$B = k \geq 3$ We can keep replacing each root of the subtree as follows: the first $\{R, (a_1, a_2, \ldots, a_k)\}$ becomes $\{R, (a_1, R_2)\}$, with subtree $\{R_2, (a_2, R_3)\}$, $\ldots$, $\{R_{k-1}, (a_{k-1}, a_k)\}$. By induction, then, any tree can be replaced by an equivalent binary tree.

While the above shows that any node with $B \geq 2$ can be replaced by a node with binary decisions, we can apply this to all nodes in the expanded tree, and thereby make the entire tree binary.

(b) The number of levels depends on the number of classes. If the number of classes is 2, then the functionally equivalent tree is of course 2 levels in depth. If the number of categories is c, we can create a tree with c levels though this is not needed. Instead we can split the root node sending $c/2$ categories to the left, and $c/2$ categories to the right. Likewise, each of these can send $c/4$ to the left and $c/4$ to the right. Thus a *tight* upper bound is $\lceil \log c \rceil$.

(c) The lower bound is 3 and the upper bound is $2B - 1$.

3. PROBLEM NOT YET SOLVED

4. PROBLEM NOT YET SOLVED

5. We use the entropy impurity given in Eq. 1,

$$i(N) = -\sum_i P(\omega_i) \log_2 (P(\omega_i)) = H(\omega_i).$$

(a) After splitting on a binary feature $F \in \{R, L\}$, the weighted impurity at the two child nodes is

$$\begin{aligned}
P(L)i(L) + P(R)i(R) &= -P(L)\sum_i P(\omega_i|L)\log_2(P(\omega_i|L)) \\
&\quad -P(R)\sum_i P(\omega_i|R)\log_2(P(\omega_i|R)) \\
&= -\sum_i P(\omega_i, L)\log_2\left(\frac{P(\omega_i, L)}{P(L)}\right) \\
&\quad -\sum_i P(\omega_i, R)\log_2\left(\frac{P(\omega_i, R)}{P(R)}\right) \\
&= -\sum_{i,F} P(\omega_i, F)\log_2(P(\omega_i, F)) \\
&\quad +P(L)\log_2(P(L)) + P(R)\log_2(P(R)) \\
&= H(\omega, F) - H(F).
\end{aligned}$$

Therefore, the drop in impurity is

$$\Delta i(N) = H(\omega) - H(\omega, F) + H(F).$$

But $H(\omega) \leq H(\omega, F) \leq H(\omega) + H(F)$, and therefore we have

$$0 \leq \Delta i(N) \leq H(F) \leq 1 \text{ bit}.$$

(b) At each node, the weighted impurity at the child nodes will be less than that at the parent, even though individual descendandant may have a greater impurity than their parent. For example at the $(x_2 < 0.61)$ node in the upper tree of example 1, the impurity is 0.65, while that at the left child is 0.0, and that at the right is 1.0. The left branch is taken $\frac{2}{3}$ of the time, however, so the weighted impurity at the children is $\frac{2}{3} \times 0 + \frac{1}{3} \times 1 = 0.33$. Similarly, at the $(x_1 < 0.6)$ node of the lower tree, the right child has higher impurity (0.92) than the parent (0.76), but the weighted average at the children is $\frac{2}{3} \times 0 + \frac{1}{3} \times 0.92 = 0.304$. In each case, the reduction in impurity is between 0 and 1 bit, as required.

(c) For B-way branches, we have $0 \leq \Delta i(N) \leq \log_2(B)$ bits.

6. PROBLEM NOT YET SOLVED

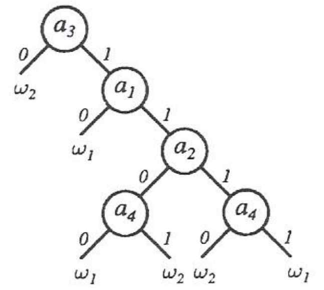
7. PROBLEM NOT YET SOLVED

8. There are four attributes, $\{a_1, a_2, a_3, a_4\}$ to be used in our decision tree.

(a) To select the query at the root node, we investigate queries on each of the four attributes. The following shows the number of patterns sent to the "left" and "right" for each value, and the entropy at the resulting children nodes:

query	sent left	left entropy	sent right	right entropy
a_1	$2\omega_1, 2\omega_2$	1	$2\omega_1, 2\omega_2$	1
a_2	$2\omega_1, 2\omega_2$	1	$2\omega_1, 2\omega_2$	1
a_3	$0\omega_1, 2\omega_2$	0	$4\omega_1, 2\omega_2$	0.9183
a_4	$2\omega_1, 3\omega_2$	0.9710	$2\omega_1, 1\omega_2$	0.9183

Because query a_3 leads to the greatest weighted reduction in impurity, a_3 should be the query at the root node. We continue and grow the tree shown in the figure.



