找的时间复杂度为O(n)。√

171.最坏情况下，折半查找的时间复杂度为O(log2n)。√

172.合并排序的基本运算是把两个或多个有序序列合并成一个有序序列。√

173.最优子结构是动态规划算法的基本要素之一。√

174.快速排序算法是基于分治策略的一种排序算法。√ 分区操作

175.快速排序算法是基于回溯的一种排序算法。×

176.快速排序算法是基于贪心法的一种排序算法。×

177.贪心法通常以自顶向下的方式求解最优解。√

178.分治法通常以自顶向下的方式求解最优解。×

179.回溯法通常以自顶向下的方式求解最优解。×

180.不断回头寻找目标的方法称为回溯法。√

181.不断回头寻找目标的方法称为概率算法。×

182.不断回头寻找目标的方法称为贪心法。×

183.拉斯维加斯算法找到的解一定是正确的。√

184.拉斯维加斯算法找到的解正确与否不确定。×

185.记号在算法复杂性的表示法中表示紧致界。√

186.记号在算法复杂性的表示法中表示上界。×

187.记号在算法复杂性的表示法中表示下界。×

188.一个算法是对特定问题求解的一种描述，它是指令的有限序列。√

189.一个递归算法必须包括终止条件和递归部分。√

190.栈和队列的共同点是只允许在端点处插入和删除元素。√

191.排序趟数与原始序列有关的排序方法是冒泡排序法。√

192.栈和队列的共同点都是先进先出。×

193.栈和队列的共同点都是先进后出。×

194.排序趟数与原始序列有关的排序方法是选择排序法。× 冒泡排序

195.在算法的三种情况下的复杂性中，可操作性最好且最有实际价值的是最坏情况下的时间复杂度。√

196.在算法的三种情况下的复杂性中，可操作性最好且最有实际价值的是最好情况下的时间复杂度。×

197.若一个算法的时间复杂度用T(n)表示，其中n的含义是问题规模。√

198.合并排序法的基本思想是：将待排序元素分成大小大致相同的2个子集合，分别对每个子集合进行排序，最终将排好序的子集合合并成为所要求的排好序的集合。√

199.算法是指解题方案的准确而完整的描述，是一系列解决问题的清晰指令。√

200.排序趟数与原始序列有关的排序方法是插入排序法。×

201.排序趟数与原始序列无关的排序方法是选择排序法。√

202.排序趟数与原始序列无关的排序方法是冒泡排序法。×

203.排序趟数与原始序列有关的排序方法是快速排序法。×

204.算法分析中，记号O表示渐进下界。×

205算法分析中，记号O表示渐近上界。√

206.算法分析中，记号Ω表示渐近下界。√

207.算法分析中，记号Ω表示渐近上界。×

208.分支限界法在问题的解空间树中，按广度优先策略，从根节点出发搜索解空间树。√

209.分支限界法在问题的解空间树中，按深度优先策略，从根节点出发搜索解空间树。×

210.常见的两种分支限界法为队列式（FIFO）分支限界法与优先队列式分支限界法。√

211.蒙特卡罗算法用于求解问题的准确解，且该解一定是正确的。×

212.对于蒙特卡罗算法，求得正确解的概率依赖于算法的计算时间。√

213.多次执行蒙特卡罗算法，可以提高获得正确解的概率。√

214.对于蒙特卡罗算法，无法有效判定所得到的解是否肯定正确。√

215. 对于拉斯维加斯算法，找到正确解的概率随算法计算时间的增加而提高。√

216.用同一拉斯维加斯算法对同一问题求解多次，对求解失败的概率没有影响。×

217.对于舍伍德算法，总能求得问题的一个解。√

218.对于舍伍德算法，不一定能求得问题的解。×

219.对于舍伍德算法，所求得的解总是正确的。√

220.将确定性算法引入随机性改造成舍伍德算法，可消除或减少问题对于好坏实例间的差别。√

221.对于数值概率算法，常用于数值问题的求解，得到的往往是近似解。√

222.对于数值概率算法，解的精度随计算时间的增加而提高。√

223.对于数值概率算法，解的精度和计算时间之间没有关系。×

224.对于数值概率算法，在很多情况下，计算出问题的精确解是不可能或没必要。√

225.操作，控制结构，数据结构属于算法三要素。√

226.有穷性不属于算法设计的质量指标。√

227.有穷性属于算法设计的质量指标。×

228.正确性，可读性，健壮性属于算法设计的质量指标。√

229.高级语言更接近算法语言，易学，易掌握。√

230.高级语言为程序员提供了结构化程序设计的环境和工具。√

231.高级语言依赖于机器语言。×

232.高级语言不依赖于机器语言。√

233.折半查找、合并排序、二叉树遍历等算法中均采用了分治策略。√

234.折半查找、合并排序、二叉树遍历等算法中均采用了回溯策略。×

235.折半查找、合并排序、二叉树遍历等算法中均采用了动态规划策略。×

236.折半查找、合并排序、二叉树遍历等算法中均采用了贪心选择策略。×

237.递归算法设计的关键在于找出递归关系和递归终止条件。√

238.递归算法设计的关键在于找出递归关系和递归初始值。×

239.无后效性是问题能用贪婪算法或动态规划算法求解的前提。√

240.问题规模不能太大是问题能用动态规划算法求解的前提。×

241.算法分析的目的是分析算法效率以求改进。√

242.回溯算法是尝试搜索算法中最为基本的一种算法，其采用了一种不能走就掉头的思想作为其控制结构。√

243.算法分析的目的是找出数据结构的合理性。×

244.算法分析的目的是研究输入输出的关系。×

245.算法分析的目的是分析算法的易懂性和文档性。×

246.一个算法必须保证执行有限步之后结束，这是算法的有穷性。√

247.一个算法必须保证执行有限步之后结束，这是算法的确定性。×

248.一个算法必须保证执行有限步之后结束，这是算法的可行性。×

249.通常，最适合描述算法的语言是自然语言。×

250.通常，最适合描述算法的语言是数学公式。×

251.通常，最适合描述算法的语言是计算机程序设计语言。×

252.通常，最适合描述算法的语言是介于自然语言和程序设计语言之间的伪语言。√

253.对于反复多次使用的程序，应尽量选用节约空间的算法。×

254.对于反复多次使用的程序，应尽量选用节约时间的算法。√

255.对于反复多次使用的程序，应尽量选用简明易懂的算法。×

256.对于反复多次使用的程序，应尽量选用容易调试的算法。×

257.评价一个算法时间性能的主要指标是算法的时间复杂度。√

258.评价一个算法时间性能的主要指标是算法易于调试。×

259.评价一个算法时间性能的主要指标是算法易于理解。×

260..评价一个算法时间性能的主要指标是算法的稳定性和正确性。×

261.对于算法的时间复杂度来说，可操作性最好、最有实用价值的是最坏情况下的时间复杂度。√

262.对于算法的时间复杂度来说，可操作性最好、最有实用价值的是最好情况下的时间复杂度。×

263.对于算法的时间复杂度来说，可操作性最好、最有实用价值的是平均时间复杂度。×

264.贪心算法不是对所有问题都能得到整体最优解。√

265.贪心算法对所有问题都能得到整体最优解。×

266.能否利用分治法完全取决于该问题的规模缩小到一定程度就可以容易地解决。×

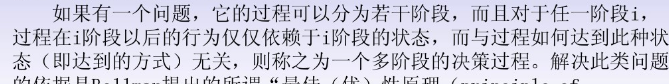
267.能否利用分治法完全取决于该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质。×

