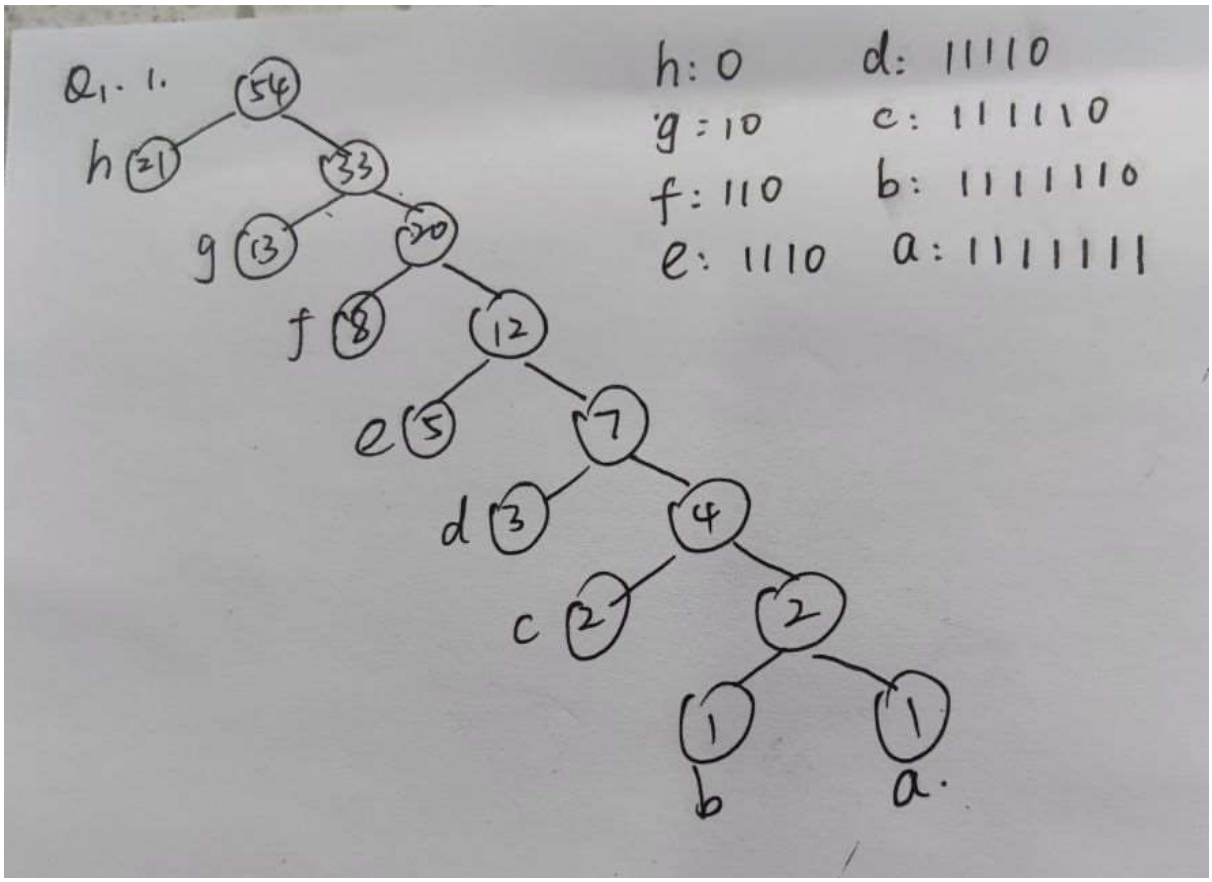
HW5

- 姓名：吴欣怡
- 学号：PB21051111

Q1

(1)



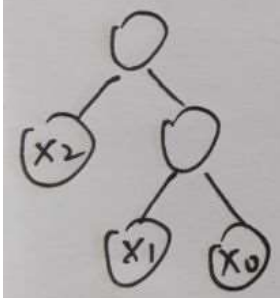
(2)

2. n个字符

首先设定如果有2个权重相同且最小的结点，则先取的结点放在右子树.

