作业反馈3-2

TC第16.1节练习2、3

TC第16.2节练习1、2

TC第16.3节练习2、5、**8**

TC第16章问题1

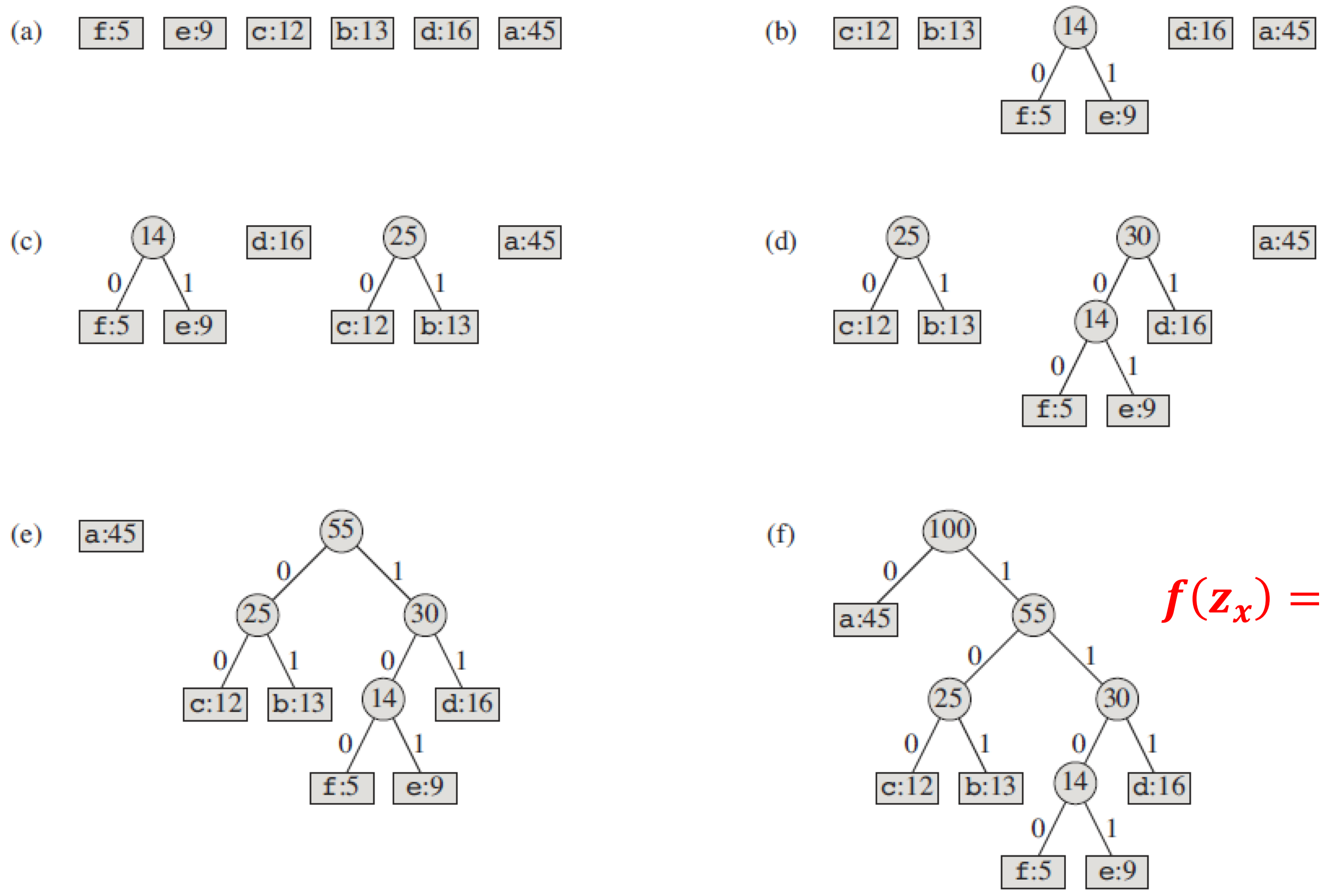
TC第17.1节练习3

TC第17.2节练习2

TC第17.4节练习1

16.3-8

Suppose that a data file contains a sequence of 8-bit characters such that all 256 characters are about equally common: the maximum character frequency is less than twice the minimum character frequency. Prove that Huffman coding in this case is no more efficient than using an ordinary 8-bit fixed-length code.



