EC 504 Midterm Practice – Fall 2023

Of course some details here are way more than anyone is expected to do on the exam. They are include here for instruction.

Instructions: Put your name and BU ID on each page. This exam is closed book and no notes – except for 2 pages of personal notes to be passed in the end. No use of laptops or internet. You have one hour and fifty minutes. Do the easy ones first.

1. (20 pts) Answer True or False to each of the questions below. Each question is worth 2 points. Answer true only if it is true as stated, with no additional assumptions. No explanation is needed, but any explanation is likely to earn partial credit, and no explanation will not earn any credit if the answer is wrong.

   (a) $N^2 e^{-1/\ln(N)} \in \Theta(N e^{\ln(N)})$.

   Solution: True: Since $Ne^{\ln(N)} = N^2$ and with $1/ln(N) \to 0$ the limt $N^2 e^{-1/\ln(N)} \to N^2$ so both are in $\Theta(N^2)$

   (b) If $f_i(n) \in O(n^2)$, then $\sum_{i=1}^{n} f_i(n) \in O(n^3)$

   Solution: True. There are n terms, each of which is $O(n^2)$, so you can bound this by a constant times n^3, making it $O(n^3)$.

   (c) The following tree is an AVL binary search tree – explain.

```
      11
     /    \
    9      23
   / \     /  \
  8   12  16   50
 / \     /       \
1   10  10        80
```

   Solution: False. The key 12 must be the right of 11 and the second 10 to the left of 11.

   (d) The second smallest element in a binary min-heap with all elements with distinct values will always be a child of the root.

   Solution: True. The children of the root must be smaller than any of their children.

   (e) The set $\Theta(f(n))$ is identical to the set $O(f(n)) \cap \Omega(f(n))$ (Note: $\cap$ means intersection of two sets.)

   Solution: True. The intersection $\Theta((f(n)) \cap \Omega(f(n))$ in the Venn diagram is defined to be the set $\Theta(f(n)$

(f) It is possible to formulate Quick Sort to be worst case $O(N \log N)$ including the cost of picking a suitable pivot.

Solution: True. You pick the pivot to be them medium using the $()(N)$ algorirthm that sorts $N/5$ columns recursive find the medium for $N/5$ in the middle row.

(g) Consider the array [2, 20, 3, 21, 90, 4,23, 22]. This array has elements in order of a min-heap.

Solution: True. All path for the root are in decreasing order.

```
             2
          /     \
         20      3
        / \     /  \
       21  90  4    23
      /
     22
```

(h) A min-Heap (or priority queue) is a null tree or a root node with a minimum value plus a left and a right min-Heap subtrees.

Solution: True. This is the correct recursive defintion of a min-Heap.

(i) The maximum subsequence sum algorithm finds i and j that maximize $\sum_{k=i}^{j} a[k]$ for an array $a[N]$ of positive and negative integers. The fastest one is a recursive algorithm with complexity $\Theta(N \log N)$.

Solution: False. As I demonstrated in class, the fastest one, which is a sort of *investors scan*, is $\Theta(N)$ – suprising!

(j) $N^N \in O(\ e^{\ln(N!)}\ )$

Solution: False The Sterling's approxmation of the factorial: $N! = \sqrt{2\pi N} N^N e^{-N}(1 + O(1/N))$ and therefore $e^{\ln(N!)} \sim \sqrt{2\pi N} N^N e^{-N}$ which is slower than N^N by a factor of $\sqrt{2\pi N} e^{-N} \to 0$. Thus $N! = O(N^N)$ but not the reverse.

2. (10 pts) Consider the binary tree illustrated below.

```
           *
       /       \
      +          -
     / \        / \
     *  8      -   10
   /  \       / \
     +     2     8    *
    /  \             / \
   6     7          2   3
```

(a) List the node values in the order in which they would be found in an in-order traversal.

(b) List nodes in pre-order.

(c) List the nodes in post=order.

