选择题（每题2分）

采用分治法解决问题时，算法时间复杂度分析典型的递推方程为，对于某一个问题，若，且，则求该问题的时间复杂度为（B）

（多选）以下哪个问题适合用贪心算法求解（BCD）

0-1背包问题 哈夫曼编码 单源最短路径 最优装载问题

贪心算法之所以不能求解0-1背包问题，是因为在贪心选择的情况下，算法无法保证背包能被装满，导致有部分背包空间浪费了。

适合用回溯法和分支定界法求解的问题均需要满足（C）

重叠子问题 最优子结构性质 多米诺性质 独立子问题

重叠子问题：动态规划

最优子结构：贪心、动态规划

多米诺性质：回溯、分支定界

独立子问题：分治

贪心选择性质：贪心

以下哪个问题的解空间树是排列树（C）

n皇后问题 最大团问题 批作业调度问题 图的m着色问题

排列树：TSP问题、批作业调度问题、电路板排列问题

子集树：n皇后问题、装载问题、0-1背包问题、最大团问题、图的m着色问题，高精度数问题、布线问题

以下关于渐进符号性质表达正确的是（C）

如果

如果

如果

回溯法一般是按（A）策略搜索问题的解空间树

深度优先 广度优先 广度深度结合 活结点优先

回溯法：深度优先

分支定界法：广度优先或函数优先

适合应用贪心算法与动态规划算法求解的问题所具有的共同特点是（B）

重叠子问题 最优子结构性质 贪心选择性质 独立子问题

对于物品数量为n，背包容量为c的情况下的0-1背包问题，若用回溯法求解，算法的时间复杂度为（D）

回溯法：遍历树的时间复杂度\*计算约束的时间（生成一个节点的时间），子集树遍历，排列数遍历

装载问题：子集树，

批处理作业调度：排列树，

N皇后问题：子集树，

0-1背包：子集树，

最大团问题：子集树，

图的m着色问题：子集树，

TSP问题：排列树，

分支限界法：

装载问题：

0-1背包问题：

TSP问题：

布线问题：

最大团问题：

批作业调度问题：

分治法：

全排列：

整数划分

二分搜索：

大整数乘法：

矩阵乘法:

合并排序：

快速排序：最坏：、最好和平均：

线性时间选择：

最近点对问题：

棋盘覆盖问题：

动态规划：

矩阵连乘问题：

凸多边形最优三角剖分：

最长公共子序列：

电路布线问题：

图像压缩问题：

最大子段和问题：

投资问题：

0-1背包问题：

0-N背包问题：

最优二叉搜索树：

流水作业调度：

序列匹配：

贪心：

活动安排：早完成的活动先安排，

背包问题和0-1背包问题：背包问题可用贪心，0-1背包不行，单位重量价值最高的物品先装，

最优装载问题：集装箱按从轻到重排序，轻者先装，

最优前缀码问题（哈夫曼）：

最小生成树问题（Prim（加点）和Kruskal（加边））：

当图的节点数为n时，Prim算法所需的计算时间是

当图的边数为e时，Kruskal算法所需的计算时间是。

当时，Kruskal算法比Prim算法差，但当时，Kruskal算法却比Prim算法好得多。

单源最短路径问题（Dijkstra算法）：

多机调度问题：

硬币找零问题：只有当，找零问题才满足贪心性质

填空题（每空2分）

算法是由若干指令组成的有穷序列，需要满足\_\_输入\_\_、\_\_输出\_\_、\_\_确定性\_\_、\_\_有效性\_\_、\_\_有限性\_\_

算法的效率一般可用从两个方面衡量，分别称为\_\_时间复杂度\_\_和\_\_空间复杂度\_\_。

假设算法A理论上的时间复杂度，现在有两台同类型计算机M1和M2。M2的计算速度是M1的64倍。若在M1和M2上分别测试算法A，则在相同的时间内，M1和M2能够求解的问题规模n1和n2的关系为\_\_n1=n2-6\_\_。若算法B理论上的时间复杂度为，若在M1和M2上分别测试算法B，则M1和M2求解相同规模问题所耗费的时间t1和t2的关系为\_\_t1=64\*t2\_\_。

A：

B:

用回溯法求解问题时，可将问题解空间看作一棵树状结构，根据问题不同，解空间树一般可分为子集树和排列树两种结构，对于N皇后问题，其解空间树是\_\_子集树\_\_结构。对于TSP问题，其解空间树是\_\_排列树\_\_。结构。