设第i个字符为x_i。由于每次组合2个子节点时总是选择当前权重(之和)最小的2个点。

先处理有3个字符的情况：x_0:11、x_1:10 x_2:0



下面证明：当已知前k个字符处理后的最优前缀编码为

$$\begin{cases} x_0: \overbrace{1\cdots1}^{(k-1)个1} \\ x_i: \overbrace{1\cdots10}^{(k-i+1)个1} & i=1,2,\cdots(k-2) \\ x_{k-1}: 0 \end{cases}$$

时

加入频率为F_{k+1}的第(k+1)个字符x_k后，新的最优前缀编码为：

$$\begin{cases} x_0: \overbrace{1\cdots1}^{k个1} \\ x_i: \overbrace{1\cdots10}^{(k-i+2)个1} & 1\le i\le k-1 \\ x_k: 0 \end{cases}$$

由于 $\sum_{j=0}$

显然 $\sum_{i=1}^{k-1} F_i > F_k = F_{k-1} + F_{k-2}$

$\sum_{i=1}^{k-1} F_i < \sum_{i=1}^{k-1} F_i + F_2 - F_1$

$= F_2 + F_1 + F_2 + \cdots + F_{k-1}$

$= F_{k-2} + F_{k-3} + F_{k-2} + F_{k-1}$

$= F_{k-1} + F_{k-2} + F_{k-1}$

$= F_k + F_{k-1} = F_{k+1}$

所以

由于 $F_{k+1} > F_1 + F_2 + \cdots + F_{k-1} > F_k$

含 $(k+1)$ 个这样字符的哈夫曼树

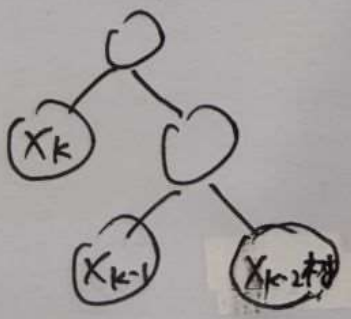
相当于在 $(k-1)$ 个这样字符的哈夫曼树

再加入 X_{k-1}、X_k 的情况.

