Special Functions

Characteristic Function :

SOME SPECIAL FUNCTIONS

CHARACTERISTIC FUNCTION OF A SET

Introduction

In this section, we shall deal with functions from the universal set U to the set $\{0, 1\}$, using which statements about sets and their operations can be represented on a computer in terms of binary numbers and hence can be dealt with easily.

Definition

If A is a subset of a universal set U, the *characteristic function* f_A of A is defined as the function from U to the set $\{0, 1\}$ such that

$$F_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}$$

For example, if $U = \{1, 2, 3, 4, 5\}$ and $A = \{2, 4\}$ then $f_A(1) = 0 = f_A(3) = f_A(5)$ and $f_A(2) = f_A(4) = 1$, since $2, 4 \in A$ and $1, 3, 5 \notin A$.

Note The values of characteristic functions are always either 1 or 0.

Properties of Characteristic Functions

1. If A is a subset of U then $f_{\bar{A}}(x) = 1 - f_A(x)$, for all $x \in U$.

Proof

$$\begin{aligned} f_{\bar{A}}(x) = 1 &\Leftrightarrow x \in \bar{A} \\ &\Leftrightarrow x \notin A \\ &\Leftrightarrow f_A(x) = 0 \end{aligned}$$

Now
$$\begin{aligned} f_{\bar{A}}(x) = 0 &\Leftrightarrow x \notin \bar{A} \\ &\Leftrightarrow x \in A \\ &\Leftrightarrow f_A(x) = 1 \\ &\Leftrightarrow 1 - f_A(x) = 0 \end{aligned}$$

$\therefore$ $f_{\bar{A}}(x) = 1 - f_A(x)$, when $x \in A$ (2)

From (1) and (2), it follows that

$$f_{\bar{A}}(x) = 1 - f_A(x), \text{ for all } x \in U.$$

2. If A and B are any two subsets of U, then

$$f_{A \cap B}(x) = f_A(x) \cdot f_B(x), \text{ for all } x \in U.$$

Proof

$$\begin{aligned} f_{A \cap B}(x) = 1 &\Leftrightarrow x \in A \cap B \\ &\Leftrightarrow x \in A \text{ and } x \in B \\ &\Leftrightarrow f_A(x) = 1 \text{ and } f_B(x) = 1 \\ &\Leftrightarrow f_A(x) \cdot f_B(x) = 1 \end{aligned}$$

$\therefore$ $f_{A \cap B}(x) = f_A(x) \cdot f_B(x)$, when $x \in A \cap B$ (1)

Now
$$\begin{aligned} f_{A \cap B}(x) = 0 &\Leftrightarrow x \notin A \cap B \\ &\Leftrightarrow x \notin A \text{ or } x \notin B \\ &\Leftrightarrow f_A(x) = 0 \text{ or } f_B(x) = 0 \\ &\Leftrightarrow f_A(x) \cdot f_B(x) = 0 \end{aligned}$$

HASHING Function

$= f_A(x)\{1 - f_B(x)\}$, by property (1)

HASHING FUNCTIONS

Introduction

When records (data) are stored in a direct access file in a computer, the computer can retrieve a specific record without reading other records first. This is possible only if the computer can identify the memory locations in which records in the form of non-negative integers, called *keys* are stored. A transformation that maps the set of keys to a set of addresses (of memory cells) is called a *hashing function*. Even though various hashing functions are used, we will discuss one of the most commonly used hashing function obtained by division method or congruence method.

Definition

If n is the number of available memory locations and k is the non-negative integer representing the key, the hashing function $h(k)$ representing the address of the memory cell in which k is stored is defined as

$$h(k) = k(\text{mod } n)$$

i.e., $h(k)$ is simply the remainder when k is divided by n and it takes values from the set $\{0, 1, 2, \ldots, n - 1\}$, known as the address set.

[redacted] As a good hashing functions should uniformly distribute the records (keys) over the elements of the address set, n is chosen suitably. Usually n is chosen as a prime number, greater than the maximum number of records in the file.

Example

Example

Let us try to get the addresses of 6 memory cells in which the integers 23, 38, 46, 55, 67 and 71 are to be stored, assuming that there are 6 records in the file.

Since the smallest prime number greater than 6 is 7, we shall choose $n = 7$. Then the address of the memory cells are given by the hashing function $h(k) = k(\text{mod } 7)$. Obviously the address set is $\{0, 1, 2, 3, 4, 5, 6\}$.

When $k = 23, 38, 46$ and 55, the values of $h(k)$ are 2, 3, 4 and 6 respectively. viz., the integers 23, 38, 46 and 55 are stored in the memory cells with addresses 2, 3, 4 and 6 as shown in Table. 4.1.

Table 4.1

$h(k)$:	0	1	2	3	4	5	6
k:	71	—	23	38	46	67	55

The next integer to be stored is 67.

When $k = 67$, $h(k) = 4$. viz., 67 must be stored in the cell with address 4. But this cell with address 4 has already been occupied by 46.

When the memory cell with address $h(k)$ is already occupied at the time we try to store k in it, a *collision* is said to occur. Thus when we try to store 67 in the memory cell with address 4, collision occurs. In general, a collision for a hash function occurs if $h(k_1) = h(k_2)$, but $k_1 \neq k_2$.

To resolve collision the following simple method called *collision resolution policy* is used. The first empty cell that follows the already occupied cell is used to store the current value of k.

In our example, the first unoccupied cell that follows the memory cell numbered 4 is that with addrss 5. The integer 67 is thus stored in this cell. The last integer 71 is then stored in the cell with address 0. The cell with address 1 will remain as an unoccupied cell.

If we wist to retrieve a stored value k, we compute $h(k)$ and start reading the value stored at the cell $h(k)$. If k is not in this cell, we scan the values stored in the succeeding cells one after the other. In this process, if we reach an empty cell, we conclude that k is not available in the file.