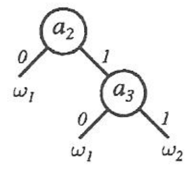
(b) We can expand the tree into rules as

$$\begin{aligned}
\omega_1 &= (a_3\ AND\ NOT a_1)\ OR\ (a_3\ AND\ a_1\ AND\ NOT\ a_2\ AND\ NOT\ a_4) \\
&\quad OR\ (a_3\ AND\ a_1\ AND\ a_2\ AND\ a_4) \\
&= a_3\ AND\ (NOT\ a_1\ OR\ a_1\ AND\ (NOT\ a_2\ AND\ NOT\ a_4)\ OR\ (a_2\ AND\ a_4)). \\
\omega_2 &= NOT\ a_3\ OR\ (a_3 AND\ a_1\ AND\ NOT\ a_2\ AND\ a_4) \\
&\quad OR\ (a_3\ AND\ a_1\ AND\ a_2\ AND\ NOT\ a_4) \\
&= NOT\ a_3 OR (a_3\ AND\ a_1)\ AND\ ((NOT\ a_2\ AND\ a_4)\ OR\ (a_2\ AND\ NOT\ a_4)).
\end{aligned}$$

9. PROBLEM NOT YET SOLVED

10. PROBLEM NOT YET SOLVED

11. PROBLEM NOT YET SOLVED



18. Problem not yet solved

19. Here our strings are composed of letters in the alphabet $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.

(a) Consider the following string (and shift positions)

"a	c	a	c	c	a	c	b	a	c"
1	2	3	4	5	6	7	8	9	10

The last-occurence function gives $\mathcal{F}(\mathtt{a}) = 9$, $\mathcal{F}(\mathtt{b}) = 8$, $\mathcal{F}(\mathtt{c}) = 10$, and 0 otherwise. Likewise, the good-suffix function gives $\mathcal{G}(\mathtt{c}) = 7$, $\mathcal{G}(\mathtt{ac}) = 6$, $\mathcal{G}(\mathtt{bac}) = 0$, and 0 otherwise.

(b) Consider the following string (and shift positions)

"a	b	a	b	a	b	c	b	c	b	a	a	a	b	c	b	a	a"
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

The last-occurence function gives $\mathcal{F}(\mathtt{a}) = 18$, $\mathcal{F}(\mathtt{b}) = 16$, $\mathcal{F}(\mathtt{c}) = 15$ and 0 otherwise. Likewise, the good-suffix function gives $\mathcal{G}(\mathtt{a}) = 17$, $\mathcal{G}(\mathtt{aa}) = 12$, $\mathcal{G}(\mathtt{baa}) = 10$, $\mathcal{G}(\mathtt{cbaa}) = 9$, $\mathcal{G}(\mathtt{bcbaa}) = 8$, $\mathcal{G}(\mathtt{abcbaa}) = 0$, and 0 otherwise.

(c) Consider the following string (and shift positions)

"c	c	c	a	a	a	b	a	b	a	c	c	c"
1	2	3	4	5	6	7	8	9	10	11	12	13

The last-occurence function gives $\mathcal{F}(\mathtt{a}) = 10$, $\mathcal{F}(\mathtt{b}) = 9$, $\mathcal{F}(\mathtt{c}) = 13$, and 0 otherwise. Likewise, $\mathcal{G}(\mathtt{c}) = 12$, $\mathcal{G}(\mathtt{cc}) = 11$, $\mathcal{G}(\mathtt{ccc}) = 1$, $\mathcal{G}(\mathtt{accc}) = 0$, and 0 otherwise.

(d) Consider the following string (and shift positions)

"a	b	b	a	b	b	a	b	b	c	b	b	a	b	b	c	b	b	a"
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

The last-occurence function gives $\mathcal{F}(\mathtt{a}) = 19$, $\mathcal{F}(\mathtt{b}) = 18$, $\mathcal{F}(\mathtt{c}) = 16$, and 0 otherwise. Likewise, $\mathcal{G}(\mathtt{a}) = 13$, $\mathcal{G}(\mathtt{ba}) = 12$, $\mathcal{G}(\mathtt{bba}) = 11$, $\mathcal{G}(\mathtt{cbba}) = 10$, $\mathcal{G}(\mathtt{bcbba}) = 9$, $\mathcal{G}(\mathtt{bbcbba}) = 8$, $\mathcal{G}(\mathtt{abbcbba}) = 7$, $\mathcal{G}(\mathtt{babbcbba}) = 6$, $\mathcal{G}(\mathtt{bbabbcbba}) = 5$, and 0 otherwise.

20. We use the information from Fig. 8.8 in the text.

(a) The string and the number of comparisons at each shift are:

"p	r	o	b	a	b	i	l	i	t	i	e	s	_	f	o	r	_	e	s	t	i	m	a	t	e	s"
1	1	1	1	1	1	1	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	9

The sum is the total number of character comparisons: 28.