由 $F_{k+1} > F_1 + \cdots + F_{k-1}$ 可知,

加入 X_{k-1} 后应该由 (X_{k-1}) 和原哈夫曼树

的根结点组合,再着的父结点再与 (X_k) 组合。



则有此树中 $X_0 \sim X_{k-2}$

均要在 X_{k-2} 树中最优

前缀编码的基础上

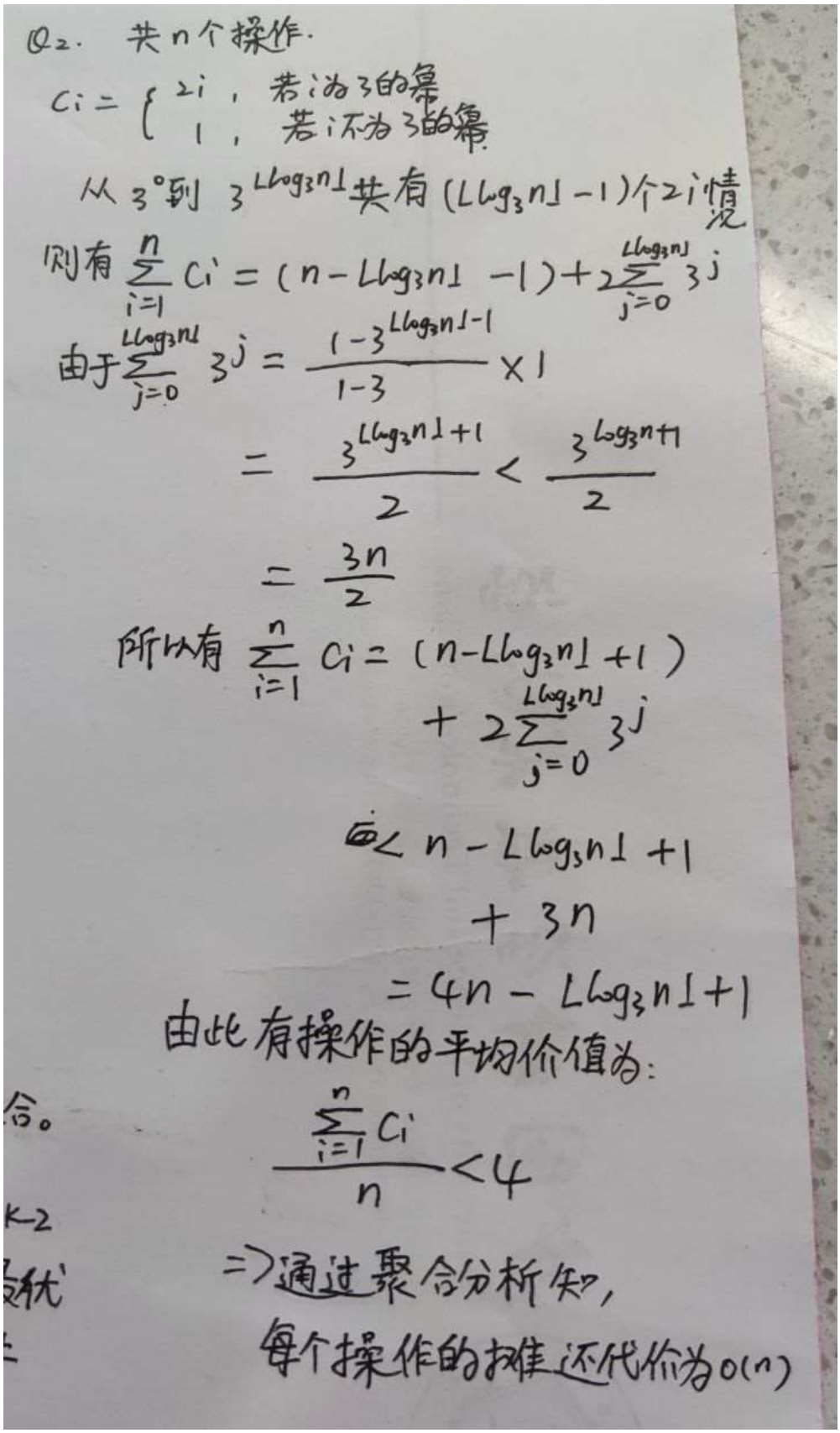
前缀加 2 个 1

X_{k-1}: 10　X_k: 0　满足通式:

X_0: $\overbrace{1\cdots}^{k\text{个}1}$

X_i: $\overbrace{1\cdots}^{(k-i+2)\text{个}1}0$　$1 \le i \le k$

Q2



Q3

不妨设所有256个字符按照出现频率从小到大排列，频率分别为P_1、P_2、……、P_{256}，且满足$P_1 \leq P_2 \leq ... \leq P_{256}$
由已知得到$P_{256} < 2 * P_1$。
那么$128.5 * P_{256} < P_1 + P_2 + ... + P_{256} = 1 \leq 256 * P_{256}$
得到$1/256 \leq P_{256} < 1/128.5$
当采用8位固定长度编码时，显然总代价为8*1=8

由于构建哈夫曼树的过程中，总是合并两个当前最小节点的概论并作为它们父节点的新概率，则有：（除最后一次合并外）每次合并后父节点的频率一定大于所有当前剩下的未合并原始节点的频率。则每次合并过程取到的2个节点都会取到剩余原始节点中频率最小的两个。哈夫曼树的结构形如第一题第一问的情况。P_i节点的对应的代价为（257-i）。
那么总的代价为：
$\sum_{i=1}^{256} P_i * (257-i) > P_{256} + 1/2 * (2+3+...+256) * P_{256} = 16448.5 * P_{256} > 16448.5 * 1/256 = 64.25 > 8$
由此证明得到哈夫曼树相比于8位固定长度编码，代价并没有减小。

Q4

对于由n个叶子的编码树：
当n=1时，命题显然成立；
当n=2时，叶子节点x、y有相同的父节点z
编码树的代价为：$x * 1 + y * 1$
内部节点z的2孩子之和为$x * 1 + y * 1$，两者相等——n=2时命题成立。
由数学归纳法：设（n-1）片叶子时命题成立，下面证明对n片叶子时命题也成立，其中n>2。
用cost(T)表示树T的总代价，P(i)表示节点i的频率，h(i)表示节点i的深度（代价）。用leaves(T)、inner(T)分别表示树T的叶子节点集合、内部节点集合。
令T为一颗n叶子节点编码树，T_1为T删去最底层的两个互为兄弟节点、具有相同父节点q的叶子节点c_1和c_2得到的新编码树。此时T_1为一颗有(n-1)个叶子节点的编码树。易知$P(q) = P(c_1) + P(c_2)$
由(n-1)个叶节点的树中命题成立可知：
$cost(T_1) = \sum_{i \in leaves(T_1)} P(i)h(i)$
$= \sum_{k \in inner(T_1)} [P(k->child1) + P(k->child2)]$
T相比T_1少了叶节点q，多了叶节点c_1和c_2;
T相比T_1多了内部节点q。
则有$cost(T) = \sum_{i \in leaves(T_1)} P(i)h(i) + P(c_1)h(c_1) + P(c_2)h(c_2) - P(q)h(q)$
$= \sum_{i \in leaves(T_1)} P(i)h(i) + [P(c_1) + P(c_2)] * [h(q) + 1] - [P(c_1) + P(c_2)] * [h(q)]$
$= \sum_{k \in inner(T_1)} [P(k->child1) + P(k->child2)] + P(c_1) + P(c_2)$
$= \sum_{k \in inner(T_1)} [P(k->child1) + P(k->child2)] + P(q)$
$= \sum_{k \in inner(T)} [P(k->child1) + P(k->child2)]$
又根据n=2时命题成立，得证。