(d) Which order is uses on stack to do the arithmetic? . (HINT the stack order is like depth first search on this tree!) Evaluate the arithmetic using stack

Solution:

In-order

$$6 \quad + \quad 7 \quad * \quad 2 \quad + \quad 8 \quad * \quad 8 \quad - \quad 2 \quad * \quad 3 \quad - \quad 10 \tag{1}$$

pre-order

$$* \quad + \quad * \quad + \quad 6 \quad 7 \quad 2 \quad 8 \quad - \quad - \quad 8 \quad * \quad 2 \quad 3 \quad 10 \tag{2}$$

post-order

$$6 \quad 7 \quad + \quad 2 \quad * \quad 8 \quad + \quad 8 \quad 2 \quad 3 \quad * \quad - \quad - \quad 10 \quad - \quad * \tag{3}$$

Evaluation of arithmatic. Easiest way is bottom up on the tree (e.g. in-order) to get Result is -272. The stack has two solutions.

You get -272 if you remeber that we pushed left and then right so for a minus sign is pop y pop x $\implies x - y$.

You could assume pop x pop y - $\implies y - x$ which gives 408. I accept this rule altough it is inconsistent with the definition of post-order. This could be corrected by exchangeing left and righ in post order. Hard becuase Minus sign hard because ther are not symmetric.

3. (20 pts) Consider the following min-heap.

```
                1
           /           \
          10            2
       /    \        /    \
     11      12     17      3
    /  \    /  \   /  \    / \
  15   18 201  13 155  20 4   5
 / \   /
17 16 21
```

(a) Show the min-heap which results after inserting the element 7 in the heap. (Indicate the sequence of steps with arrows.) Then in this new heap show the steps required to delete element 1 and then delete 11. Draw the final min-heap.

Solution:

The insertion starts with putting 7 in the next empty leaf.

```
                1
           /           \
          10            2
       /    \        /    \
     11      12     17      3
    /  \    /  \   /  \    / \
  15   18 201  13 155  20 4   5
 / \   /\
17 16 21 7
```

The is unique path from the leaf to the root. To put in sorted order this just a trivial bubble sort: 7 is push up swaping with 18 and then swaping with 11 and then swaping with 10 to get

```
                1
           /           \
          7            2
       /    \        /    \
     10    12     17      3
    /  \    /  \   /  \    / \
  15   11 201  13 155  20 4   5
 / \   /\
17 16 21 18
```

(b) Re-arrange the original min-heap (repeated below) into a max-heap by a "bottom up" $O(N)$ algorithm. Describe the steps level by level.

Solution:

Start with next to bottom row of 8

```
                1
           /           \
          10            2
       /    \        /    \
     11      12     17      3
    /  \    /  \   /  \    / \
  17   21 201  13 155  20 4   5   <---- Start with row of 8
 / \   /
15 16 18
```

```
                1
           /           \
          10            2
       /    \        /    \
     21     201     155     5   <---- Next row of 4
    /  \    /  \   /  \    / \
  17   18  12  13 17  20  4   3
 / \   /
15 16 11
```

```
                1
           /           \
         201           155      <---- Next row of 2
       /    \         /    \
     21      13      20     5
    /  \    /  \    /  \    / \
  17   18  12   10 17  2  4   3
 / \   /
15 16 11
```

```
               201                 <---- Final Root node
           /           \
         21          155
       /    \         /    \
     18      13      20     5
    /  \    /  \    /  \    / \
  17    11   12    10 17   2   4    3
 / \    /
15 16  1
```

(c) Using this example for a min-heap with the size $N = 18$ and **a[0] = N**. Describe in a few word an an $\Theta(N \log N)$ algorithm to sort `in place` the array elements $a[1], a[2], \cdots, a[N]$ in **descending order** (i.e. $a[1] > a[2] > \cdots > a[N]$)