268.能否利用分治法完全取决于该问题分解出的各个子问题是相互独立的，即子问题之间不包含公共的子问题。×

269.如果有一个问题，它的过程可以分为若干阶段，而且对于任一阶i，过程在i阶段以后的行为仅仅依赖于i阶段的状态，而与过程如何达到此种状态（即达到的方式）无关，则称之为一个多阶段的决策过程。√

270.如果有一个问题，它的过程可以分为若干阶段，而且对于任一阶段i，过程在i阶段以后的行为仅仅依赖于i阶段的状态，而与过程如何达到此种状态（即达到的方式）无关，则称之为状态转换。×

271.如果有一个问题，它的过程可以分为若干阶段，而且对于任一阶段i，过程在i阶段以后的行为仅仅依赖于i阶段的状态，而与过程如何达到此种状态（即达到的方式）无关，则称之为最佳性原理。×



272.当需要找出它的解集或者要求回答什么解是满足某些条件的最佳解时，往往要使用回溯法。√

273.当需要找出它的解集或者要求回答什么解是满足某些条件的最佳解时，往往要使用分治法。×

274.当需要找出它的解集或者要求回答什么解是满足某些条件的最佳解时，往往要使用贪心法。×

275.当需要找出它的解集或者要求回答什么解是满足某些条件的最佳解时，往往要使用动态规划法。×

276.将问题分支为子问题，采用广度优先产生状态空间树的结点，并使用剪枝函数对这些子问题限界而求解问题的方法称为分支限界法。√

277.分支限界法只能应用于解决最优化问题。√

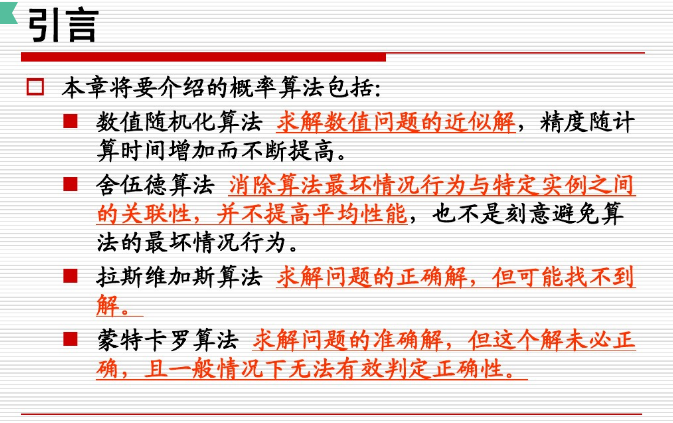
278.分支限界法只能应用于解决非最优化问题。×

279. 概率算法的一个基本特征是用同一概率算法求解问题的同一实例两次，得到的结果可能完全不同。√

280.回溯算法的一个基本特征是用同一概率算法求解问题的同一实例两次，得到的结果可能完全不同。×

281.贪心算法的一个基本特征是用同一概率算法求解问题的同一实例两次，得到的结果可能完全不同。×

282.近似算法的一个基本特征是用同一概率算法求解问题的同一实例两次，得到的结果可能完全不同。×



283.使用蒙特卡罗算法求解问题，通常是求解问题的准确解。√

284.使用数值概率算法求解问题，通常是求解问题的准确解。×

285.程序是算法用某种程序设计语言的具体实现，它可以不满足有穷性。√

286.程序是算法用某种程序设计语言的具体实现，它可以不满足确定性。×

287.程序是算法用某种程序设计语言的具体实现，它可以不满足可行性。×

288.程序是算法用某种程序设计语言的具体实现，它可以不满足有输入，输出。×

289.算法一般分为两类：精确算法和近似算法。√

290.算法一般分为两类：精确算法和概率算法。×

291.算法一般分为两类：精确算法和数值算法。×

292.算法一般分为两类，精确算法和启发式算法。√

293.分治法将一个难以直接解决的大问题，分割成一些规模较小的类型相同的问题，这些子问题相互独立，以便各个击破，分而治之。√

294.对于分治法，如果原问题可以分割成m个子问题，并且这些子问题都可解，然后求解这些子问题，那么就可以用这些子问题的解求出原问题的解。√

295.对于分治法，如果子问题还比较复杂而不能直接求解，还可以继续细分，直到子问题足够小，能够直接求解为止。√

296.对于分治法，子问题只能细分一次，不能继续细分。×

297.对于已按升序排好序的元素，采用二分检索，将待查找的数据与升序数组中的中间元素进行比较，若二者相等则表示找到。√

298.对于已按升序排好序的元素，采用二分检索，若待查找的数据小于中间元素的值，则下次在数组的前半部分中继续找。√

299.对于已按升序排好序的元素，采用二分检索，若待查找的数据大于中间元素的值，则下次在数组的后半部分中继续找。√

300.对于已按升序排好序的元素，采用二分检索，若待查找的数据大于中间元素的值，则下次在数组的前半部分中继续找。×

301.所谓贪心法的最优度量标准，是指可以根据该度量标准，实行多步地决策进行求解，虽然在该度量意义下所做的这些选择都是局部最优的，最终得到的解却不一定是全局最优的。√