用分治法求解n个元素的全排列问题时，算法时间复杂度递推方程为\_\_\_\_。

0-1背包问题可用适用动态规划、回溯和分支定界三种方法求解，其中排序对求解过程没有影响的方法是\_\_动态规划\_\_。

若序列，则X和Y的一个最长公共子序列为\_\_\_\_；采用动态规划方法求解该类问题（序列X和Y的规模分别为m和n）的时间复杂度为\_\_\_\_。

采用分治法解决问题时，算法时间复杂度分析典型的递推方程为，对于某一问题，若且，则求解该问题的时间复杂度为\_\_\_\_；若且，则求解该问题的时间复杂度为\_\_\_\_；若且，则求解该问题的时间复杂度为\_\_\_\_；若且，若希望将算法时间复杂度降低为不超过线性情况，则a和b应该满足的条件为\_\_\_\_。

应用贪心算法求解问题的关键在于\_\_贪心策略的选择\_\_。

动态规划算法的求解步骤有四步，分别为\_\_找出最优子结构\_\_、\_\_建立递推关系式\_\_、\_\_自底向上计算最优解\_\_、\_\_构造最优解\_\_。

分治的求解步骤有三步：划分、求解子问题、合并

回溯和分支定界的求解步骤：定义解空间（问题解向量、解向量分量取值集合、构造解空间树）、判断问题是否满足多米诺性质、确定界函数和剪枝函数，搜索解空间树（、确定存储搜索路径的数据结构）

对于有n个顶点，m条边的连通带权图，如果用Kruskal算法求其最小生成树，则时间复杂度为\_\_\_\_；如果用Prim算法求其最小生成树，则时间复杂度为\_\_\_\_。

遗传算法主要借鉴的是生物界的\_\_自然选择\_\_和\_\_自然遗传\_\_，遗传操作有\_\_选择\_\_、\_\_交叉\_\_和\_\_变异\_\_。

遗传算法评价一个解的好坏用\_\_适应度函数\_\_，其设计标准是\_\_适应度函数值越大，解的质量越好\_\_。

遗传算法与收敛性相关的因素：种群规模、选择操作、交叉概率、变异概率。

基本遗传算法主要由4部分组成分别是\_\_编码\_\_、\_\_适应度函数\_\_、\_\_遗传算子\_\_、\_\_运行参数\_\_。

分支定界法主要有两种搜索方式，分别为\_\_FIFO搜索\_\_和\_\_优先队列搜索\_\_。

节点v的上界UB(v)是指从v出发得到的所有叶子节点的效益值均\_\_不大于\_\_ UB(v)，如果所有叶子节点的最小效益值等于节点v的下界LB(v)，则LB(v)为节点v的\_\_下确界\_\_。

