Q1 DS & A

(a)

```
boolean findNumber(Treenode* node, int x){
    while (node != nullptr){
        if (x == node->val){
            return true;
        } else if (x < node->val){
            node = node->left
        } eles {
            node = node->right
        }
    }
    return false
}
```

$O(\log n)$

(b)

```
boolean findSumtoANumber(Treenode* node, int z){
    Static Treenode* root = node
    int temp=0

    if (node == nullptr) return false;
    temp = z - node->val
    if (findNumber(root, temp)) return true
    return (findSumtoANumber(node->left, z) ||
            findSumtoANumber(node->right, z));
```

$O(n\log n)$

更快的方法是遍历，哈希表存储，查询 $O(n)$

Q 2

Q3 OS

(a)

Stream	1	2	3	1	4	4	5	1	3	
FIFO	1	1	1	1	4	4	4	4	3	
		2	2	2	2	2	5	5	5	
			3	3	3	3	3	1	1	
	F	F	F		F		F	F	F	7
LRU	1	1	1	1	1	1	1	1	1	
		2	2	2	4	4	4	4	3	
			3	3	3	3	5	5	5	
	F	F	F		F		F		F	6
OPT	1	1	1	1	1	1	1	1	1	
		2	2	2	4	4	5	5	5	
			3	3	3	3	3	3	3	
	F	F	F		F		F			5

(b) $0.9 \times (10+200) + 0.1 \times (10+200+200)$

(c) OS, 9-th edition English, 8.1, P387

Both paging and segmentation have their strengths. Paging, which is transparent to the programmer, eliminates external fragmentation and thus provides efficient use of main memory. In addition, because the pieces that are moved in and out of main memory are of fixed, equal size, it is possible to develop sophisticated memory management algorithms that exploit the behavior of programs, as we shall see. Segmentation, which is visible to the programmer, has the strengths listed earlier, including the ability to handle growing data structures, modularity, and support for sharing and protection. To combine the advantages of both, some systems are equipped with processor hardware and OS software to provide both.

Q4 Database

(a) (1)
```
select category
from Book
group by category
having count(bid) ≥ 2000
```

(2)
```
select title
from Book natural join Borrow
group by bid
having count(distinct sid) ≥ all (select count(distinct sid), from brrow
                                              groupby bid )
```

第二问，Hongru 版本

```
select title
from (select bid, count(distinct sid) as cnt
      from Borrow
      groupby bid) as T join book using bid
where T.cnt = (select max(cnt) from T)
```

(b)

公式：$\left\lceil \frac{br}{M-2} \right\rceil \cdot bs + br$ block transfers $2 \cdot \left\lceil \frac{br}{M-2} \right\rceil$ seeks

$b_{student} = \frac{6000}{10} = 600$ $\quad b_{borrow} = \frac{30000}{30} = 1000$

(1) if no memory buffer size restriction $M \to +\infty$

need $br + bs = 1600$ block transfers, and 2 seeks

(2) if $M = 3$

need $br \times bs + br = 600600$ block transfers, $2br = 1200$ seeks

(c) (Student ⋈ Book) ⋈ Borrow

6000 × 20000

(Student ⋈ Borrow) ⋈ Book

30000

Q5 **Answer:**

The error rate for the data without partitioning on any attribute is

$$E_{orig} = 1 - \max(\frac{50}{100}, \frac{50}{100}) = \frac{50}{100}.$$

After splitting on attribute A, the gain in error rate is:

	$A = T$	$A = F$
$+$	25	25
$-$	0	50

$$E_{A=T} = 1 - \max(\frac{25}{25}, \frac{0}{25}) = \frac{0}{25} = 0$$

$$E_{A=F} = 1 - \max(\frac{25}{75}, \frac{50}{75}) = \frac{25}{75}$$

$$\Delta_A = E_{orig} - \frac{25}{100}E_{A=T} - \frac{75}{100}E_{A=F} = \frac{25}{100}$$

After splitting on attribute B, the gain in error rate is:

	$B = T$	$B = F$
$+$	30	20
$-$	20	30

$$E_{B=T} = \frac{20}{50}$$

$$E_{B=F} = \frac{20}{50}$$

$$\Delta_B = E_{orig} - \frac{50}{100}E_{B=T} - \frac{50}{100}E_{B=F} = \frac{10}{100}$$

After splitting on attribute C, the gain in error rate is:

	$C = T$	$C = F$
$+$	25	25
$-$	25	25

$$E_{C=T} = \frac{25}{50}$$

$$E_{C=F} = \frac{25}{50}$$

$$\Delta_C = E_{orig} - \frac{50}{100}E_{C=T} - \frac{50}{100}E_{C=F} = \frac{0}{100} = 0$$

The algorithm chooses attribute A because it has the highest gain.

(b) Repeat for the two children of the root node.

Answer:

Because the $A = T$ child node is pure, no further splitting is needed. For the $A = F$ child node, the distribution of training instances is:

B	C	Class label	
		$+$	$-$
T	T	0	20
F	T	0	5
T	F	25	0
F	F	0	25

The classification error of the $A = F$ child node is:

$$E_{orig} = \frac{25}{75}$$

After splitting on attribute B, the gain in error rate is:

	$B = T$	$B = F$
$+$	25	0
$-$	20	30

$$E_{B=T} = \frac{20}{45}$$
$$E_{B=F} = 0$$
$$\Delta_B = E_{orig} - \frac{45}{75}E_{B=T} - \frac{20}{75}E_{B=F} = \frac{5}{75}$$

After splitting on attribute C, the gain in error rate is:

	$C = T$	$C = F$
$+$	0	25
$-$	25	25

$$E_{C=T} = \frac{0}{25}$$
$$E_{C=F} = \frac{25}{50}$$
$$\Delta_C = E_{orig} - \frac{25}{75}E_{C=T} - \frac{50}{75}E_{C=F} = 0$$

The split will be made on attribute B.

