计算机算法设计与分析

083500M01001H-2

2022 期末试卷解答

2022 年 12 月 19 号

Professor: 刘玉贵

学生: 周胤昌

学号: 202228018670052

学院: 网络空间安全学院

所属专业: 网络空间安全

方向: 安全协议理论与技术

一、多选填空 (10 分, 每题 2 分)

1. $f(n) = \log(n!)\,, g(n) = n^{1.05}$, 比较他们的阶:________.
Solution: 根据斯特林公式可知 $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, 因此 $\log(n!) \sim \log\left[\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right] \sim n\log n$.
于是答案为 $f(n) = O(g(n))$.

2. 以比较为基础的检索问题的下界是:________.
Solution: 显然以比较为基础的检索问题/排序问题的下界分别是 $\Omega(\log n)$ 和 $\Theta(\log n)$.

3. 判定问题的蒙特卡洛算法, 当返回 true 时解总是正确的, 但当返回 false 时解可能正确也可能错误, 该算法是 ________.

(1) .偏真的蒙特卡洛算法 (2) .偏假的蒙特卡洛算法
(3) .一致的蒙特卡洛算法 (4) .不一致的蒙特卡洛算法

Solution: 根据偏真的蒙特卡罗算法的定义可知答案选 (1).

4. 下面属于模拟退火算法实现的关键技术问题的有 ______.

(1) .初始温度 (2) .温度下降控制 (3) .邻域定义 (4) .目标函数

Solution: 模拟退火算法实现的关键技术问题有**邻域的定义 (构造)、起始温度的选择、温度下降方法、每一温度的迭代长度以及算法终止规则**. 因此选择 (1), (2), (3).

5. n 皇后问题, 每一个皇后的位置无任何规律, 要求求得一个解即返回. 下面合适的方法有 ______.

(1) .贪心算法 (2) .回溯算法 (3) .分支限界算法 (4) .Las Vegas 算法

Solution: n 皇后问题的求解可以用回溯算法 (第 5-7 章 PPT 内容), 也可用 Las Vegas 算法 (第 9 章 PPT 内容). 因此选择 (2) 和 (4).

二、判断正误 (10 分, 每题 2 分)

1. 0/1 背包问题的贪心算法 (单位价值高优先装入...) 是 ϵ-近似算法. ( )
Solution: **正确** (✓), 题目中叙述的贪心策略所对应的的贪心算法是 **GKK** 算法是 2-近似算法 (即 $\epsilon = 2$).

2. 禁忌搜索中, 禁忌某些对象是为了避免邻域中的不可行解. ( )
Solution: **错误** (✗), 选取禁忌对象是为了引起解的变化, 根本目的在于避开邻域内的局部最优解而不是不可行解.

3. 回溯算法是以深度优先的方式搜索问题的整个解空间树. ( )
Solution: **正确** (✓), 确定了解空间的组织结构后, 回溯法就是从根节点出发, 以深度优先的方式搜索整个解空间树.

4. $\mathcal{NPC}$ 问题也一定是 $\mathcal{NPH}$ 问题. ( )
Solution: **正确** (✓), $\mathcal{NP}$ 问题与 $\mathcal{NPH}$ 的交集就是 $\mathcal{NPC}$ 问题.

5. Las Vegas 算法可能会得到一个不正确的解. ( )
Solution: 错误 (✗), Las Vegas 算法可能会找不到解, 但是如果一旦找到了那就一定是对的 (Las Vegas 算法本身的固有性质).

三、简答题 (25 分)

1. 写出遗传算法的主要步骤. (5 分)
Solution: 遗传算法的主要步骤如下算法1中所示:

Algorithm 1 遗传算法步骤

选择问题的一个编码并初始化种群 (N 个染色体)$\text{pop}(1) := \left\{\text{pop}_j(1) \mid j = 1, 2, \cdots, N\right\}, t := 1$;
对种群 pop(1) 的每个染色体 $\text{pop}_i(1)$ 计算其适应性函数 $f_i = \text{fitness}(\text{pop}_i(1))$;
while 停止规则不满足 **do**
    计算得出概率分布 $p_i = \dfrac{f_i}{\sum_{1 \le j \le N} f_j}$ $(*)$;
    根据概率分布 $(*)$ 从 $\text{pop}(t)$ 中随机选取 N 个染色体并形成种群

$$\text{newpop}(t+1) := \left\{\text{pop}_j(t) \mid j = 1, 2, \cdots, N\right\}$$

