中南大学考试试卷

2019 -- 2020 学年 2 学期　　时间 100 分钟　　年 月 日

算法分析与设计 课程 48 学时 3 学分 考试形式：闭 卷

专业年级：计科、物联、信安、大数据 18 总分 100 分，占总评成绩 60 %

注：此页不作答题纸，请将答案写在答题纸上

一、判断正误题，正确的后面标 T，错误的后面标 F（本题 10 分，每空 1 分）

1. 对于任意 x; y > 1，有 $n^x = O(y^n)$.　T
2. 对于动态规划算法，以自底而上的方式计算所有值的渐近速度比使用递归和备忘录的速度快。　F
3. 如果图中包含负权重边，Dijkstra 的算法可能不会终止。F
4. 考虑加权有向图 G =（V; E; w），令 X 为一条最短 s-t 路径，s, t ϵV。 如果将图中每条边的权重加倍，对于每条边 e, w'（e）= 2w（e），则 X 仍将是（V; E; w'）中的最短 s-t 路径。T
5. 给定一个无向图，可以在 O（V + E）时间内对其进行测试以确定它是否是一棵树。T
6. Bellman-Ford 算法适用于单源最短路径问题的实例，这些实例不具有负权重有向环，但如果存在负权重有向环，则不会检测到。 F
7. 任意有向无环图 G =（V; E）的拓扑排序可以在线性时间内计算得出。　T
8. 在基于比较的排序中，可以最多使用七次比较对五个元素进行排序。T
9. 给定一个由 n 个整数组成的数组，每个整数都属于{-1； 0; 1}，在最坏的情况下，可以按 O（n）时间对数组进行排序。T
10. 背包问题的最佳解决方案始终包含具有最大单位价值比 vi/ ci 的对象 i。F

二、简答题（本题 12 分，每小题 6 分）

1. 试比较 Prim 算法与 Kruskal 算法的异同。

答：

相同点：Prim(普里姆)算法和 Kruskal(克鲁斯卡尔)算法都可以看作是应用贪心算法构造最小生成树的例子。利用了最小生成树性质。

不同点：

Prim(普里姆)算法：在这个过程中选取到的所有边恰好构成 G 的一棵最小生成树 T，T 中包含 G 的 n-1 条边，且不形成回路。

Kruskal(克鲁斯卡尔)算法：是构造最小生成树的另一个常用算法。该算法不是通过扩充连通子集来进行贪心选择。而是通过选择具有最小权的边的集合来进行贪心选择。在选择的同时可以进行连通操作以便形成生成树。

2. 阐述动态规划算法与贪心算法的区别，它们都有那些优势和劣势？

答： 动态规划算法与贪心算法都要求问题具有最优子结构性质，这是二者的一个共同点。但是对于具有最优子结构的问题应该选择前者还后者来解决？下面通过两个经典的组合优化问题谈谈动态规划算法与贪心算法的主要差异

动态规划法与分治法和贪心法类似，它也是将原问题分解为若干个更小的、相似的子问题，并通过求解子问题产生一个全局最优解。与分治法和贪心法不同之处在于：

① 使用贪心法时，当前的选择可能要依赖于已经作出的所有选择，但不依赖于有待于做出的选择和子问题。因此贪心法是自顶向下(即从起点到终点)，一步一步地作出贪心选择。当然，如果当前的选择可能要依赖于子问题的解时，则难以通过局部的贪心策略达到全局最优解。

② 使用分治法时，由原问题分解出的各子问题通常是相互独立的，即不包含公共的子问题，因此一旦递归地求出各子问题的解后，便可自下而上地将各子问题的解合并成问题的解。如果各子问题不是相互独立的，则分治法要做许多不必要的工作，重复地求解公共的子问题。

③ 动态规划允许由原问题分解出的子问题之间相互依赖。每一个子问题只求解一次，并将结果保存起来，避免每次碰到此子问题时都要重复计算。

三、 计算与算法应用题（本题 48 分，每小题 12 分）

1. 假设您在以下三种算法之间进行选择：

(1) 算法 A：通过将大小为 n 问题分解为大小为 n / 4 的四个子问题，递归地解决每个子问题，然后在固定时间内组合解（取 T（1）= 1），从而解决原问题。