分支限界法求解极小化问题时，主要用节点的\_\_下界\_\_进行剪枝。

回溯法一般按照深度优先的策略搜索解空间树，在搜索的过程中主要通过\_\_约束函数\_\_和\_\_界限函数\_\_避免无效的搜索。

算法设计与分析题（每题10分）

设B为数组，B[1…n]中存储了n个元素，分析下面的算法并回答问题。

（1）该算法的输出是什么？

（2）以比较作为基本操作，给出算法ALG的时间复杂度递推方程，并求解算法ALG的时间复杂度。

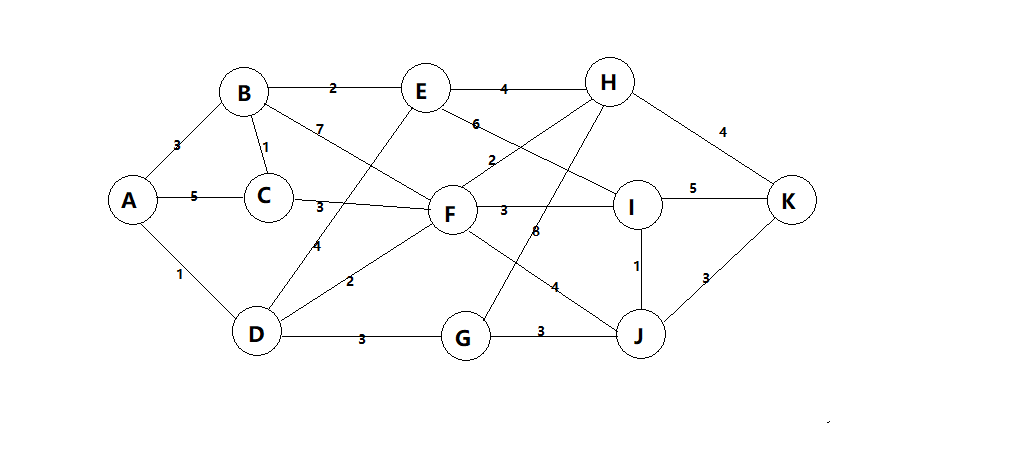
（3）对于求解该问题，算法ALG效率是否已最高，为什么？

（1）

（2）

（3）是，因为要找出数组中最大元素，需要遍历数组，效率至少是到

有如下图所示的地图，图中有A-K共11个地点，两点之间的路径长度为边中的数字，如A到B的路径为3。求A到K的最短路径？

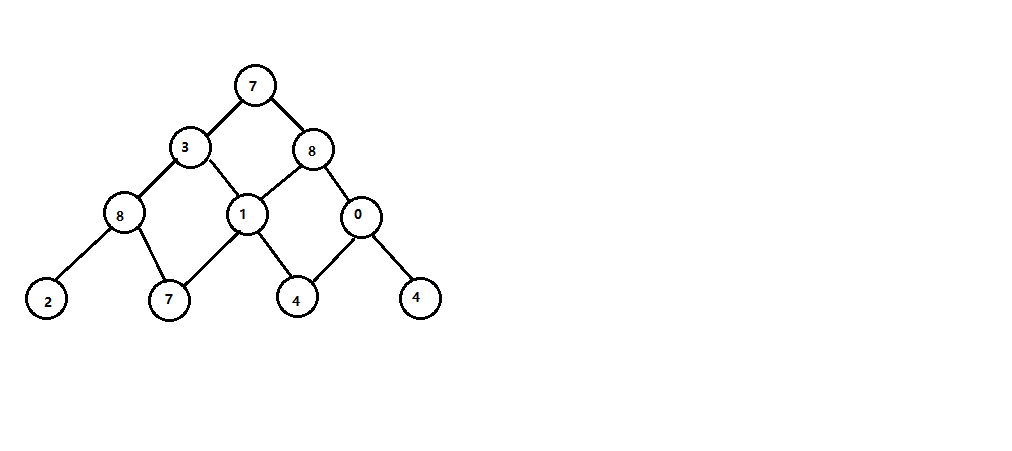




（1）若用动态规划算法求解该问题，请给出最短路径的递推定义

（2）针对上图，按照你所给出的递推定义，逐步求解从A走到K的最短路径

有形如下的数塔，若要求从数塔顶层出发，每个节点可以选择向左走或向右走，一直走到塔底，使得走过路径上节点的数值之和最大，回答以下问题：



（1）若用动态规划算法求解该问题，请给出最优值的递推定义，包括初值说明

（2）针对上图的数塔，按照你所给出的递推定义，逐步求解从塔顶走到塔底的最大数值之和与得到最大数值和的路径

（1）定义数塔的存储结构是，

DP的初值为

则递推公式为

表示从顶层到第j层，走过路径上节点的最大数值之和，表示的左父母节点，表示的右父母节点。最优值为

（2）

最优值为，路径自底向上为，

自顶向下即

已知一个国家有n个城市，已知每个城市过去一年的GDP总量，请找出其中GDP总量最接近的两个城市，回答以下问题：

（1）用数学方法描述该问题的求解目标

（2）为该问题设计一个有效算法，需要给出算法思想和伪代码

（3）对你所设计的算法进行时间复杂度分析

（4）对于求解该问题，你所设计的算法的效率是否已最高，为什么？

（1）已知中存储了n个城市过去一年的GDP总量，请找出两个城市i和j，

使，对都成立。

（2）算法思想：先对city数值从大到小排序，排序的过程中，记录排序后的下标与原数组下标之间的映射关系，然后遍历数组计算，找出最小的那个GDP差值对应的和，然后通过映射表，找到原本的两个城市。

伪代码：

// 对city数组从大到小排序，并将排序前后下标的映射关系存入table字典

// 初值为第一个城市和第二个城市的GDP差值

// 存储最小差值下标

// 输出最小GDP差值对应的下标的原城市

（3）时间复杂度：

（4）的效率已经是最高，这个问题可以类比书上的最接近点对问题的一维情况，书上证明了的效率是该类问题的下界。

已知有n个学生，若要求找出其中体重最为接近的两位学生，回答以下问题：

（1）用数学方法描述该问题求解目标

（2）为该问题设计一个有效算法，需要给出算法思想和伪代码

（3）对你所设计的算法进行时间复杂度分析

同上

算法设计与实现（每题分值不定，大概10+）

设有n个工作分配给n个人，将工作i分配给第j个人的费用为。给定列表L，表示不能将工作分配给工人的集合，其中表示工作i不能分配给第j个人。请设计算法，计算最佳工作分配方案，使得总费用最小。