    通过交叉 (交叉概率为 P_c) 得到一个有 N 个染色体的种群 $\text{crosspop}(t+1)$;
    以较小的概率 p, 使得染色体的基因发生变异, 形成种群 $\text{mutpop}(t+1)$;
    $t := t + 1$, 诞生新种群 $\text{pop}(t) := \text{mutpop}(t)$;
    对种群 $\text{pop}(t)$ 的每个染色体 $\text{pop}_i(t)$ 计算其适应性函数 $f_i = \text{fitness}(\text{pop}_i(t))$;
end while

2. 基于贪心规则写一个近似算法, 求多机调度问题的一个上限估计. 该算法近似比是多少 (不要求证明)? (5 分)

Solution: 贪心算法 GMPS 为: 按输入的顺序分配作业, 把每一项作业分配给当前负载最小的机器. 并且 GMPS 算法的近似比为 $\epsilon = 2$. 也可以有 DGMPS 算法: 将任务按处理时间从大到小排序, 然后按 GMPS 算法进行调度. 该算法的近似比为 $\epsilon = 3/2$.

3. 已知带权集合的划分问题是 $\mathcal{NPC}$ 问题, 试证明 0/1 背包判定问题是 $\mathcal{NPC}$ 问题. (10 分)
Solution: 证明分成如下两步:

- **证明 $\mathcal{NP}$ 性:** 通过计算向量的内积并逐分量的比较来验证不等式是否成立即可对 0/1 背包判定问题完成判定, 因此 0/1 背包判定问题是 $\mathcal{NP}$ 的;
- **利用已知 $\mathcal{NPC}$ 问题并归约至目标问题:** 考虑特殊的 0/1 背包问题, $\forall x \in X$, 有 $w(x) = p(x)$, 且取 $M = K = \dfrac{1}{2}\sum_{x \in X} w(x)$. 显然该 0/1 背包判定问题回答为"是"当且仅当集合 X 的划分问题回答为"是"(即归约变换的充要性满足). 而划分问题是 $\mathcal{NPC}$ 的, 因此 0/1 背包判定问题也是 $\mathcal{NPC}$ 的.

4. 下述算法是一维最近点对距离的分治算法. 请用迭代法分析该算法的时间复杂度. (5 分)

Algorithm 2 算法 **ClosePair**(S, d)

Input: 实轴上的点集合 S, S 中最近点对的距离 d
Output: **true** or **false**

global S, d;
integer $n := |S|$;
float m, p, q;
if $n < 2$ **then**
  $d := \infty$;
  return false;
end if
$m :=$ **PartSelect**$(S, 1, n, n/2)$ ▷ 调用 **PartSelect** 算法来求 S 中各点坐标的中位数
$S_1 := \{x \in S | x \leq m\}, S_2 := \{x \in S | x > m\}$; ▷ 扫描 S 并对其进行划分 ($O(n)$ 的时间)
ClosePair(S_1, d_1), **ClosePair**(S_2, d_2); ▷ 对规模为 $n/2$ 的子问题分别进行递归求解
$p := \max(S_1), q := \min(S_2), d := \min(d_1, d_2, q - p)$; ▷ 求解出最近点对的距离
return true;
end {ClosePair}

Solution: 问题划分为 2 个规模为 $n/2$ 的子问题, 调用 **PartSelect** 算法需要耗时 $O(n)$, 扫描并划分集合 S 也需要耗时 $O(n)$. 因此 $T(n) = 2T(n/2) + cn$. 不妨设 $n = 2^k$(否则 $\exists k \in \mathbb{N}^*$, 使得 $2^k \leq n < 2^{k+1}$). 于是有如下:

$$
\begin{aligned}
T\left(2^k\right) &= 2^1 T\left(2^{k-1}\right) + c \cdot 2^k \\
&= 2^2 T\left(2^{k-2}\right) + 2c \cdot 2^k \\
&= 2^3 T\left(2^{k-3}\right) + 3c \cdot 2^k \\
&\vdots \\
&= 2^k T(1) + kc \cdot 2^k
\end{aligned}
$$

不妨设 $T(1) = 0$, 因此 $T(n) = \Theta(n\log n)$.

四、算法设计题 (55 分)

1. 设有一条边远山区的道路 AB, 沿着道路 AB 分布着 n 所房子. 这些房子到 A 的距离分别是 $d_1, d_2, \cdots, d_n (d_1 < d_2 < \cdots < d_n)$. 为了给所有房子的用户提供移动电话服务, 需要在这条道路上设置一些基站. 为了保证通讯质量, 每所房子应该位于距离某个基站的 4km 范围内. 设计一个算法找基站的位置, 并且使得基站的总数最少, 并证明算法的正确性. (20 分)