Solution: $a[0]$ store the size of heap starting with $a[0] = N$ The algorithm is iterative. You delete the min in $a[1]$ restoring the heap and copying $a[a[0]]] = a[1]$ reducing the size by by 1 $(a[0] = a[0] - 1)$ until the heap is empty $(a[0] = 0)$ Now the array $a[1], a[2], ..a[N]$ is sorted array in descending order. Restorig the heap each time is $(log(N)$ so the alogritym is $O(NlogN)$

(d) Solve the recursion relation for the Heap, $T(H) = 2T(H-1) + c_0H$ to show that "bottom up" scales like $T(H) = \Omega(N)$ (HINT: Use $N \simeq 2^H$)

Solution:

Substitute the guess in the equation that is suggested: $T(H) = c\ 2^H$

$$c\ 2^H \simeq c\ 2\ 2^{H-1} + c_0 H = c\ 2^H + c_0 H \simeq c\ 2^H \tag{4}$$

So the leading behavior is $T[H] = c\ 2^H \in \Omega(H)$.

4. (20 pts) This problem is to construct step by step BST and AVL trees given the following list of $N = 12$ elements.

$$1 \quad 2 \quad 12 \quad 7 \quad 3 \quad 6 \quad 30 \quad 36 \quad 37 \quad 2 \quad 40 \quad 51$$

(a) Insert them sequentially into a BST (Binary Search Tree).

Solution: The BST (depth of a node is the numbe of hop form the root. height is the maximun number of hops to the deapest node.

```
            1    h=7, d = 0
             \
              2     h=6, d = 1
             / \
h=0, d = 2  2    12   h=5, d = 2
               /  \
h=2, d = 3    7   30    h=4, d = 3
             /      \
h= 1,d = 4  3        36     h=3, d = 4
             \         \
h=0, d = 5    6        37     h=2, d = 5
                         \
                          40    h=1, d = 6
                           \
                            51    h=0, d = 7
```

(b) Insert them sequentially into an empty AVL tree, restoring the AVL property after each insertion. Show the AVL tree which results after each insertion and name the type of rotation (RR or LL zig-zig or versus RL or LR zig-zag).

Solution: Final AVL form; See next page for sequence of steps!

```
          7                 h = 3, d = 0
      /       \
     3         30           h = 2 d = 1
    /   \     / \
   2      6  12   37        h = 1 d = 2  (excep 12 is h =0)
  / \    /       /  \
1    2  5       36   40     h = 0 d = 3
```

The sequence of AVL moves are on the next page:

(c) Give the total height $T_H(N)$ and the total depth $T_D(N)$ for both the resulting BST and AVL.

(d) Compare the sum $T_H(N)+T_D(N)$ for the BST and AVL. For each compare these values with HN where H is the height of each tree.

Solution: To part (c) and (d) above:
BST:

$$T_H(N) = 1+2+3+4+5+6+7+1+2 = 31$$

and

$$T_D(N) = 1+2+3+4+5+6+7+2+3+4+5 = 42$$

So $T_H(N) + T_D(N) = 73 < HN = 7 * 12 = 84$
AVL:

$$T_H(N) = 3 + 2*2 + 4*1 = 11$$

and

$$T_D(N) = 2*1 + 4*2 + 5*3 = 25$$

So' $T_H(N) + T_D(N) = 36 = HN = 3*12 = 36$
Note a perfect tree of $N = 15$ and height $H = 3$ has