302.当问题的规模递增时，将复杂度的极限称为算法的渐进复杂度。√

303.当问题的规模递增时，将复杂度的极限称为算法的最坏复杂度。×

304.当问题的规模递增时，将复杂度的极限称为算法的平均复杂度。×

305.问题的规模递增时，将复杂度的极限称为算法的最优复杂度。×

306.P中的一切问题都是多项式时间可解的。√

307.P中的一切问题都是常数阶时间可解的。×

308.近似算法就是指在解决最优化问题中，最后得到的结果能保证在一定误差之类的算法。√

309.概率算法就是指在解决最优化问题中，最后得到的结果能保证在一定误差之类的算法。×

310.并行算法就是指在解决最优化问题中，最后得到的结果能保证在一定误差之类的算法。×

311.针对许多不能在多项式时间内求解的NP完全问题，提出求解接近精确解的相似解来代替最优解。√

312.通常，对那些无法在多项式时间内求解的NP完全问题采用近似算法。√

313.通常，对那些无法在多项式时间内求解的NP完全问题采用概率算法。×

314.通常，对那些无法在多项式时间内求解的NP完全问题采用递归算法。×

315.0/1背包问题装入的物品是不可拆分的。√

316.背包问题装入的物品是可拆分的。√

317.0/1背包问题装入的物品是可拆分的。×

318.0/1背包问题可用动态规划的思想求解。√

319.算法由操作，控制结构，数据结构三要素组成。√

320.算法由程序，控制结构，数据结构三要素组成。×

321.算法由代码，控制结构，数据结构三要素组成。×

322.算法由问题规模，控制结构，数据结构三要素组成。×

323.算法是指解决问题的方法或过程。√

324.回溯算法的效率不依赖于问题的规模。×

325.回溯算法的效率不依赖于可行解的个数。×

326.回溯算法的效率不依赖于最优解的个数。×

327.回溯算法的效率不依赖于输入的数据。√

328.一个算法至少有一个输入和一个输出。×

329.算法的每一个步骤必须确切的定义。√

330.一个算法在执行有穷步后必须结束。√

331.算法中有待执行的运算和操作必须是相当基本的。√

332.在使用流程图描述算法时，表示变量的计算与赋值应使用的符号框为矩形框。√

333.在使用流程图描述算法时，表示变量的计算与赋值应使用的符号框为菱形框。×

334.在使用流程图描述算法时，表示变量的计算与赋值应使用的符号框为平行四边形框。×

335.在使用流程图描述算法时，表示变量的计算与赋值应使用的符号框为椭圆形框。×

336.算法复杂性的高低体现在运行该算法所需要的计算机资源上。对于计算机资源，最重要的是时间和空间资源。√

337.程序可以不满足有限性特征。√

338.程序可以不满足确定性特征。×

339.某个问题的最优解包含着其子问题的最优解。这种性质称为最优子结构性质。√

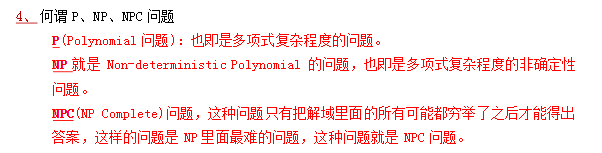
340.某个问题的最优解包含着其子问题的最优解。这种性质称为贪心选择性质。×

341.某个问题的最优解包含着其子问题的最优解。这种性质称为递归性质。×

342.某个问题的最优解包含着其子问题的最优解。这种性质称为动态规划性质。×

343.P问题，也即是多项式复杂程度的问题。√

344.NP就是Non-deterministic Polynomial的问题，也即是多项式复杂程度的非确定性问题。√



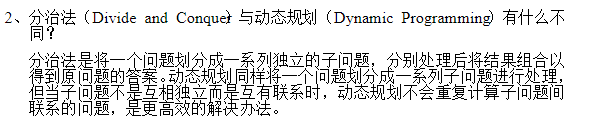
345.装载问题不适合用回溯法。×

346.0/1背包问题不适合用回溯法。×

347.图的m着色问题不适合用回溯法。×

348.分治法是将一个问题划分成一系列独立的子问题，分别处理后将结果组合以得到原问题的。

349.动态规划的效率比分治法低。×



350.当子问题间不是互相独立而是互有联系时，动态规划不会重复计算子问题间联系的问题。√

351.回溯法解旅行售货员问题时的解空间树是排列树。√

352.回溯法解旅行售货员问题时的解空间树是子集树。×

353.回溯法解0/1背包问题的解空间树是子集树。√

354.回溯法解0/1背包问题的解空间树是排列树。×

355.直接或间接地调用自身的算法称为递归算法，用函数自身给出定义的函数称为递归函数。√

356.递归方程和约束函数（递归终止条件）是递归函数的两个要素。√

357.在快速排序，插入排序和合并排序算法中，合并排序算法不是分治算法。×

358.在快速排序，插入排序和合并排序算法中，插入排序算法不是分治算法。√

359.在快速排序，插入排序和合并排序算法中，快速排序算法不是分治算法。×

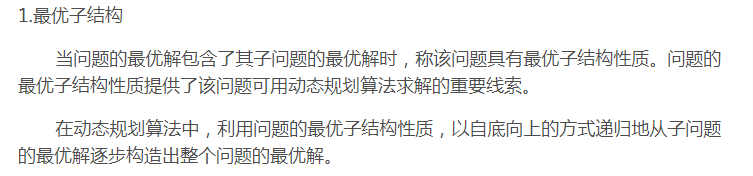
360.在动态规划算法中存储子问题的解是为了避免重复求解子问题。√

361.最优子结构是问题能用动态规划求解的前提。√

362.最优子结构不是问题能用动态规划求解的前提。×

363.最优子结构不是动态规划算法的基本要素。×

364.最优子结构性质是指原问题的最优解不包含其子问题的最优解。×



365.动态规划算法求解时，分解出来的子问题不是相互独立的。√

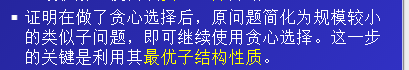
366.动态规划算法求解时，分解出来的子问题是相互独立的。×

367.矩阵连乘问题的算法不能由动态规划算法设计实现。×

368.动态规划算法的基本要素不包括最优子结构。×

369.最优化原理是动态规划的基础，任何问题，如果失去了最优化原理的支持，就不可能用动态规划方法计算。√

370.满足贪心选择性质不一定满足最优子结构性质。√



371.回溯法与分支限界法搜索方式不同，回溯法按深度优先搜索子空间。√

372.回溯法与分支限界法搜索方式不同，回溯法按广度优先或最小耗费优先搜索子空间。× 分支界限法

373. 分支限界法与回溯法搜索方式不同，分支限界法按深度优先搜索子空间。×

374.分支限界法与回溯法搜索方式不同，分支限界法按广度优先或最小耗费优先搜索子空间。√

375. 回溯法中限界函数的目的是剪去得不到次优解的子树。√