Solution: 使用贪心法, 令 $a_1, a_2, \cdots$ 表示基站的位置. 贪心策略为: 首先令 $a_1 = d_1 + 4$. 对 $d_2, d_3, \cdots, d_n$ 依次检查, 找到下一个不能被该基站覆盖的房子. 如果 $d_k \leq a_1 + 4$ 但 $d_{k+1} > a_1 + 4$, 那么第 $k+1$ 个房子不能被基站覆盖, 于是取 $a_2 = d_{k+1} + 4$ 作为下一个基站的位置. 照此下去, 直到检查完 d_n 为止. 伪代码见如下算法3:

Algorithm 3 Location 算法

Input: 距离数组 $d[1,\cdots,n]=[d_1,d_2,\cdots,d_n]$, 满足 $d[1]<d[2]<\cdots<d[n]$
Output: 基站位置的数组 a

```
a[1] := d[1] + 4; k := 1;
for j = 2; j <= n; j++ do
    if d[j] > a[k] + 4 then
        a[++k] := d[j] + 4;
    end if
end for
return a;
end {Location}
```

结论: 对任何正整数 k, 存在最优解包含算法前 k 步选的的基站位置.

证明. $k=1$, 存在最优解包含 $a[1]$. 如若不然, 有最优解 OPT, 其第一个位置是 $b[1]$ 且 $b[1]\neq a[1]$, 那么 $d_1-4\leq b[1]<d_1+4=a[1]$. $b[1]$ 覆盖的是距离在 $[d_1,b[1]+4]$ 之间的房子. $a[1]$ 覆盖的是距离在 $[d_1,a[1]+4]$ 的房子. 因为 $b[1]<a[1]$, 且 $b[1]$ 覆盖的房子都在 $a[1]$ 覆盖的区域内, 故用 $a[1]$ 替换 $b[1]$ 得到的仍是最优解;

假设对于 k, 存在最优解 A 包含算法前 k 步选择的基站位置, 即

$$A=\{a[1],a[2],\cdots,a[k]\}\cup B \tag{1}$$

其中 $a[1],a[2],\cdots,a[k]$ 覆盖了距离为 $d_1,d_2,\cdots,d_j$ 的房子. 那么 B 是关于 $L=\left\{d_{j+1},d_{j+2},\cdots,d_n\right\}$ 的最优解. 否则, 存在关于 L 的更优解 B^*, 那么用 B^* 替换 B 就会得到 A^* 且 $|A^*|<|A|$, 这与 A 是最优解相矛盾. 根据归纳假设可得知 L 有一个最优解 $B'=\{a[k+1],\cdots\},|B'|=|B|$. 于是

$$A'=\{a[1],a[2],\cdots,a[k]\}\cup B'=\{a[1],a[2],\cdots,a[k],a[k+1],\cdots\} \tag{2}$$

且 $|A'|=|A|$, 故 A' 也是最优解, 从而命题对于 $k+1$ 也成立. 故根据数学归纳法可知, 对任何正整数 k 命题都成立. □

算法的关键操作是 for 循环, 而循环体内部的操作都是常数时间, 因此算法在最坏情况下的时间复杂度为 $O(n)$.

2. 最大子段和问题: 给定整数序列 $a_1,a_2,\cdots,a_n$, 求该序列形如 $\sum\limits_{k=i}^{j}a_k$ 的子段和的最大值:

$$\max_{1\leq i\leq j\leq n}\sum_{k=i}^{j}a_k$$

设计一个动态规划算法求解最大子段和问题, 并说明递推对象的最优子结构性质, 分析算法的时间复杂度 (20 分).

Solution: 先证明此问题具有最优子结构性质: 依次考虑 $(1\leq i\leq n)$ 以 $a[i]$ **为结尾**的最大子段和 $C[i]$, 然后在这 n 个值当中取最大值即为原问题答案. 假设以 $a[i]$ 为结尾的最大 (和) 子段为 $\{a[k],\cdots,a[i]\}$, 那么 $\{a[k],\cdots,a[i-1]\}$ 一定是以 $a[i-1]$ 为结尾的最大 (和) 子段. 否则若 $\{a[m],\cdots,a[i-1]\}$ 为以 $a[i-1]$ 为结尾的最大 (和) 子段, 那么 $\{a[m],\cdots,a[i-1],a[i]\}$ 就是以 $a[i]$ 为结尾的最大 (和) 子段, 这显然与假设相矛盾, 也就是说该优化函数是满足优化原则的 (即此问题具有最优子结构性质).