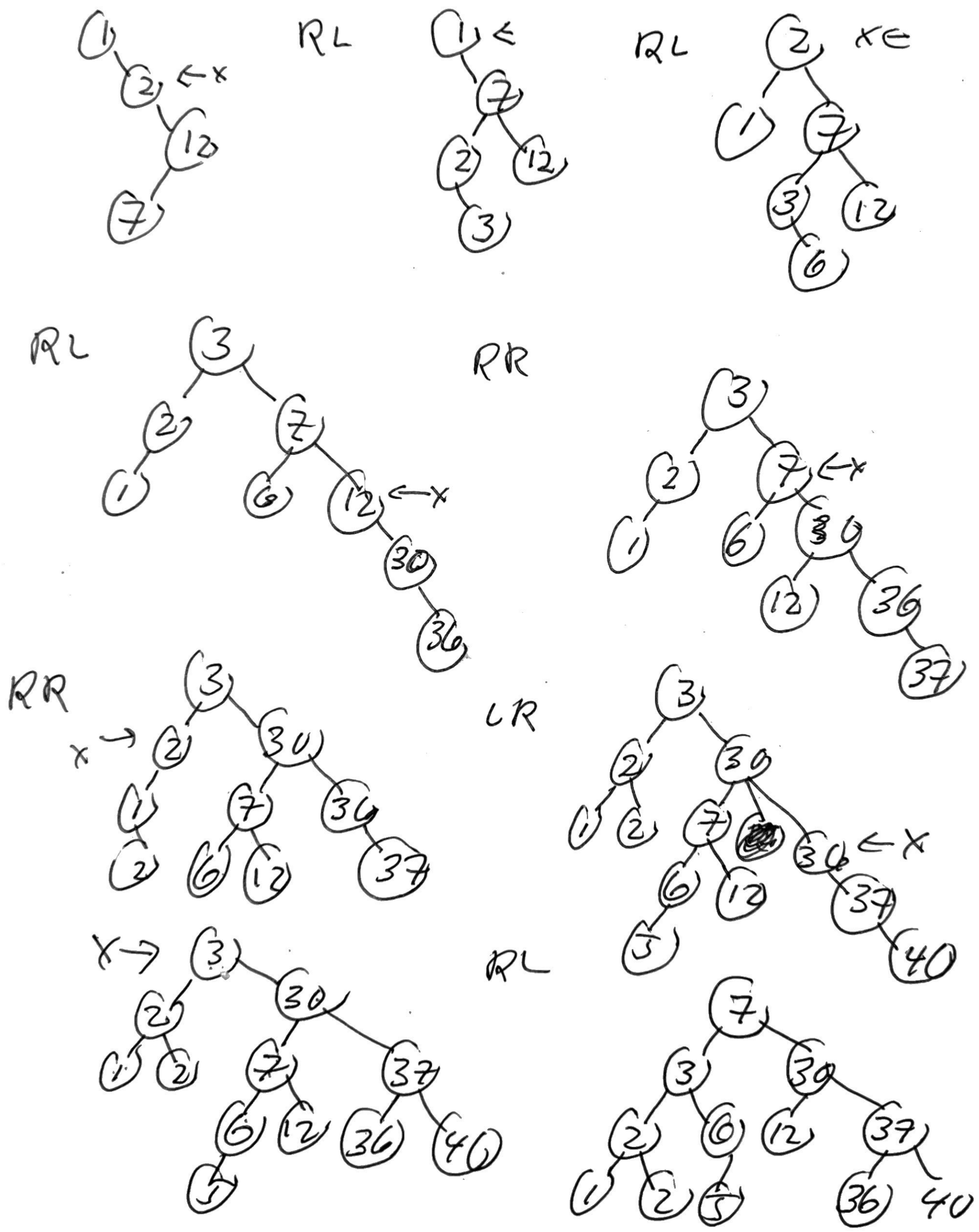
$$T_H(N) = 3 + 2*2 + 4*1 = 11$$

and

$$T_D(N) = 2*1 + 4*2 + 8*3 = 34$$

so $T_H(N) + T_D(N) = 45 = HN = 3*15 = 45$
I thought this is the only one that saturates the upper bound but a counter example is to remove leaf nofrd $NH \to NH - H = (N-1)H$ and that leaf remove d = H from the sum of $T_D(N)$!

AVL: 1 2 12 7 3 6 30 36 37 2 40 5



5. (10 POINTS) Consider the following recursion relation.

$$T(N) = 3T(N/2) + N^k$$

(a) Substitute $T(N) = c_0 N^\gamma$ into the homogeneous equation (i.e. dropping N^k) and determine γ and for what values of k does the exact solution satisfy $T(N) = \Theta(N^\gamma)$.