376.分支限界法类似于回溯法，也是一种在问题的解空间树T上搜索问题解的算法，两者的搜索方式是相同的。×

377.按照活结点表的组织方式的不同，分支限界法包括队列式分支限界算法和优先队列式分支限界算法两种形式。√

378.队列分支限界算法中，通常用队列来实现，体现先进先出的原则。√

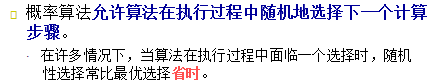
379.队列分支限界算法中，通常用队列来实现，体现后进先出的原则。×

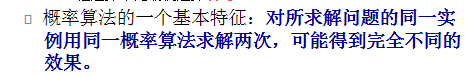
380.优先队列式分支限界算法，通常采用堆实现。√

381.回溯法与分支限界算法求解问题时，需要构造对该问题的解空间结构，其解空间结构通常有3种形式，分别是子集树，排列树和满m叉树。√

382.概率算法允许算法在执行的过程中随机选择下一个计算步骤。√

383.概率算法有一个基本特征，即是对所求解问题的同一实例用同一概率算法求解两次可能得到完全不同的效果。√





384.对所求解问题的同一实例用同一概率算法求解两次，则两次会得到完全相同的效果×

385.概率算法可分为数值概率算法，蒙特卡罗算法，拉斯维加斯算法，舍伍德算法4类。√

386.人们将存在多项式时间算法的问题称为易解问题。√

387.人们将需要在指数时间内解决的问题称为难解问题。√

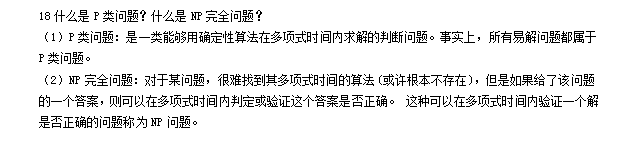
388.旅行商问题是NP完全问题。√

389.哈密尔顿回路问题是NP完全问题。√

390.排序问题是NP完全问题。×

391.对于某问题，很难找到其多项式时间的算法（或许根本不存在），但是，如果给了该问题的一个

392.P类问题：是一类能够用确定性算法在多项式时间内求解的判断问题。事实上，所有易解问题都属于P类问题。√

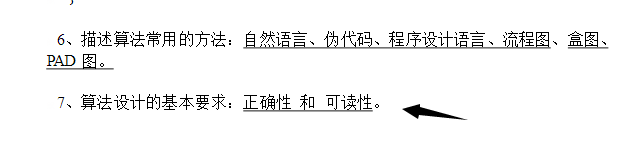


393.算法运行所需要的计算机资源的量，称为算法复杂性，主要包括时间复杂度和空间复杂度。√

394.可以从时间复杂度和空间复杂度两个方面评价一个算法的优劣。√

395.只能从时间复杂度一个方面评价一个算法的优劣。×

396.算法设计的基本要求是正确性和可读性。√



397.递归与循环都是解决“重复操作”的机制。√

398.就效率而言，递归算法的实现往往要比迭代算法耗费更多的时间与存储空间。√

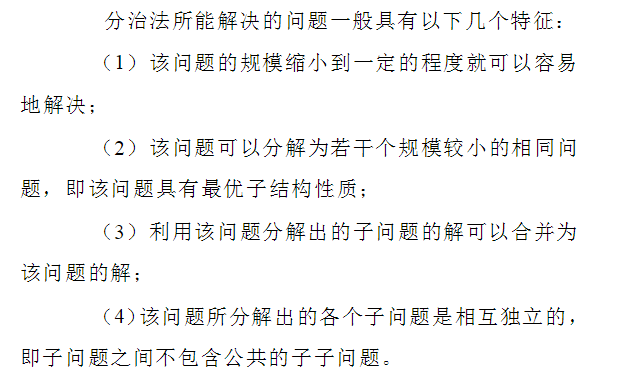
399.就效率而言，迭代算法的实现往往要比递归算法耗费更多的时间与存储空间。×

400.每个迭代算法原则上总可以转换成与它等价的递归算法；反之不然。√

401.递归的层次是可以控制的，而循环嵌套的层次只能是固定的，因此递归是比循环更灵活的重复操作的机制。√

402.循环是比递归更灵活的重复操作的机制。×

403.若某问题具有该问题的规模缩小到一定的程度就可以容易解决的特征，则此特征是该问题能用分治法求解的基本特征之一。 √



404. 若某问题可以分解为若干个规模较小的相同问题，即称该问题具有最优子结构性质。×

405.若某问题所分解出的各个子问题是相互独立的，则称该问题的子问题之间不包含公共的子问题。√

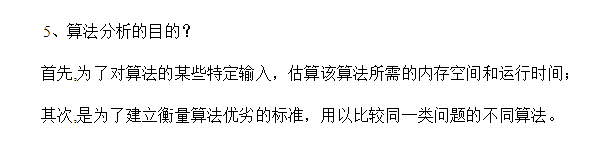
406.动态规划算法与分治法类似，其基本思想也是将待求解问题分解成若干个子问题；但是经分解得到的子问题往往不是互相独立的。√

407.算法由操作、控制结构、数据结构三要素组成。√

408.按照时间复杂度从低到高排列：O(2)，O(logn)，O(n2/3)，O(20n)，O(4n2)，O(n3

)，O(n!)√

409.算法分析的目的： 一是为了对算法的某些特定输入，估算该算法所需的内存空间和运行时间； 二是为了建立衡量算法优劣的标准，用以比较同一类问题的不同算法。√



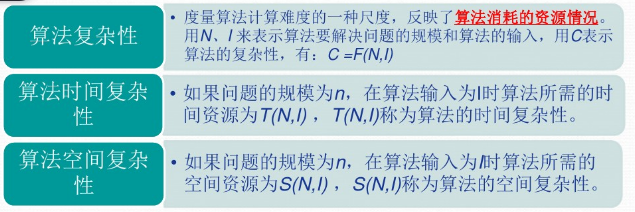
410.算法具备有限性、确定性、可行性、输入、输出等5方面特性，且缺一不可。×

411.在实际应用中常会遇到一些问题，不论采用确定性算法或概率算法都无法保证每次都能得到正确的解答。× 针对蒙特卡罗算法

412.如果问题的规模为n，解决该问题的某算法在算法的输入为I时所需要的时间为T（n，I），那么称T（n，I）为算法的时间复杂性。√

413.计算机资源最重要的是时间和空间资源，因此，算法的复杂性具有时间复杂性和空间复杂性之分。√

414.如果问题的规模为n，解决该问题的某算法在算法的输入为I时所需要的空间为S（n，I），那么称S（n，I）为算法的时间复杂性。× 空间

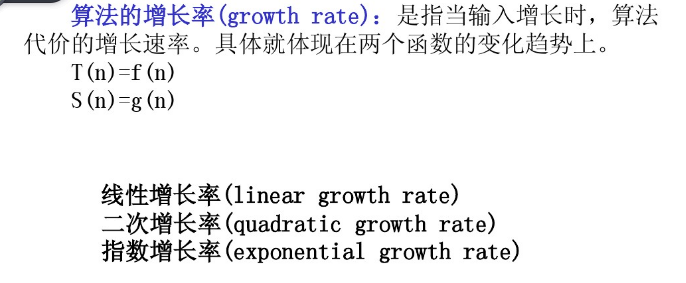


415.对某一算法，在不同的执行时间，它占用的内存空间量是相等的。×

416.精确的计算出算法的复杂性，往往是有必要的。×

417.精确的算出算法的性能指标很困难，其中一个原因是测试环境和测试数据的选择很难对各个算法公正。√

418.算法的增长率是指当输入增长时，算法代价的增长速率。如线性增长率、二次增长率、指数增长率等。√



419.算法的渐进分析指当输入规模很大，或大到一定程度时对算法的研究和分析。√

420.无论问题规模大小，系数起到的作用都可以忽略不计，因此往往省略系数。 ×

421.算法分析的步骤一般包括：计算基本操作、基本存储占用的数量；求出累计和函数T（n），S（n）；给出渐进表示。√

422.算法至少产生一个量作为输出，以反映对输入数据加工后的结果，没有输出的算法是毫无意义的。√

423.算法的可行性是指算法中有待实现的运算都是基本运算组成，原则上能够精确运行，而且在有限次运算后可以完成。√

424.Hanoi塔问题中，下面算法正确√

void hanoi(int n, int A, int B, int C) {

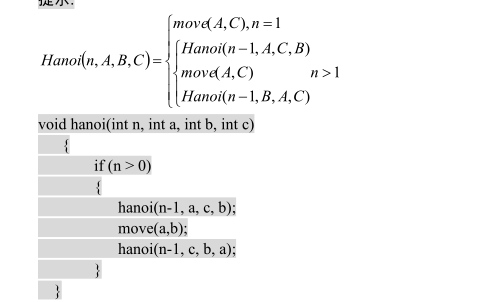
if (n > 0) {

hanoi(n-1, A, C, B);

move(n,a,b);

hanoi(n-1, C, B, A);

