UE19CS332

Algorithms For Web And Information Retrieval

Assignment – 1

NAME : PRIYA MOHATA
SRN : PES2UG19CS301
SECTION: E

Program 1:

Problem Statement :
Write a program to count word frequency in a text file

Week-0 > program1.py > ...

```python
# WAP to count word frequency in a text file

# PRIYA MOHATA
# PES2UG19CS301
# SECTION E – WEEK 1 ASSIGNMENT

def main():

    f=open('sample.txt','r')
    print('Enter your word')
    word=input()

    freq_word=0
    for line in f.readlines():
        line=line.split('\n')[0].split()
        freq_word=freq_word+line.count(word)

    print('The frequency of',word,'is:',freq_word)
    f.close()

if __name__=="__main__":
    main()
```

Sample Text File :

Week-0 > sample.txt

```
Even the saints say, “God! Do not put us to test.”
A test saps the energies of a person, generates mental agony and makes him incapable of thinking in a proper and logical
Examinees shudder at the thought of the examination and the fear of failure makes them depressed
Evil dreams haunt the students and the dose of admonition given to them regularly disturbs them.
Before the examination, the students suspend pleasure-oriented activities.
They do not go to the playground.
They also cancel their picnic schedules and forget about going to the latest movies.
They are busy with their books.
They go on reading while sipping a cup of tea.
They go on revising while lying in their beds.
They discuss questions, talk about books and dream about answers.
Some confine themselves to their rooms and stick to chairs while some others sit in remote corners.

```

O/P :

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
apple@Apples-MacBook-Air PES2UG19CS301-PRIYA MOHATA-AIWIR % cd Week-0
apple@Apples-MacBook-Air Week-0 % python3 program1.py
Enter your word
go
The frequency of go is: 3
apple@Apples-MacBook-Air Week-0 % python3 program1.py
Enter your word
they
The frequency of they is: 0
apple@Apples-MacBook-Air Week-0 % python3 program1.py
Enter your word
They
The frequency of They is: 6
apple@Apples-MacBook-Air Week-0 % python3 program1.py
Enter your word
confine
The frequency of confine is: 1
apple@Apples-MacBook-Air Week-0 %
```

Program 2a:

Problem Statement:

Implement searching [with respect to a particular word , check if it exists or not , return its index]

Week-0 > program2a.py > ...

```
# WAP - take an array and store the words in the array
# Implement searching [ wrt a particular word , check if it exists or not , return its index ]

# PRIYA MOHATA
# PES2UG19CS301
# SECTION E - WEEK 1 ASSIGNMENT


def main():
    arr=['pes university','compiler design','cloud computing','ooad','aiwir','nam']
    print("Enter the word to be searched")
    word=input()
    pos=-1
    for i in range(0,len(arr)):
        if word==arr[i]:
            pos=i;
    if(pos==-1):
        print("Element not found!")
    else:
        print("Element found at position:",pos)

if __name__=="__main__":
    main()

```

O/P :

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
apple@Apples-MacBook-Air PES2UG19CS301-PRIYA MOHATA-AIWIR % cd Week-0
apple@Apples-MacBook-Air Week-0 % python3 program2a.py
Enter the word to be searched
ooad
Element found at position: 3
apple@Apples-MacBook-Air Week-0 % python3 program2a.py
Enter the word to be searched
priya
Element not found!
apple@Apples-MacBook-Air Week-0 %
```

Program 2b:

Problem Statement : Implement a function to access the first and last element print the words as well

Week-0 > program2b.py > ...

```python
# Implement a function to access the first and last element print the words as well

# PRIYA MOHATA
# PES2UG19CS301
# SECTION E - WEEK 1 ASSIGNMENT

def access_first(arr):
    print(arr[0])

def access_last(arr):
    print(arr[len(arr)-1])

def main():
    arr=['pes university','compiler design','cloud computing','ooad','aiwir','nam']
    access_first(arr)
    access_last(arr)

if __name__=="__main__":
    main()

```

O/P:

```
apple@Apples-MacBook-Air Week-0 % python3 program2b.py
pes university
nam
apple@Apples-MacBook-Air Week-0 %
```

Program 2c:

Problem Statement :

Insert an element at the end of the array and traverse the array

```
Week-0 > program2c.py > ...
1   # Insert an element at the end of the array and traverse the array
2
3   # PRIYA MOHATA
4   # PES2UG19CS301
5   # SECTION E - WEEK 1 ASSIGNMENT
6
7   def main():
8       arr=['pes university','compiler design','cloud computing','ooad','aiwir','nam']
9       print("Enter the word to be inserted")
10      word=input()
11
12      arr.append(word)
13
14      for i in arr:
15          print(i,end=' ')
16
17  if __name__=="__main__":
18      main()
19
```

O/P:

```
apple@Apples-MacBook-Air Week-0 % python3 program2c.py
Enter the word to be inserted
semester6
pes university compiler design cloud computing ooad aiwir nam semester6
apple@Apples-MacBook-Air Week-0 %
```

Program 3 :

Problem Statement :

Write a program using linked list to

- Store words
- Insert new word
- Search for a given word and return its index position
- Access element in the first, last and a particular position
- Compare both the data structures and analyze which approach is efficient.