提示：
合并过程？
假定c1,c2,频率最低；那在合并c1+c2=z之后；要等到所有其他cx，(x!=1,2) 分别合并之后，才可能进一步处理z。

$$f(c_x) \le f(c_1) + f(c_2) \le 2f(c_x)$$

$$f(z_x) = f(c_{x_1}) + f(c_{x_2}) \le 2\left(f(c_{y_1}) + f(c_{y_2})\right) = f(z_y)$$

Def: two chars are about equally common $\leftrightarrow$ the maximum character frequency is less than twice the minimum character frequency.
Mathematical Induction: Let T denote the tree constructed by Huffman coding, T' denote the tree with ordinary 8-bit fixed-length code.

1. When there's only $2^0 = 1$ char, certainly $B(T) \geq B(T')$
2. Supposing for $n = 0, 1, 2, \ldots k-1$, where all 2^n chars are equally common, $B(T) \geq B(T')$
3. When $n = k$, where all 2^n chars are equally common, for convenience, we can first sort chars first by frequency . Then we make tree by pairs, that is, $\{z_1 = a_1$ with $a_2\}$, $\{z_2 = a_3$ with $a_4\}$, $\{z_i = a_{2*i-1}$ with $a_{2*i}\}$, so on and so forth. Since $frec(2*i-1) + frec(2*i) \geq 2*min(frec) > max(frec)$, we guarantee that the construction steps follows the huffman coding principle. Also, $z_0.frec * 2 \geq 4*min(frec) > 2*max(frec) \geq z_{n-1}.frec$. Therefore the problem is turned into constructing a tree with 2^{k-1} nodes which is solved previously. It's trivial that $B(T) \geq B(T')$.

17.4-1

Suppose that we wish to implement a dynamic, open-address hash table. Why might we consider the table to be full when its load factor reaches some value α that is strictly less than 1? Describe briefly how to make insertion into a dynamic, open-address hash table run in such a way that the expected value of the amortized cost per insertion is $O(1)$. Why is the expected value of the actual cost per insertion not necessarily $O(1)$ for all insertions?

```
TABLE-INSERT(T, x)
if T.size == 0
    allocate T.table with 1 slot
    T.size = 1
if T.num == T.size
    allocate new-table with 2 · T.size slots
    insert all items in T.table into new-table
    free T.table
    T.table = new-table
    T.size = 2 · T.size
insert x into T.table
T.num = T.num + 1
```

```
HASH-INSERT(T, k)
i = 0
repeat
    j = h(k, i)
    if T[j] == NIL
        T[j] = k
        return j
    else i = i + 1
until i == m
error "hash table overflow"
```

Example1

Cost for probe?

?

For example, insertion into a dynamic open-address hash table can be made to run in $\mathbf{O}(1)$ time by expanding when $\alpha \le 0.75$ and contracting when $\alpha \le 0.25$.

$$num_i = num_i + 1$$

If expanding:

$$size_i = 2size_{i-1}$$

$$num_{i-1} = \frac{3}{4}size_{i-1}$$

Define the potential function

$$\Phi_i = \begin{cases} \frac{8}{3}num_i - size_i & \text{table at least half full} \\ \frac{1}{2}size_i - num_i & \text{table is less than half full} \end{cases}$$

So the amortized cost **If expanded**

$$\begin{aligned} \hat{c} &= c_i + \Phi_i - \Phi_{i-1} \\ &= num_i + 1 + \frac{1}{2}size_i - num_i - (\frac{8}{3}num_{i-1} - size_{i-1}) \\ &= 1 = \mathbf{O}(1) \end{aligned}$$

Example2

由 Corollary 11.7 可知, 一次插入最多需要 $1/(1-\alpha)$ 次试探, 因此当 $\alpha \to 1$ 时, 所需的试探次数趋于无穷大, 因此必须使得 α 严格小于 1.

可以如此设定: 当 $\alpha = \frac{3}{4}$ 时, 认为 hash 表已经满了, 创建一个 2 倍大小 ($size$) 的 hash 表, 插入原表中所有的条目, 再插入所需插入的条目. 仍然定义势能 $\Phi(T) = \frac{8}{3} \cdot T.num - T.size$. 由于 $\alpha \geq \frac{3}{8}$, Φ 始终非负, 因此基于 Φ 的均摊开销是真实开销的一个上界.