}



424.快速排序算法在最坏情况下退化成冒泡排序，需要比较n的立方次运算。×

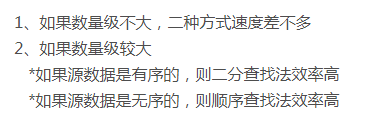
425.用确定的图灵机，可以在多项式时间内可解的判定问题称为P类问题。√

426.用不确定的图灵机，在多项式时间内可解的判定问题称为P类问题。×

427.NPC(NP Complete)问题，这种问题只有把解域里面的所有可能都穷举了之后才能得出。√

428.选择排序、插入排序和归并排序算法中，归并排序算法是分治算法。√

429. 二分搜索算法的效率比顺序搜索算法的效率高。 ×



430.程序运行的时间只取决于程序的复杂度。×

431.编写一个递归函数时，函数停止的条件不一定是必须存在的。×

432.直接或间接地调用自身的算法称为迭代算法，用函数自身给出定义的函数称为迭代函数。× 递归算法

433.程序调用自身的编程技巧称为递归，递归作为一种算法在程序设计语言中广泛应用。√

434.每个迭代算法原则上总可以转换成与它等价的递归算法；反之亦然。× 迭代转递归

435.递归算法的效率较低，在计算时间和占用的空间上都比非递归算法要多。√

436.递归算法的实现类似于多个算法的嵌套调用，只是调用算法和被调用算法是同一个算法。√

437. 一个使用函数自身给出定义的函数称为递归函数，定义递归函数时可以没有初始值。×

438.分治法对一个规模为n的问题分解为k个规模较小的子问题，这k个子问题的规模一定是相同的。×

439.从分治法设计出的程序一般都具有递归的特点，因此分治法的算法效率通常用递归方程来分析。√

440.分治法的基本思想是对一个规模为n的问题分解为k个规模较小的子问题，这些子问题相互独立且与原问题相同。√

441.分治法在将子问题求解后，需要将子问题的解合并，并得到原问题的解。√

442.在采用分治法时，在最坏的情况下讨论时间复杂度，用等号和小于等于是有本质区别的。×

443.在采用二分搜索算法时运用分治策略，给定的元素可以是没有排好序的元素序列。×

444. 动态规划求解问题，分解得到的子问题往往是相互独立的。×

445.采用分治法求解时，有些子问题被重复求解多次，因此，如果保存已经解决的子问题的

446.动态规划法采用一个表记录已解决子问题的

447.备忘录方法是动态规划法的变形，动态规划法是自顶向下的，而备忘录法是自底向上的。 × 动态规划自底向上，备忘录法自顶向下

448.动态规划思想设计的算法从整体上来看基本都是按照得出的递推关系式进行递推，这种递推相对于计算机来说，只要设计得当，效率往往是比较高的。√

449.动态规划法是多阶段决策过程，就是把研究问题分成若干个相互联系的阶段，由每个阶段都作出决策，从而使整个过程达到最优化。√

450.动态规划法求解问题时，最优子结构性质指原问题的最优解包含其子问题的最优解。√

451.矩阵连乘问题的算法可以通过动态规划算法设计实现。√

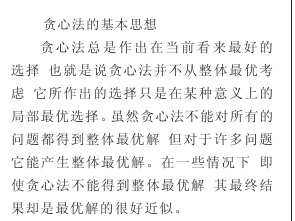
452.动态规划法中存储子问题的解是为了节省存储空间。×

453.Johnson算法的流水作业调度采用的算法是动态规划法。√

454．贪心算法在当前状态下做出做好的选择，即局部最优解。√

455.贪心算法不是对所有问题都能得到整体最优解，但是对范围相当广的许多问题能够产生整体最优解。√

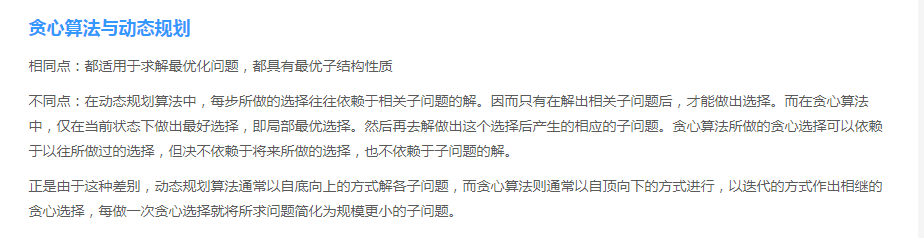
456.贪心算法虽然不能得到整体最优解，但是一定可以得到最优解的很好的近似解。×



457．贪心算法通常以自顶向下的方式解各子问题，而动态规划法通常以自底向上的方式解各自问题。√

458.贪心算法和动态规划法都要求问题具有最优子结构的性质。√

459.贪心算法做的贪心选择可以依赖于以往做过的选择，同时依赖于将来所做的选择。×



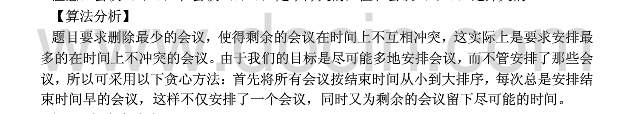
460.能采用贪心算法求最优解的问题，一般具有的重要性质为：最优子结构性质与贪心选择性质。√

461.背包问题的目标函数和贪心算法最优化量度相同。× 目标函数：获得最大利润。最优量度：最大利润/重量比

462.贪心算法总是做出在当前看来最优的选择，也就是说贪心算法并补充整体最优考虑。 √

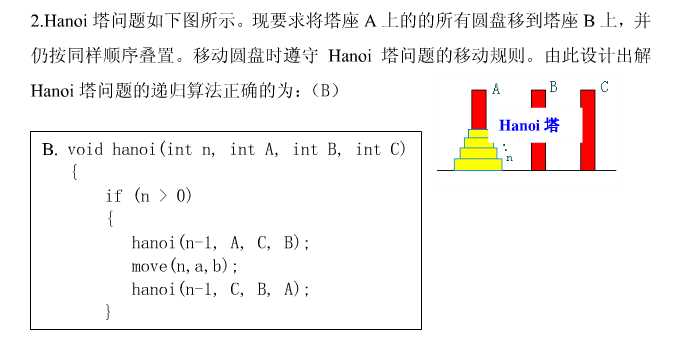
463.在一个医院B超室，有n个人要做不同身体部位的B超，已知每个人需要处理的时间为ti，（0<i<=n）,请求出一种排列次序，使每个人排队等候时间总和最小。n个人时间从小到大排序，就是这n个人最佳排队方案。√

464.某公司的会议日益增多，以至于全公司唯一的会议室都不够用了。现在给出这段时期的会议时间表，要求你适当删除一些会议，使得剩余的会议在时间上互不冲突，要求删除的会议最少。可采用以下贪心方法：首先将所有会议按结束时间从小到大排序，每次总是安排结束时间早的会议，这样不仅安排了一个会议，同时又为剩余的会议留下尽可能多的时间。√



465.回溯法在问题的解空间树中，按深度优先策略，从根结点出发搜索解空间树，如果某个结点不包含问题的解，则跳过对以此结点为根的子树的搜索。√

466.回溯法的效率不依赖于问题的解空间的形式；√



467.回溯法（探索与回溯法）是一种选优搜索法，又称为试探法，按选优条件向前搜索，以达到目标。但当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择。√