要求：

（1）分析该问题适合用何种算法求解

（2）给出问题解的形式

（3）给出求解问题的代码框架

（4）分析算法的时间复杂度

（1）适合用蛮力法，回溯法，分支限界法求解。

（2）问题解的形式：n元组，其中表示将第个工作分配给第i个人。

（3）

使用回溯法，递归框架：

; // 当前最小费用

; // 当前最小费用对应的解

// 存储当前的费用

：

// 第t个工人是否能安排做第j个工作

:

;

;

使用回溯法，迭代框架：

{

：

（4）此题的解空间树是排列数，时间复杂度是

某集团领导年底要到该集团的各个分部视察工作。已知该集团共有n个分部。给定n阶矩阵C，其中表示领导从分部i到分部j所需要花费的代价。如果分部i到分部j不可达，则。求从分部1出发，视察每个分部一次后，再回到分部1所需的最少花费？若用优先队列式分支限界法求解该问题，回答以下问题：

（1）每个搜索节点需要包含哪些信息，为什么？

（2）给出部分解的下界函数，越大越好

（3）给出部分解的上界函数，越小越好

（1）包含：

：子树的部分解下界

：当前费用

：中顶点最小出边费用和

：当前层，根节点到当前节点的路径为

: 需要进一步搜索的顶点是

（2）问题下界：将每个节点的最短路径求和。每个顶点选择最短的两条边作为出入边，求和再除2。

部分解下界：将已经经过的顶点的一条短的且没经过的边长换成已经经过的那条边。

部分解下界公式：

（3）问题上界：贪心算法，假设最后都能回到分部1，从分部1开始，每次去最近的分部。

部分解上界：贪心算法，假设最后都能回到分部1，从分部i开始，每次去最近的分部。

设某游艇俱乐部在长江沿岸设置了n个游艇出租站，1,2,…,n。游客可在这些游艇出租站租用游艇，并在任何另一个游艇出租站归还游艇。已知游艇出租站i到游艇出租站j之间的租金为。若要计算从游艇出租站1到游艇出租站n所需的最少花费，完成以下问题。

（1）若用动态规划方法求解该问题，给出问题解的递推定义。

（2）填空完成以下动态规划算法程序。

/\*\*

\*@param n 游艇租赁站数量

\*@param r 租赁站之间的租赁费用矩阵

\*/

（1）

（2）

算法思想：

分治：将一个难以直接解决的大问题，分解成规模较小的相同子问题，直至这些子问题容易直接求解，并可利用这些子问题的解求出原问题的解。

动态规划：和分治法类似，将原问题分解成多个子问题，子问题之间往往不相互独立，保留已解决的问题的子问题，避免重复计算

贪心：总是找出当前最优解，局部最优解。

回溯、分支限界：从初始状态出发，搜索能达到是所有状态；当一条路走到尽头，则回退一步或多步，从另一个状态继续出发，直到所有路径都搜索过。

分支限界法：

装载问题（FIFO）：

// 集装箱重量

// 父节点

// 是否是左儿子节点

装载问题（优先队列）：

// 载重上界

// 所在层

// 父节点

// 是否为左儿子节点

最大团问题（优先队列）：

// 最大团节点数上界

// 所在层

// 父节点

// 是否为左儿子节点

批作业调度问题（优先队列）：

// 优先级信息

// 所在层

// 当前作业调度顺序

// 当前调度的完成时间和

// 两个机器当前的完成时间

分支限界代码框架：

// Q存储所有的活结点，初始化为根节点

// 从中选择一个活结点

// 对q进行分支，产生，分支时利用约束和界限进行剪枝

// 将新产生的活结点加入

装载问题：

// 当前最优载重和

// 最优载重对应的解向量

// 当前扩展节点

// 扩展节点的层

// 扩展节点当前的载重

// 定义剩余重量数组

// 初始化

// 保留最优解

：

: // 如果搜索到叶子节点层，且该节点的上界是所有节点的上界中最大的，那么从其他节点出发产生的最大载重量都不可能比这个值大

// 搜索左子树

// 更新bestw

// 更新bestw，构造新的bestNode

// 将左节点加入队列

// 利用上界剪枝

// 取下一个节点

回溯法：

N皇后问题（递归）：

// 解法+1

;

// 检查新放置的皇后是否和之前的皇后之间有冲突