Q5

(1)

首先已知：在最优解中，任意硬币数量必然不超过2枚。(其中5角、1角、5分、1分不超过1个)
因为$n < 50 * 2$——5角不能超过两枚
$50 + 10 = 20 * 3$——1个50和1个10可以代替3个20
$20 = 10 * 2$——1个20可以代替1个10
$10 = 5 * 2$——1个10可以代替1个5
$5 + 1 = 2 * 3$——1个5和1个1可以代替3个2
$2 = 1 * 2$——1个2可以代替2个1
1）证明对于$n < 10$的情况，贪心算法可以得到最优解：
n=1，2，5显然；
其它的最优解：$3 = 2 + 1$，$4 = 2 * 2$，$6 = 5 + 1$，$7 = 5 + 2$，$8 = 5 + 2 + 1$，$9 = 5 + 2 * 2$，都满足贪心算法
2）对于$10 \leq n < 20$,贪心算法可以得到最优解。若贪心不能有最优解，则最优解不应取面值1角的硬币
但由于最优解中最多有1个5分、2个2分、1个一分，达不到n的值（当n=10时可以达到，但是可以直接替换为一个1角），所以非贪心不可能得到最优值，对于$10 \leq n < 20$的情况必须取一个1角硬币，这之后就是解决n=n-10的问题，前面也证明了此时贪心能够得到最优。
3）对于$20 \leq n < 50$,贪心算法可以得到最优解。若贪心不能有最优解，则最优解不应取面值2角的硬币，而由于最优解中最多有1个10分，1个5分、2个2分、1个一分，达不到n的值（当n=20时可以达到，但是可以直接替换为一个2角），所以非贪心不可能得到最优值，对于$10 \leq n < 20$的情况必须取一个2角硬币，这之后就是解决n<20的子问题，前面也证明了此时贪心能够得到最优。
4）对于$50 \leq n \leq 99$,贪心算法可以得到最优解。易知这个问题可以分解成n=n1+n2的问题，其中n1为一个个位数为0的数，也可以理解为$N3 * 10$（其中$4 < N3 < 9$，$0 \leq < N2 < 10$的数。这样分解的原因是因为n的任何解法中，个位数部分肯定无法用1角、2角或者5角解决，只能用1、2、5分解决，由于n1可以理解为n3*10，5角、2角、1角分别可以理解为*5*10、*2*10、*1*10，所以用1角、2角、5角处理n1与前面的情况1完全相同，可以用贪心处理。而n3也和情况1相同，可以用贪心处理。
综上贪心能够得到最优。

(2)

首先已知：在最优解中，使用的面值为c^i的硬币数量必然不超过（c-1）枚。
因为若使用大于等于c枚,则完全可以用1枚$c^{(i+1)}$硬币代替c枚c^i，总钱数完全相同，但硬币数量减少。
下面来证明，非贪心的算法得不到最优解：（假设首选元素不是贪心选择所要的元素,证明将首元素替换成贪心选择所需元素,依然得到最优解）
令j=max{$0 \leq i \leq k : c^i \leq$ n}，贪心算法下肯定会选择c^j硬币，而非贪心算法则不会选择c^j硬币，只会从c^0、c^1、.....、c^{j-1}中选择。设a^i表示此非贪心算法选择的c^i硬币数。则有：非贪心算法中$\sum_{i=1}^{j-1} a^i * c^i = n \geq c^j$
但是已知$(c-1) * \sum_{i=1}^{j-1} c^i = c^j - 1 < c^j$
若要大于c^j则会和前面证明的最优解中c^i硬币最多只能去(c-1)个矛盾。由此得到非贪心算法不能得到最优解。也就是每次面临j=max{$0 \leq i \leq k : c^i \leq$ n}的（子）问题，最优解中都应该选择c^j硬币，所以这样一次分解和递归下去可以知道，贪心算法可以获得最优解。

(3)

设有4种不同的硬币：1分、3分、4分、5分，如果要解决对12分的找零问题：
按照贪心算法，会选择5分*2+1分*2 共4枚硬币
而最优解为4分*3 共3枚硬币
而由于这组硬币中有1分硬币，所以对于每个n都存在找零方案。