468.回溯法中有扩展节点、活结点、死结点：所谓扩展节点通过与约束函数的对照，节点本身和其父节点均满足约束函数要求的节点。×

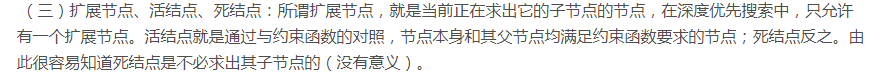
https://blog.csdn.net/tc\_1337/article/details/80454362

469.用回溯法求问题的所有解时，要回溯到根，且根结点的所有可行的子树都要已被搜索遍才结束。而若使用回溯法求最优解时，只要搜索到问题的一个解就可以结束。×

https://blog.csdn.net/GoOnDrift/article/details/5846506



470.回溯法中有扩展节点、活结点、死结点：所谓活结点，就是当前正在求出它的子节点的节点，在深度优先搜索中，只允许有一个活节点。 扩展结点 ×



471.即使利用限界函数也不能完全避免搜索移动到不可能产生解的子空间。√

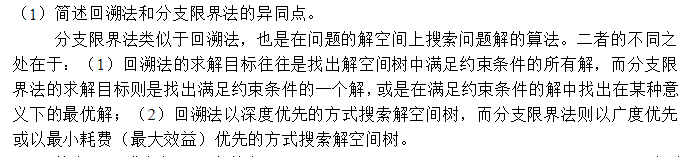
472.问题的解空间通常是在搜索问题的解的过程中动态产生的，这是回溯算法的一个重要特性。√

473.解决一个问题的所有可能决策序列构成该问题的解空间，任一满足约束条件的决策序列都一定是最优解。 × 可行解

474.如果采用树的非递归深度优先搜索遍历算法；也可以将回溯法表示为一个非递归的迭代过程。√

475.n皇后问题回溯算法的判别函数的基本流程是：将第K行的皇后分别与前k-1行的皇后比较，看是否与它们相容，如果不相容就返回false，测试完毕则返回true。√

476.回溯法的求解目标是找出解空间树中满足约束条件的所有解，而分支限界法的求解目标则是找出满足约束条件的一个解，或是在满足约束条件的解中找出在某种意义下的最优解。√



477.回溯法以深度优先的方式搜索解空间树，而分支限界法则以广度优先或以最小耗费优先的方式搜索解空间树。√

478.在分支限界法中，每一个活结点有多次机会成为扩展结点。× 只有一次机会

479.分支限界法常以广度优先或以最小耗费（最大效益）优先的方式搜索问题的解空间树。问题的解空间树是表示问题解空间的一棵有序树，常见的有子集树和排列树。√

480.在搜索问题的解空间树时，分支限界法与回溯法对当前扩展结点所使用的扩展方式完全相同。×

481.分支限界法则以广度优先或以最小耗费优先的方式搜索解空间树T。分支限界法的搜索策略是：在扩展结点处，先生成其部分的儿子结点（分支），然后再从当前的活结点表中选择下一个扩展结点。× 所有

482.分支限界法问题的解空间的形式：队列式（FIFO）分支限界法与优先队列式分支限界法 ×

483.随机化算法是这样一种算法，在算法中使用了随机函数，且随机函数的返回值直接或者间接的影响了算法的执行流程或执行结果。√

484.随机化算法允许执行过程中随机选择下一步计算步骤。√

485.随机化算法基于随机方法，但不依赖于概率大小。× 依赖于概率大小

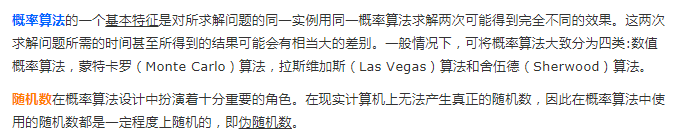
486.随机化算法是一个很好的概率算法，但是它并不能保证正确。√

487.随机化算法是一个很好的概率算法，但是它并不能保证正确，而且它单独使用的情况很多。× 很少

488.概率算法的一个基本特征是，对所求解问题的同一实例用同一概率算法求解两次可能得到完全不同的效果。√

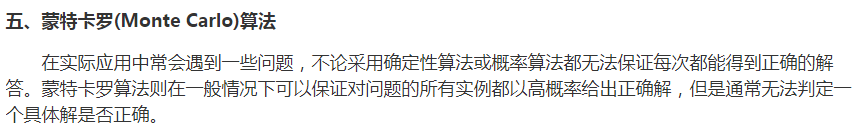
489.概率算法的一个基本特征是对所求解问题的同一实例用同一概率算法求解两次可能得到完全不同的效果。但是这两次求解问题所需的时间甚至所得到的结果的差别一定不会太大。× 相当大的差别

490.在现实计算机上无法产生真正 的随机数，因此在概率算法中使用的随机数都是一定程度上随机的，即伪随机数。 √



491.有些蒙特卡罗算法除了具有描述问题实例的输入参数外，还具有描述错误解可接受概率的参数。√

492.在实际应用中常会遇到一些问题，不论采用确定性算法或概率算法都无法保证每次都能得到正确的解答。但蒙特卡罗算法可以保证对问题的所有实例都以高概率给出正确解。√

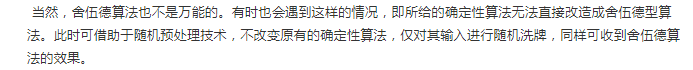


493.在实际应用中常会遇到一些问题，不论采用确定性算法或概率算法都无法保证每次都能得到正确的解答。蒙特卡罗算法则在一般情况下可以保证对问题的所有实例都以高概率给出正确解，但是通常无法判定一个具体解是否正确。√

494.在一般输入数据的程序里，输入或多或少会影响到算法的计算复杂度。这时可用舍伍德算法消除算法所需计算时间与输入实例间的这种联系。√



495.舍伍德算法也不是万能的。如果所给的确定性算法无法直接改造成舍伍德型算法。此时可借助于随机预处理技术，不改变原有的确定性算法，仅对其输入进行随机洗牌，同样可获得舍伍德算法的效果。√