先考虑 $\alpha < \frac{3}{4}$ 的情况, 即不触发表的扩张. 此时

$$
\begin{aligned}
\widehat{c_i} &= c_i + \Phi_i - \Phi_{i-1} \\
&\leq \frac{1}{1-\alpha_{i-1}} + (\frac{8}{3} \cdot num_i - size_i) - (\frac{8}{3} \cdot (num_i - 1) - size_i) \\
&< 4 + \frac{8}{3} \\
&= \frac{20}{3}
\end{aligned}
$$

再考虑 $\alpha = \frac{3}{4}$ 的情况, 即触发表的扩张. 此时

$$
\begin{aligned}
\widehat{c_i} &= c_i + \Phi - \Phi_{i-1} \\
&\leq size_i \cdot \int_0^{\frac{3}{8}} \frac{1}{1-\alpha} \mathrm{d}\alpha + \frac{1}{1-\alpha_i} + (\frac{8}{3} \cdot (num_{i-1} + 1) - \frac{8}{3} \cdot num_{i-1}) \\
&\quad - (\frac{8}{3} \cdot num_{i-1} - \frac{4}{3} \cdot num_{i-1}) \\
&= \left(\ln\left(\frac{8}{5}\right) - \frac{1}{2}\right) \cdot size_i + \frac{64}{15} \\
&\leq \frac{64}{15}.
\end{aligned}
$$

综上, 此动态 hash 表的插入操作均摊开销为 $O(1)$.

少了insert本身的开销

Example3

少了insert本身的开销

If $\alpha = 1$, the table is full. Thus there is no room for insertion.

We double the size of the hash table when $\alpha > \frac{3}{4}$.

Suppose for each insertion we need to pay $\frac{1}{1-\alpha}$ dollars on average and resizing a hash table with k elements means to insert all the existing elements into a new hash table.

Suppose there are n insertions to be performed. For each insertion we charge 12 dollars. Thus we will save at least 8 dollars per operation if it does not trigger resizing. When the resizing of the hash table is triggered, we need to pay at most $\boxed{\frac{1}{1-\alpha}\frac{3m}{4} = 3m,}$ where m stands for the size of the table. Assuming that after the last resizing our balance is nonnegative, then there is at least $8(\frac{3m}{4} - \frac{3m}{8}) = 3m$ dollars before the resizing, which is sufficient for the payment. Prove inductively and we will arrive at the conclusion that the balance is always nonnegative. Thus every operation runs in $\Theta(1)$ amortized time.

16-1 Coin changing

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

a. Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

证明：由于大的硬币可以代替若干枚小面值的硬币，因此从大的开始找是最优解：

1、n<5 时，我们只能找 1 美分的硬币，此时找 1 美分为最优解

2、5<=n<10，假设最优解不包括 5 美分，我们只能用 1 美分的钱找，如果这样，找的 1 美分的钱>=5 枚，我们可以用 1 枚 5 美分代替 5 枚 1 美分，这样使得解更优，与最优解不包括 5 美分矛盾，因此 5<=n<10 时，最优解包含 5 美分

3、10<=n<25，假设最优解不包括 10 美分，我们只能用 5 美分和 1 美分的钱找，然而 2 枚 5 美分或者 1 枚 5 美分+5 枚 1 美分或者 10 枚 1 美分可以兑换 1 枚 10 美分，这样使得解更优，与最优解不包括 10 美分矛盾，因此最优解包括 10 美分，而且兑换足够多的 10 美分后，我们可以转换成问题 1,2 进行求解

4、n>=25 时，假设最优解不包括 25 美分，我们只能用 10、5、1 美分找钱，然而 25 美分可由 2 枚 10 美分和 1 枚 5 美分甚至更多的钱兑换，这样包含 25 美分会使得解更优，与最优解不包括 25 美分矛盾，因此最优解包括 10 美分，而且兑换足够多的 25 美分后，我们可以转换成问题 1、2、3 求解。

b. Suppose that the available coins are in the denominations that are powers of c, i.e., the denominations are $c^0, c^1, \ldots, c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.

Determine the largest j that $c^j \leq n$, give one coin of denomination c^j, and then recursively solve the subproblem of making change for $n - c^j$ cents.

To prove that the greedy algorithm produces an optimal solution, first we claim that, for $i = 0, 1, .., k-1$, the number of coins of denomination c^i used in an optimal solution for n cents is less than c. If not, we can improve the solution by using one more coin of denomination $c^i + 1$ and c fewer coins of denomination c^i.

不够严谨，确实都可以转换为这样

Let $j = \arg\max_{0 \leq i \leq k}(c^i \leq n)$, so that the greedy solution uses at least one coin of denomination c^j; a nongreedy solution must use no coins of denomination c^j or higher. Thus for the non-greedy solution, we have, $\sum_{i=0}^{j-1} ai * c^i = n \geq c^j$.

However we have $a_i \leq c - 1$, so

$$\begin{aligned} \sum_{i=0}^{j-1} a_i * c^i &\leq \sum_{i=0}^{j-1} (c-1) * c^i \\ &= (c-1) \sum_{i=0}^{j-1} c^i \\ &= (c-1)(c^j - 1)/(c-1) < c^j \end{aligned}$$

This contradiction shows the non-greedy solution is not optimal. And greedy algorithm has the running time $\mathbf{O}(\mathrm{k})$.