现在来推导 $C[i]$ 的递推表达式: 当 $C[i-1]\leq 0$, 说明 $C[i-1]$ 对应的子段对于整体的贡献是没有的, 所以 $C[i]\leftarrow a[i]$; 当 $C[i-1]>0$, 说明 $C[i-1]$ 对应的子段对于整体的是有贡献的, 于是

$C[i] \leftarrow a[i] + C[i-1]$. 两种可能情况 (对应两种决策) 取最大值即可:

$$\begin{cases} C\,[i] = \max\left\{a\,[i]\,, C\,[i-1] + a\,[i]\right\}, 2 \le i \le n \\ C\,[1] = a\,[1] \end{cases}$$

最后返回数组 C 中的最大值 ($\max\limits_{1 \le i \le n} C\,[i]$) 即可. 计算 $C[i]$ 的过程需要消耗 $O(n)$ 的时间, 找出数组最大值也需要 $O(n)$ 的时间 (一次遍历), 所以算法的总时间复杂度为 $T(n) = O(n)$. 并且我们可以给出该算法的 C++ 伪码:

```cpp
#include <bits/stdc++.h>
using namespace std;

int mostvalue(vector<int>& a) { //时间复杂度为 O(n), 空间复杂度为 O(n)
    int n = a.size();
    vector<int> dp(n);
    dp[0] = a[0];
    for(int i = 1; i < n; i++) {
        dp[i] = max(dp[i - 1] + a[i], a[i]);
    }
    int index = max_element(dp.begin(), dp.end()) - dp.begin();
    return dp[index];
}

int mostvalue2(vector<int>& a) {  //时间复杂度为 O(n), 空间复杂度为 O(1)
    int n = a.size();
    int dp = a[0];
    int res = INT_MIN;
    for(int i = 1; i < n; i++) {
        dp = max(dp + a[i], a[i]);  //使用滚动数组思想来优化空间
        res = max(res, dp); //迭代式更新求得最大值
    }
    return res;
}
```

3. 分派问题: 给 n 个人分派 n 件工作, 给第 i 人分派第 j 件工作的成本是 $C(i, j)$, 试用分枝限界算法求成本最小的工作分配方案. (15 分)

Solution: 设 n 个人的集合是 $\{1, 2, \cdots, n\}$, n 项工作的集合是 $\{1, 2, \cdots, n\}$, 每个人恰好 1 项工作. 于是有

$$\text{把工作} j \text{分配给} i \Leftrightarrow x_i = j, \quad i, j = 1, 2, \cdots n$$

设解向量为 $X = \langle x_1, x_2, \cdots, x_n \rangle$, 分配成本为 $C\,(X) = \sum\limits_{i=1}^{n} C\,(i, x_i)$. 搜索空间是排列树. 部分向量 $\langle x_1, x_2, \cdots, x_k \rangle$ 表示已经考虑了人 $1, 2, \cdots, k$ 的工作分配. 节点分支的约束条件为:

$$x_{k+1} \in \{1, 2, \cdots, n\} \setminus \{x_1, x_2, \cdots, x_k\}$$

可以设立代价函数:

$$F\,(x_1, x_2, \cdots, x_k) = \sum_{i=1}^{k} C\,(i, x_i) + \sum_{i=k+1}^{n} \min\left\{C\,(i, t) : t \in \{1, 2, \cdots, n\} \setminus \{x_1, x_2, \cdots, x_k\}\right\}$$

界 B 是已得到的最好可行解的分配成本. 如果代价函数大于界, 则剪枝并回溯. 可用优先队列分支限界算法, 以 F 最小优先扩展. 根节点有 n 个儿子 $x_1 = 1, 2, \cdots n$, 儿子节点有 $n-1$ 个儿子 $x_2 = 2, 3, \cdots n$; $x_2 = 1, 3, 4, \cdots n; \cdots$. 算法描述如下, 其时间复杂度为 $O(n \cdot n!)$.

```
Node{
    int Path[n];
    int work[n];
    int T[k];
    int Time;
    int length;
}
Proc BestDispatch(int n, int k, int t[]){
    Node Boot, X, P, result;
    int f;
    f = n * max(t[]);
    Boot.T[n] = {0};
    Boot.Time = 0;
    Boot.Path[n] = {0};
    Boot.length=0;
    AddHeap(Boot);
    while (!Heap.empty()) do {
        P = DeleteMinHeap();
        for i = 1 to n do {
            if(work[i] == 0) {
                X = Newnode(P.Path[], P.T[], P.length + 1);
                work[i] = 1;
            }
            X.Path[X.length] = i;
            X.T[i] = X.T[i] + t[X.length];
            X.Time = max(X.T[]);
            if X.length == n then {
                if X.Time < f then {
                    f = X.Time;
                    result = X;
                }
            }
            else {
                if X.Time < f then {
                    AddHeap(X);
                }
            }
        }
    }
}
end {BestDispatch}
```

中国科学院大学
University of Chinese Academy of Sciences