496. 数值随机化算法求解数值问题的近似解，精度随计算时间增加而不断提高。√

497.舍伍德算法的精髓不是避免算法的最坏情况行为，而是设法消除这种最坏行为与特定实例之间的关联性。√

498.拉斯维加斯算法的目标是求解问题的正确解，但是可能找不到解。√

499.蒙特卡罗算法求解问题的准确解，但在一般情况下无法判定解的正确性。√

500.蒙特卡罗算法求得正确解的概率随时间增大而增大，当时间增加到一定程度，一定会保证解的正确性。×

501.对于两个正整数m和n，可使用欧几里得算法求出它们的最大公约数。√

502.对于两个正整数m和n，可使用辗转相除法和辗转相减法求出它们的最大公约数。√

503.对于两个正整数m和n，可使用连续整数检测算法求出它们的最大公约数。√

504.完全平方数的约数个数为奇数个，其他所有数的约数个数为偶数个。√

505.完全平方数的约数个数为偶数个。×

506.对于扩展欧几里得算法，不仅能求出两个正整数m和n的最大公约数d，还能求出两个整数x和y（不一定为正），使得mx+ny=d。√

507.求圆周率的问题能够精确求解。×

508.求圆周率的问题只能近似求解。√

509.可以使用投针法计算圆周率。√

510.旅行商问题（TSP），就是要找出访问n个城市的最短路径，并且保证每个城市只访问一次。√

511.对于排序算法来说，如果一个算法还需要额外的存储空间，我们就说它是在位的。×

512.排序算法有些是在位的，有些不是在位的。√

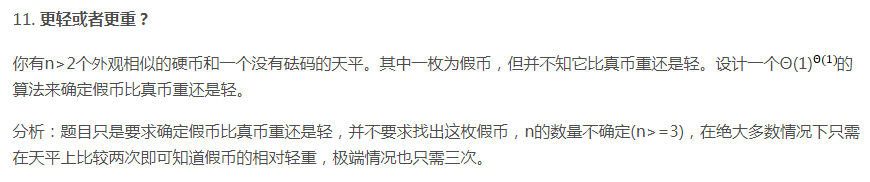
513.对于下面的应用：“按照来电的优先顺序接电话”，采用优先队列是最适合的数据结构。√