(c) How many instances are misclassified by the resulting decision tree?

Answer:

20 instances are misclassified. (The error rate is $\frac{20}{100}$.)

(d) Repeat parts (a), (b), and (c) using C as the splitting attribute.

Answer:

For the $C = T$ child node, the error rate before splitting is:
$E_{orig} = \frac{25}{50}$.
After splitting on attribute A, the gain in error rate is:

	$A = T$	$A = F$
$+$	25	0
$-$	0	25

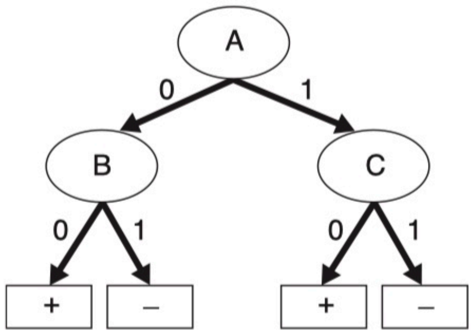
$$E_{A=T} = 0$$
$$E_{A=F} = 0$$
$$\Delta_A = \frac{25}{50}$$

After splitting on attribute B, the gain in error rate is:

	$B = T$	$B = F$
$+$	5	20
$-$	20	5

$$E_{B=T} = \frac{5}{25}$$
$$E_{B=F} = \frac{5}{25}$$
$$\Delta_B = \frac{15}{50}$$

Therefore, A is chosen as the splitting attribute.



Training:

Instance	A	B	C	Class
1	0	0	0	+
2	0	0	1	+
3	0	1	0	+
4	0	1	1	−
5	1	0	0	+
6	1	0	0	+
7	1	1	0	−
8	1	0	1	+
9	1	1	0	−
10	1	1	0	−

Validation:

Instance	A	B	C	Class
11	0	0	0	+
12	0	1	1	+
13	1	1	0	+
14	1	0	1	−
15	1	0	0	+

Figure 4.2. Decision tree and data sets for Exercise 8.

For the $C = F$ child, the error rate before splitting is: $E_{orig} = \frac{25}{50}$.
After splitting on attribute A, the error rate is:

	$A = T$	$A = F$
$+$	0	25
$-$	0	25

$$E_{A=T} = 0$$
$$E_{A=F} = \frac{25}{50}$$
$$\Delta_A = 0$$

After splitting on attribute B, the error rate is:

	$B = T$	$B = F$
$+$	25	0
$-$	0	25

$$E_{B=T} = 0$$
$$E_{B=F} = 0$$
$$\Delta_B = \frac{25}{50}$$

Therefore, B is used as the splitting attribute.
The overall error rate of the induced tree is 0.

(e) Use the results in parts (c) and (d) to conclude about the greedy nature of the decision tree induction algorithm.
The greedy heuristic does not necessarily lead to the best tree.

Q6 Information retrival

(a) (1) Why do we use a harmonic mean rather than the simpler average (arithmetic mean)? Recall that we can always get 100% recall by just returning all documents, and therefore we can always get a 50% arithmetic mean by the same process. This strongly suggests that the arithmetic mean is an unsuitable measure to use. In contrast, if we assume that 1 document in 10,000 is relevant to the query, the harmonic mean score of this strategy is 0.02%. The harmonic mean is always less than or equal to the arithmetic mean and the geometric mean. When the values of two numbers differ greatly, the harmonic mean is closer to their minimum than to their arithmetic mean; see Figure 8.1.

English book P157

(2) one is more important than the other in many circumstances. Typical web surfers would like every result on the first page to be relevant (high precision) but have not the slightest interest in knowing let alone looking at every document that is relevant. In contrast, various professional searchers such as

English book P156

(b) (1)

TrainRocchio($\mathbb{C}$, $\mathbb{D}$)

for each $c_j \in \mathbb{C}$
do $D_j \leftarrow \{d : \langle d, c_j \rangle \in \mathbb{D}\}$
$\quad \vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$
return $\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}$

ApplyRocchio($\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}$, d)

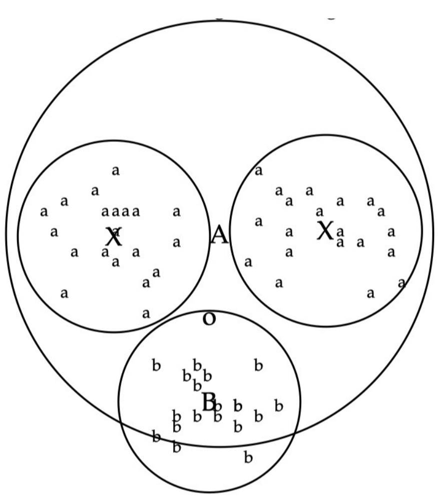
return $\arg\min_j |\vec{\mu}_j - \vec{v}(d)|$ ← 替换为cosine

(2) $\vec{w}^T \vec{x} = b$

$\vec{w} = \vec{\mu}_{c_1} - \vec{\mu}_{c_2}$

$b = \frac{1}{2}(|\vec{\mu}_{c_1}|^2 - |\vec{\mu}_{c_2}|^2)$

(c)



English Book P295