(2) 2.算法 B：通过递归求解每个大小为（n / 2）的四个子问题，然后在 O（n^2）时间内组合解（取 T（1）= 0），从而解决大小为 n 的问题。

(3) 3.算法 C：通过将大小为 n / 3 的三个子问题分成三个问题，递归求解每个子问题，然后在线性时间内组合解（取 T（1）= 0），从而解决了问题。

给出每种算法的运行时间（以 big-O 表示），并按复杂度进行升序排列。

Solution:

$$A:\quad T(n)=4T(n/4)+b,\quad T(1)=1,\quad m=\log_4 n$$

$$T(n)=4^m+(b/3)(4^m-1)=n+(b/3)(n-1)=O(n)$$

$$B:\quad T(n)=4T(n/2)+bn^2,\quad T(1)=0,\quad m=\log n$$

$$T(n)=bn^2\log n=O(n^2\log n)$$

$$C:\quad T(n)=3T(n/3)+bn,\quad T(1)=0,\quad m=\log_3 n$$

$$T(n)=bn\log_3 n=O(n\log n)$$

$$A<C<B$$

2. 对于下表中的每对函数 f（n）和 g（n），填写 0，Ω，Θ 在适当的空间，具体取决于 f（n）= 0（g（n）），f（n）=Ω（g（n））或 f（n）=Θ（g（n））。 如果 f（n）和 g（n）之间存在多个关系，则仅写最强的一个。 第一行是示例。我们使用 lg 表示以 2 为底的对数。

solution:

	n	$n\lg n$	n^2
$n\lg^2 n$	Ω	Ω	O
$2^{\lg^2 n}$	Ω	Ω	Ω
$\lg(n!)$	Ω	Θ	O
$n^{\lg 3}$	Ω	Ω	O

3. 用动态规划法求如下 0/1 背包问题的最优解：有 5 个物品，其重量分别为（3, 2, 1, 4, 5），价值分别为（25, 20, 15, 40, 50），背包容量为 6. 写出求解过程（设计表格和填写表格）

解：设计一个二维表 V(i, j) 表示将前 i 个物品装进容量为 j 的背包所能获得的最大价值，根据以下递推式填表：

- 若 $w_i > j$, **V(i, j) = V(i − 1, j)**
- 若 $w_i <= j$, V(i, j) = max{ V(i-1, j) , V(i-1, j - wi) + vi}

	j = 0	j = 1	j = 2	j = 3	j = 4	j = 5	j = 6
i = 0	0	0	0	0	0	0	0
i = 1	0	0	0	25	25	25	25
i = 2	0	0	20	25	25	45	45
i = 3	0	15	20	35	40	45	60
i = 4	0	15	20	35	40	55	60
i = 5	0	15	20	35	40	55	65

V(5,6) 即为问题的最优解，最大价值为 65。 经过回溯得到物品组合为第 3 和第 5 个物品装入背包。

4. 考试前，A 老师给同学答疑，同一时间只能给一个同学答疑，有 n 个人等待答疑，已知每个人需要答疑的时间为 ti（0<i<=n），请设计贪心算法安排排队次序，使每个人排队等候时间总和最小。（总分 12 分）

（1）请写出两种以上的贪心策略，比较它们，选出一种用于贪心算法；（4 分）

(2) 写出贪心算法的主要思路；（4 分）

(3) 该算法一定能够保证排队时间总和最小？请简要说明理由。（4 分）

解法：

（1）答疑时间短先安排；答疑时间长先安排。

假设 3 人的答疑时间分别为：3，5，7

按照答疑时间升序择优的等待总时长为：0+3s+8s=11s

按照答疑时间降序择优的等待总时长为：0+7s+12s=19s

综上，采用答疑时间升序择优

（2）本题贪心算法：n 个人时间从小到大排序，就是这 n 个人最佳排队方案。求部分和的和即为所求。

（3）反证法证明：假设有最优解序列：s1,s2…sn,如 s1 不是最小的 Tmin，不妨设 sk=Tmin,将 s1 与 sk 对调，显然，对 sk 之后的人无影响，对 sk 之前的人等待都减少了，(s1-sk)>0,从而新的序列比原最优序列好，这与假设矛盾，故 s1 为最小时间，同理可证 s2…sn 依次最小。