514.对于下面的应用：“按照客户订单的接收顺序向客户发货”，采用队列是最适合的数据结构。√

515.对于下面的应用：“实现一个简单算术表达式的计算器”，采用栈是最适合的数据结构。√

516.对于n的阶乘的计算，使用史特林公式，可以得到n阶乘的一个近似值。√

517.假设有n>2个外观相似的硬币和一个没有砝码的天平。其中一枚为假币，但并不知道它比真币重还是轻，无法设计一个θ(1)的算法来确定假币比真币重还是轻。×



518.n的平方等于前n个奇数的和。× n个连续奇数的和等于个数的平方

519.对于汉诺塔问题，可以用非递归的思想求解。√

520.对于汉诺塔问题，如果第一根柱子上有n个圆盘，要将这n个圆盘移动到第三根柱子上，总共需要移动2n-1次。 √

521. 针对大整数计算anmodm，是一种重要密码算法的主要组成部分。√

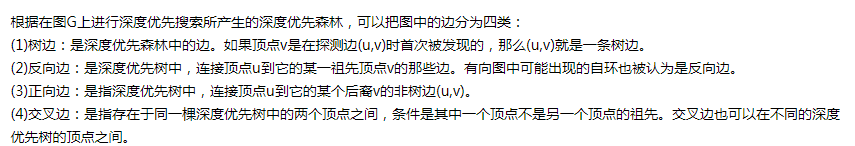
522.对于字符串匹配问题，可以用蛮力法求解。√

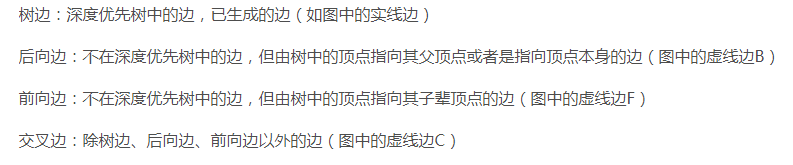
523.对于最近对问题，可以使用蛮力法求解。√

524.对于凸包问题，可以使用蛮力法求解。√

525. 对于分配问题，有一个效率较高的算法，称为匈牙利方法。

526.对于深度优先查找算法，边的类型有树向边和回边两种。 √

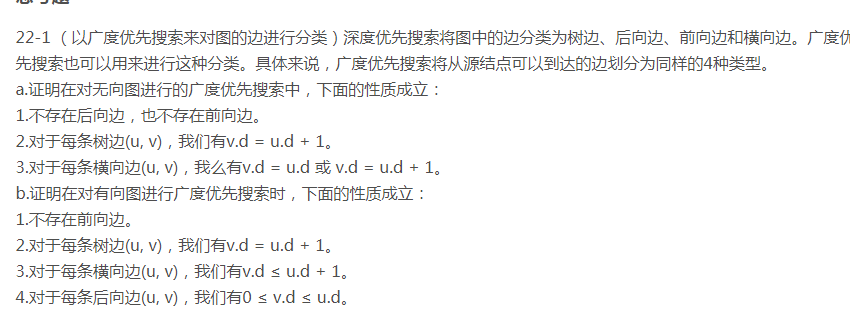




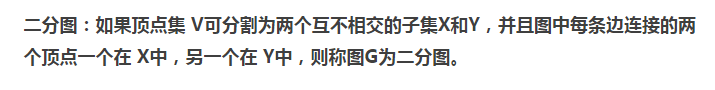
527.对于深度优先查找算法，边的类型有树向边和交叉边两种。√

528.对于广度优先查找算法，边的类型有树向边和回边两种。×

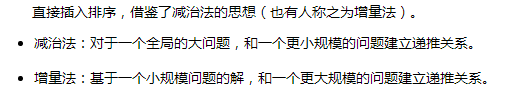
529.对于广度优先查找算法，边的类型有树向边和交叉边两种。√



530.如果图中的顶点可以分为两个互不相交的子集X和Y，使得每条连接X中顶点的边都连接着Y中的顶点，这样的图叫二分图。√



531.直接插入排序是基于减治法的思想。√



532. 拓扑排序可以判定一个有向图是否存在回路。√

533. 对于一个有向无环图，拓扑排序算法可以输出节点的一个先后次序，从而实现任务的安排。√

534.对于二进制反射格雷码，可以使用递归算法来生成。√

535.一个集合的所有子集的集合称为它的幂集。√

536.三重查找的效率比折半查找更高。√

537.三重查找的效率比折半查找低。×

538.对于俄式乘法，该算法只包括折半，加倍和相加这几个简单操作，就可以实现两个正整数向乘。√

539.对于约瑟夫斯问题，在n=40的情况下，J(40)=17。√

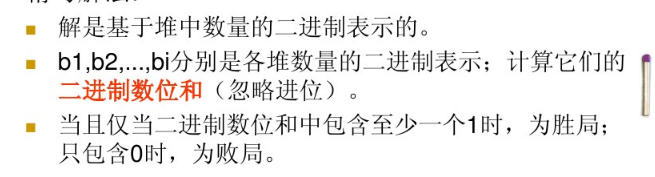
540.对于约瑟夫斯问题，在n=40的情况下，J(40)=15。×

541.对于所有为2的乘方的n来说，它的约瑟夫斯问题的解是1。√

542.对于一个具有n个数的列表，要求找出一个元素，该元素比列表中一半的元素大，又比另一半的元素小，这个元素值被称为中值。×

543.对于拈游戏的胜负，和各堆棋子数的二进制数位和有关。√

544.对于拈游戏问题，当且仅当二进制数位和中包含至少一个1时，该实例是一个胜局。√



545.对于拈游戏问题，当且仅当二进制数位和只包含0时，该实例是一个败局。√

546.对于拈游戏问题，当且仅当二进制数位和中包含至少一个1时，该实例是一个败局。×

547.对于拈游戏问题，当且仅当二进制数位和只包含0时，该实例是一个胜局。×

548.快速排序是基于分治法的思想。√

549.合并排序是基于分治法的思想。√

550.二叉树遍历是基于分治法的思想。√

551.大整数乘法可以运用分治技术获得更好的渐近效率。√

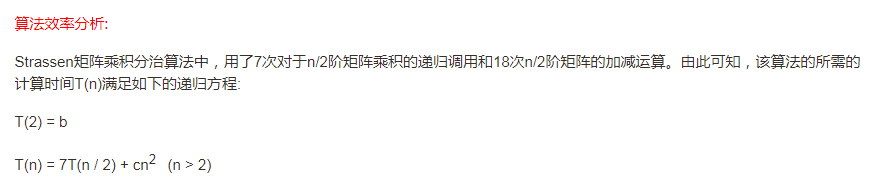
552.对于最近对问题，可以使用分治法求解。√

553.对于凸包问题，可以使用分治法求解。√

554.对于最近对问题，不能使用分治法求解。×

555.对于凸包问题，不能使用分治法求解。×

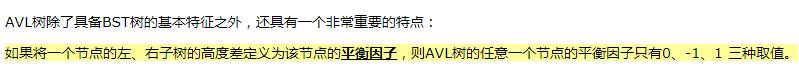
556. 使用Strassen算法，对两个n>1阶矩阵相乘，该算法要对n/2阶矩阵做7次乘法和18次加法。√



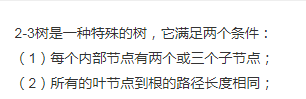
557.高斯消去法是对线性方程组求解的一种有效算法。√

558. LU分解是对线性方程组求解的一种有效算法。√

559.对于AVL树，每个节点的平衡因子要么为0，要么为+1，要么为-1。√

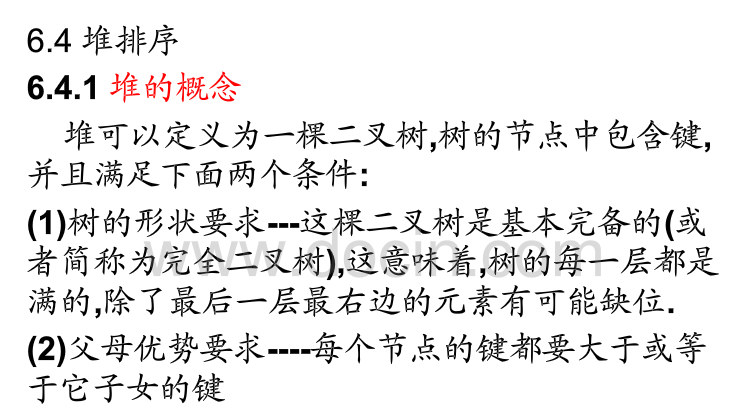


560. 2-3树是一种可以包含两种类型节点的树，2节点和3节点。√



561. 2-3树只能包含一种类型的节点。×

562. 堆可以定义为一棵二叉树，树的节点中包含键，并且满足下面两个条件：树的形状要求和父母优势要求。√



563. 对于堆的构造，有两种主要的做法，第一种叫自底向上堆构造，第二种叫自顶向下堆构造。√

564. 霍纳法则是一个古老的计算多项式的算法，十分优雅和高效。√

565.对于二进制幂的计算，有从左至右二进制幂算法和从右至左二进制幂算法两种方式。√

566.对于两个正整数m和n，lcm(m, n)表示两者的最小公倍数，则lcm(m, n)可以通过欧几里得算法间接计算出来。× 欧几里得求最大公约数

567.对于两个正整数m和n，lcm(m, n)表示两者的最小公倍数，则lcm(m, n)无法通过欧几里得算法间接计算出来。√

568.从图（无向图或有向图）中第i个顶点到第j个顶点之间，长度为k>0的不同路径的数量等于Ak的第(i, j)个元素，其中A是该图的邻接矩阵。√

569.单纯形法是求解线性规划问题的一种有效算法。√

570.单纯形法无法求解线性规划问题。×

571.农夫带羊、狼、白菜过河问题可以用状态空间图加以描述。√

572. KMP算法可用来解决字符串匹配问题。√

573. KMP算法无法用来解决字符串匹配问题。×

574. Horspool算法可用来解决字符串匹配问题。√

575. Boyer-Moore算法可用来解决字符串匹配问题。√

576. 币值最大化问题，可以使用动态规划法求解。√

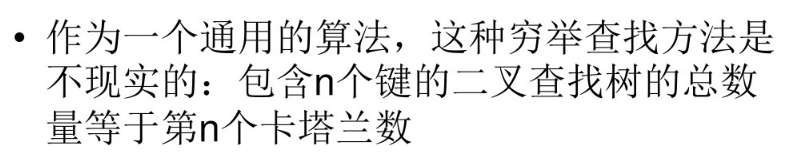
577.找零问题，可以使用动态规划法求解。√

578. 硬币收集问题，可以使用动态规划法求解。√

579.对于最优二叉查找树问题，可以使用动态规划法求解。√

580.对于最优二叉查找树问题，无法使用动态规划法求解。×

581.包含n个键的二叉查找树的总数量等于第n个卡塔兰数。√

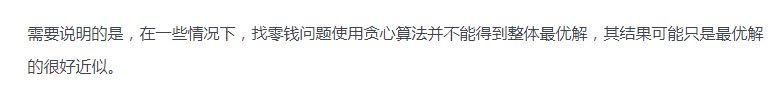


582.Warshall算法是基于动态规划的思想。√

583.对于具有n个顶点的有向图，其传递闭包矩阵和邻接矩阵是完全相同的。

584.对于计算完全最短路径的Floyd算法，其是基于动态规划的思想。√

585.对于找零问题的所有实例，都能用贪心算法找到最优解。×



586.Prim算法是用来解决最小生成树问题的。√

587. Kruskal算法是用来解决最小生成树问题的。√

588.Dijkstra算法是基于贪婪法的思想。√

589.Prim算法和Kruskal算法的流程是完全相同的。×

590.n皇后问题可以使用回溯法求解。√

591.哈密尔顿回路问题可以通过回溯法求解。√

592.子集和问题可以通过回溯法求解。√

593.分配问题可以通过分支限界法求解。√

594.离散背包问题可以通过分支限界法求解。√

595.旅行商问题可以通过分支限界法求解。√

596.对于旅行商问题，有一种基于最近邻居的近似算法。√

597.对于旅行商问题，基于最近邻居的近似算法精确率较高。√

598.对于旅行商问题，有一种绕树两周的近似算法。√

599.Christofides算法是一种解决旅行商问题的近似算法。√

600.2选算法是一种解决旅行商问题的近似算法。

601.3选算法是一种解决旅行商问题的近似算法。

602.对于离散背包问题的所有实例，使用贪婪算法总能产生一个最优解。×

603.对于连续背包问题，使用贪婪算法总是能够产生最优解。×

604.对于所有的非线性方程f(x)=0，都能够求出它的精确解。×

605.折半查找解决的是离散问题，平分法求根解决的是连续问题。√

606.折半查找速度快，平分法求根速度不快。×

607.平分法求根要用到函数f(x)的导数。×

608.牛顿法求根要用到函数f(x)的导数。√

609.牛顿法求根不需要使用函数f(x)的导数。×

610.平分法求根的速度比牛顿法快。×

611.平分法求根的速度比牛顿法慢。√

612.试位法是对非线性方程求根的一种近似算法。√