Solution: The homogeneous equation (i.e. drop the last term has solution by substitution

$$c_0\ N^\gamma = c_0\ 3\ (N/2)^\gamma \implies 1 = 3/2^\gamma \quad \text{or} \quad \gamma = \log(3)/\log(2) \tag{5}$$

Base 2 is nice since $\log_2(2) = 1$ or $\gamma = \log_2(3)$

This is gives a leading solution $T(N) = \Theta(N^\gamma)$ if $k < \gamma$

(b) Now assume that $k = \gamma$ and find the exact solution in the form: $T(N) = c_0 N^\gamma + c_1 N^\gamma \log_2(N)$. (HINT: First show that $c_1 = 1$. Then Determine c_0 in terms of $T(1)$ by setting $N = 1$.)

Solution: Note a homogeneous solution never determines the multiplicative const c_0. For this we must find a fulls solution. The substution gives

$$\begin{aligned} c_0\ N^\gamma + c_1 N^\gamma \log_2(N) &= c_0\ 3\ (N/2)^\gamma + c_1 3(N/2)^\gamma \log_2(N/2) + N^\gamma \\ &= c_0\ 3\ (N/2)^\gamma + c_1 (N)^\gamma [\log_2(N) - \log_2(2)] + N^\gamma \end{aligned} \tag{6}$$

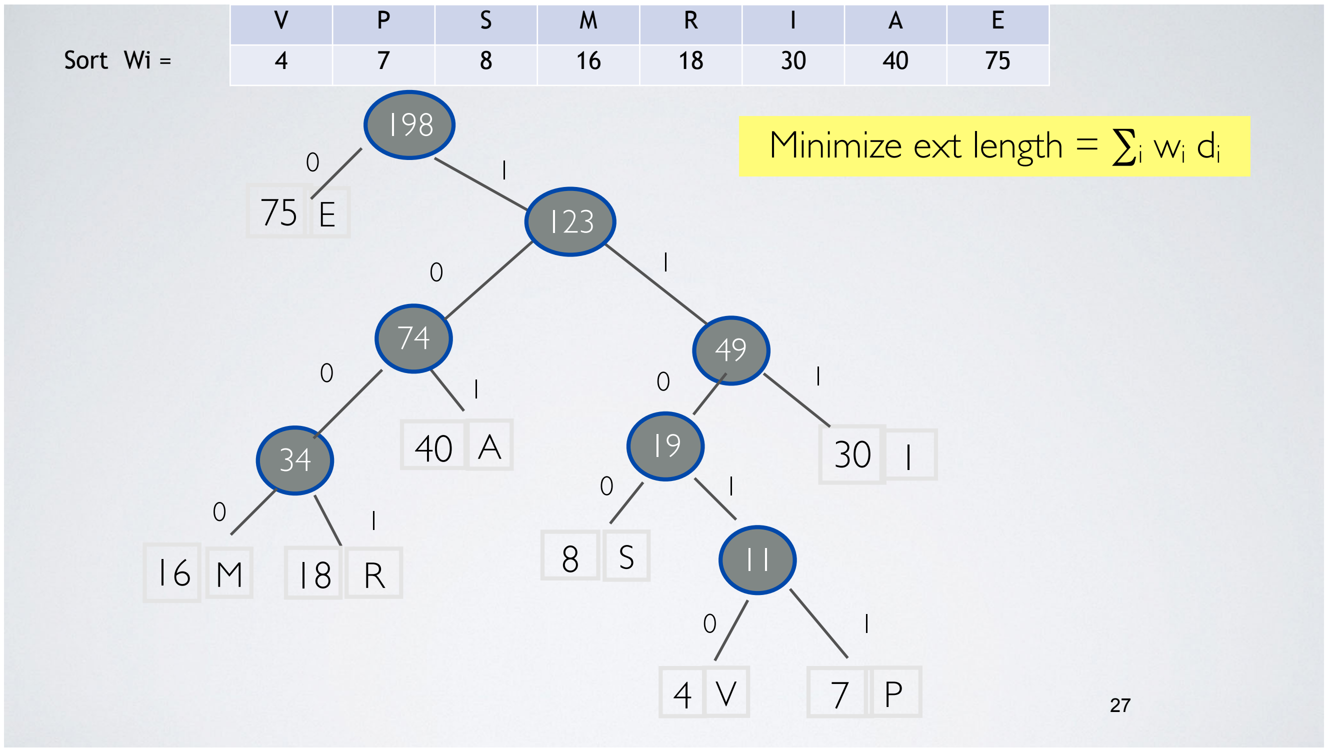
Obviously the first (homegeneous term) still cancels. We could have ignore this piece for now.