四．算法设计题（本题 30 分，每小题 15 分，选做 2 题）

1. 考虑由以下过程构造的数组A [1 n]：首先对n个不同的元素进行排序，然后将排序后的数组向右旋转k位。 例如对于排序数组[1; 4; 5; 9; 10]，将其向右旋转k = 3位，得到[5; 9; 10; 1; 4]。 试给出一个O（log n）时间算法，该算法查找并返回给定元素x在数组A中的位置，或者如果x不在A中，则返回None。您的算法使用数组A [1 n]，但k值是不知道的。

解决方案1

```
SEARCH(A, i, j, x)
    first ← A[i]
    middle ← A[(i + j)/2]
    last ← A[j]
    if x ∈ {first, middle, last}:
      then
            return the corresponding index
      else :
            if (x < first and x > middle) or (x > first and x > middle):
              then
                    return SEARCH(A, (i + j)/2, j, x)
              else :
                    return SEARCH(A, i, (i + j)/2, x)
```

解决方案II：令k ≥0为元素A旋转的数量。 我们展示了如何识别k的值，之后可以简单地在“左侧部分” A [1 ::: k]和“右侧部分” A [k + 1 ::: n]中执行二叉查找。 为此，请注意，左侧的所有元素都不小于A [1]，而右侧的所有元素都小于A [1]。 通过二分查找，我们查找最小的j> 1，从而使A [j]小于A [1]（如果不存在j，则将j设置为1）。 然后可得k = j-1。

2. 在最长路径问题中，有一个加权有向图G =（V; E; w），一个源s$^{\in}$ V，并且要求找到从s到G中每个顶点的最长简单路径。 对于一般的图形，尚不知道是否存在多项式时间算法来解决此问题。 但是，如果将G限制为无环的，则可以在多项式时间内解决此问题。 给出一种有效的算法，以从加权有向无环图G中的s找到最长的路径，给出其运行时间，并解释为什么当G非无环时您的解决方案不起作用。

解决方案：算法：通过创建一个新图G ‘，将其映射到单源最短路径问题，该图的顶点和边缘与G相同，但权重函数为原始图的负值。
现在，我们可以使用课堂中学习的动态规划中有关DAG的单源最短路径算法，时间性能为Θ（V + E）中的最短路径。 该算法仅以拓扑排序的顺序松弛G ‘的边一次。

（1）初始化，入度为 0 的节点 d 为 0，其他的节点 d 值为 INF。

（2）对 DAG 进行拓扑排序，得到拓扑序列。

（3）按照拓扑序列遍历 DAG 的点，对于每个点 u，遍历其所有的出边<u,v>，如果 d[v] > d[v] + length<u,v>，则更新。

```
DagShortestPath(G, w, s){
        对节点按照拓扑顺序进行排序
        topologically sort the nodes  in G
```

```
        初始化
        for each vertex v  in G{
                dist[v] = INF;
                pre[v] = NULL;
        }
        dist[s] = 0;
        //根据拓扑顺序，遍历顶点 v
        for each v  in G, taken  in topologically sorted order{
                for each edge w[u][v]{
                        if (dist[v] > dist[u] + w[u][v]){
                                pre[v] = dist[u] + w[u][v];
                                pre[v] = u;
                        }
                }
        }
}
```

可以选择在这里使用Bellman Ford，尽管那样会使我们的运行时间欠佳。

时间性能：创建G'是一个简单的过程，只需要Θ（V + E）时间就可以遍历所有边和顶点，从而创建新的图形和权重函数。对边进行拓扑排序需要Θ（V + E），因为拓扑排序是使用DFS算法的修改完成的。 最后，一次松弛所有边仅需Θ（E）时间。 因此，整个运行时为Θ（V + E）。

为什么 G 需要是无环的：如果 G 不是无环的，我们就不能对 DAG 使用单一来源最短路径算法，因为我们将不再拥有 DAG。 但是，即使假设使用的 Bellman Ford 可以处理负环，仍然会遇到麻烦。 需要 G 无环的主要原因是我们正在寻找最长的简单路径（即不重复任何顶点）。 如果我们不限制路线的简单性，那么 G'负权重循环就不会成为问题。