CODE :

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include <string> // for string class
using namespace std;

// PRIYA MOHATA
// PES2UG19CS301
// SECTION E - WEEK 1 ASSIGNMENT

struct Node
{
    string data;
    struct Node * link;
};

struct Node* getnode()
{
    struct Node  *x;
    x=(struct Node *)malloc(sizeof(struct Node));
    if(x==NULL)
    {
        printf("insufficient memory");
        return(0);
    }
    return(x);
}
struct Node* insert_at_front(struct Node* head,string ele)
{
    struct Node *temp;
    temp=getnode();
    temp->data=ele;
    temp->link=head;
    return(temp);
}
```

```
struct Node* insert_at_rear(struct Node* head,string ele)
{
    struct Node *temp=NULL;
    temp=getnode();
    temp->data=ele;
    temp->link=NULL;
    struct Node *cur=head;
    if(head==NULL)
    {
        return(temp);
    }
    while(cur->link!=NULL)
    {
        cur=cur->link;
    }
    cur->link=temp;
    return(head);
}


void display(struct Node *head)
{
    struct Node *cur=head;
    while(cur!=NULL)
    {
        cout<<cur->data<<" ";
        cur=cur->link;
    }
}
```

```
struct Node* insert_at_position(struct Node *head,string ele,int pos)
{
    struct Node *temp=getnode();
    temp->data=ele;
    temp->link=NULL;
    struct Node* first=head;
    if(pos==1)
    {
        temp->link=head;
        return(temp);
    }
    struct Node *cur=first;
    struct Node *prev=NULL;
    for(int i=1;i<=pos-1;i++)
    {
        prev=cur;
        cur=cur->link;
    }
    prev->link=temp;
    temp->link=cur;
    return(head);
}
```

```
void search(struct Node* head,string ele){
    struct Node *cur=head;
    int count=0;
    int pos=-1;
    while(cur!=NULL)
    {
        if(cur->data==ele)
        pos=count;
        count++;
        cur=cur->link;
    }
    if(pos!=-1){
        cout<<"Element found at position : "<<pos<<"\n";
    }
    else{
        cout<<"Element not found\n";
    }
}

int main(void)
```

```
int main(void)
{
  struct Node *first=NULL;
string ele;
int position;
for(;;)
{
printf("\n 1.Insert at front\n 2.Insert at rear\n 3.Display\n 4.Insert at position\n 5.Search for a word\n");
printf("\n enter your choice\n");
int ch;
scanf("%d",&ch);

switch(ch)
{
    case 1: printf("\n enter the element\n");
            cin>>ele;
            first=insert_at_front(first,ele);
            break;
   case 2: printf("\n enter the element\n");
           cin>>ele;
           first=insert_at_rear(first,ele);
           break;
    case 3: display(first);
           break;
    case 4:printf("\n enter the element\n");
           cin>>ele;
           printf("enter the position");
           scanf("%d",&position);
           first=insert_at_position(first,ele,position);
           break;
    case 5:printf("\n enter the element\n");
           cin>>ele;
           search(first,ele);
           break;
   default:exit(0);
}
}
return(0);
}
```

O/P:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

apple@Apples-MacBook-Air PES2UG19CS301-PRIYA MOHATA-AIWIR % cd Week-0
apple@Apples-MacBook-Air Week-0 % g++ program3.cpp
apple@Apples-MacBook-Air Week-0 % ./a.out

 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
1

 enter the element
priya

 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
1

 enter the element
pesu

 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
2

 enter the element
srn

 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
2
```

```
 enter the element
pes2ug19cs301

 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
3
pesu priya srn pes2ug19cs301
 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
5

 enter the element
srn
Element found at position : 2

 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
4

 enter the element
sem5
enter the position
3

 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word
```

```
 enter the element
sem5
enter the position
3

 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
3
pesu priya sem5 srn pes2ug19cs301
 1.Insert at front
 2.Insert at rear
 3.Display
 4.Insert at position
 5.Search for a word

 enter your choice
90
apple@Apples-MacBook-Air Week-0 %
```

Compare both the data structures and analyze which approach is efficient.

Operation	Array	Linked List
Accessing the element	Direct access because of index	Sequential access therefore its slow.
Insertion and Deletion	Slow relatively as shifting is required.	Easier, fast and efficient.
Searching	Supports linear and binary search	Supports linear search
Memory Utilization	Ineffective	Efficient