But to cancel the additive term we need to use both $3/2^\gamma = 1$ and again $\log_2(2) = 1$, to get $c_1 = 1$. The setting $N = 1$ we finally fix c_0 by $T(1) = c_0\ 1^\gamma + c_1 1^\gamma \log_2(1) = c_0$. Actually you can do this step first if you prefer.

6. (20 pts) You are interested in compression. Given a file with characters, you want to find the binary code which satisfies the prefix property (no conflicts) and which minimizes the number of bits required. As an example, consider an alphabet with 8 symbols, with relative weights (frequency) of appearance in an average text file [1] give below:

alphabet:	V	A	M	P	I	R	E	S
weights:	4	40	16	7	30	18	75	8

(a) Determine the Huffman code by constructing a tree with **minimum external path length**: $\sum_{i=1}^{8} w_i d_i$. (Arrange tree with smaller weights to the left.) **Solution:**



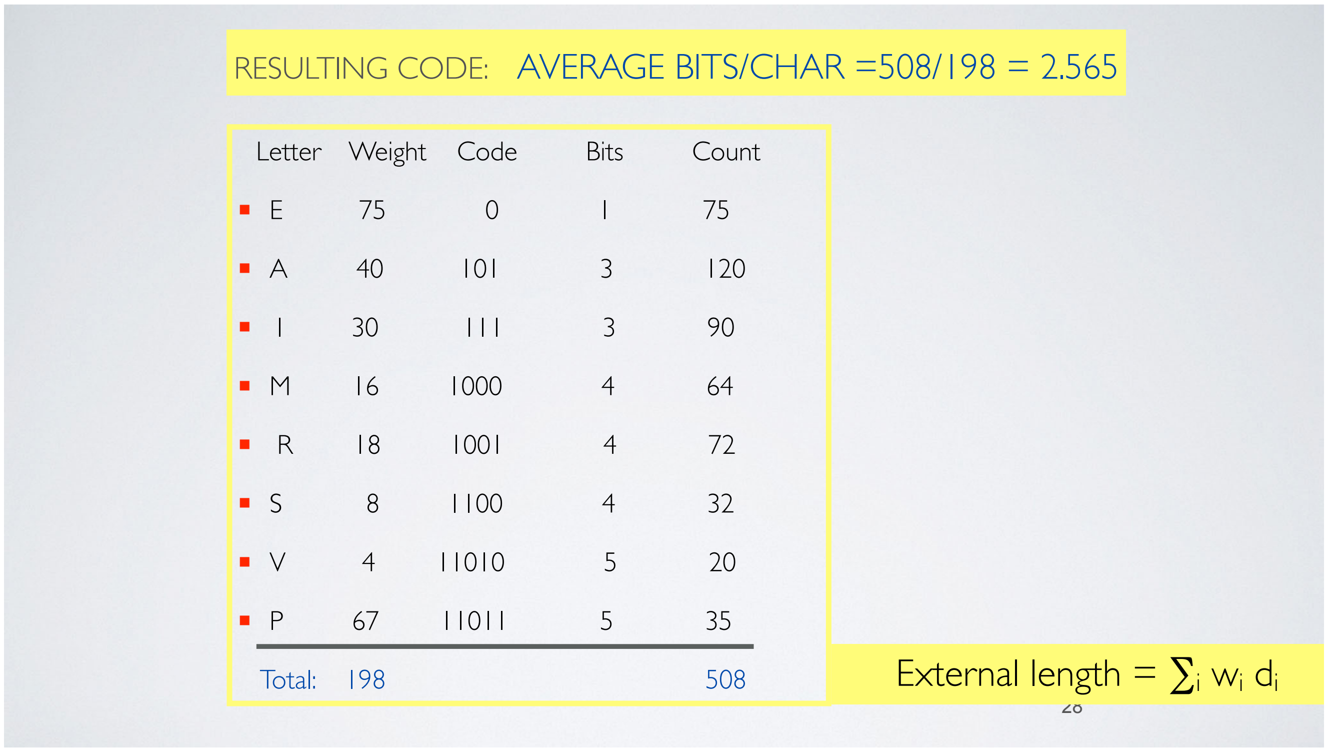
Sort Wi =	V	P	S	M	R	I	A	E
	4	7	8	16	18	30	40	75

Minimize ext length = $\sum_i w_i d_i$

(b) Identity the code for each letter and list the number of bits for each letter and compute the average number of bits per symbol in this code. Is it less than 3? (You can leave the answer as a fraction since getting the decimal value is difficult without a calculator.) **Solution:**

(c) Give an example of weights for these 8 symbols that would saturate 3 bits per letter. What would the Huffman tree look like? Is a Huffman code tree always a full tree? **Solution:**

[1]vampire: Origin mid 18th cent: from French, from Hungarian **vampir**, perhaps form Turkish **uber "witch"**. HAPPY POST HALLOWEEN: In European folklore, a monster with long pointed canine teeth!



RESULTING CODE: AVERAGE BITS/CHAR =508/198 = 2.565

Letter	Weight	Code	Bits	Count
E	75	0	1	75
A	40	101	3	120
I	30	111	3	90
M	16	1000	4	64
R	18	1001	4	72
S	8	1100	4	32
V	4	11010	5	20
P	67	11011	5	35
Total:	198			508

External length = $\sum_i w_i d_i$

It is perfect binary tree. One can have each have weight 10 for example

```
              ( 80 )
          /            \
      (40)              (40)
     /    \            /    \
  (20)     (20)     (20)    (20)
  /   \    /  \    /  \     / \
V     A   M    P  I    R   E   S
```

Actually the weight don't have to exactly equal. Try it with weights:

$$100, 101, 102, 103, 104, 105, 106, 107$$

It is still perfect tree1

7. (10 POINTS EXTRA CREDIT) Given an array of N-distinct positive integers $a[N]$ and a second array of N-distinct positive integers $b[N]$ to construct the sum:

$$\sum_{i=0}^{N-1} a[i] * b[i] \tag{7}$$

   (a) Assume that $a[k]$ is sorted in ascending order. Give an $O(N \log N)$ algorithm that maximizing the sum S by permuting the values of $b[k]$. **Solution:** To maximize the sum you sort the array $b[k]$ into ascending order by any $\Theta(NlogN)$ algorithm such as merg sort. This is an example of every pair $\{i, j\}$ maximizing the scalar product $a[i]b[i] + a[j]b[j]$ since $a[i] < a[j]$ and $b[i] < b[j]$. Any exchang of i, j reduces this contribution the sum.

   (b) Given the result of part (a) give an algorithm that minimizes S in $O(N)$.

   Solution: To minimize the result of part (a) one just reverse the order of $b[k]$ with $N/2$ swaps of $b[i]$ and $b[N-1-i]$ for $i = 0, 1, \cdots N/2$

   (c) If you are allowed to permute independently both $a[k]$ and $b[k]$ how many different solutions will maximize S? **Solution:** There are $N!$ solution because give maximun

$$S_{max} = \sum_{k=0}^{N-1} a[k] * b[k] \tag{8}$$

   as describe in part (a) is unchange by any simultaineous permutation of k ($\pi(k)$

$$S_{max} = \sum_{k=0}^{N-1} a[\pi(k)] * b[\pi(k)] \tag{9}$$