Assignment No. 2

AIM: Write a program to solve optimal storage on tapes problem using Greedy approach.

OBJECTIVE:

1. To understand the concept of Greedy Approach.
2. To obtain the minimum retrieval time of processes on tapes.

THEORY:

Greedy Method:

A **greedy algorithm** is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

In general, greedy algorithms have five components:

1. A candidate set, from which a solution is created
2. A selection function, which chooses the best candidate to be added to the solution
3. A feasibility function, that is used to determine if a candidate can be used to contribute to a solution
4. An objective function, which assigns a value to a solution, or a partial solution, and
5. A solution function, which will indicate when we have discovered a complete solution

Examples:

- The activity selection problem is characteristic to this class of problems, where the goal is to pick the maximum number of activities that do not clash with each other.
- In the Macintosh computer game Crystal Quest the objective is to collect crystals, in a fashion similar to the travelling salesman problem. The game has a demo mode, where the game uses a greedy algorithm to go to every crystal. The artificial intelligence does not account for obstacles, so the demo mode often ends quickly.
- The matching pursuit is an example of greedy algorithm applied on signal approximation.
- A greedy algorithm finds the optimal solution to Malfatti's problem of finding three disjoint circles within a given triangle that maximize the total area of the circles; it is conjectured that the same greedy algorithm is optimal for any number of circles.
- A greedy algorithm is used to construct a Huffman tree during Huffman coding where it finds an optimal solution.

- In decision tree learning greedy algorithms are commonly used however they are not guaranteed to find the optimal solution.

Optimal storage problem:

Input: We are given 'n' problem that are to be stored on computer tape of length L and the length of program i is Li

Such that 1 ≤i≤ n and Σ 1≤k≤j Li≤ 1

Output: A permutation from all n! For the n programs so that when they are stored on tape in the order the MRT is minimized.

Example:

Let n = 3, (l1, l2, l3) = (8, 12, 2). As n = 3, there are 3! =6 possible ordering.
All these orderings and their respective d value are given below:

Ordering	d (i)	Value
1, 2, 3	8 + (8+12) + (8+12+2)	50
1, 3, 2	8 + 8 + 2 + 8 + 2 + 12	40
2, 1, 3	12 + 12 + 8 +12 + 8 + 2	54
2, 3, 1	12 + 12 +2 +12 + 2 + 8	48
3, 1, 2	2 + 2 + 8 + 2 + 8+ 12	34
3, 2, 1	2 + 2 +12 + 2 + 12 + 8	38

The optimal ordering is 3, 1, 2.

The greedy method is now applied to solve this problem. It requires that the programs are stored in non-decreasing order which can be done in O (nlogn) time.

Greedy solution:

i. Make tape empty

ii. Fori = 1 to n do;

iii. Grab the next shortest path

iv. Put it on next tape.

The algorithm takes the best shortest term choice without checking to see whether it is a big long term decision.

Algorithm:

```
Algorithm Optimal_Storage (n, m)
{
K = 0; // Next tape to be stored.
For i = 1 to n do
        {
        Write (i, k); // "Assign program", j, "to tape", k;
        k = (k +1) mod m;
        }
}
```

Example:

For e.g. Suppose there are 3 programs of lengths 2, 5 and 4 respectively. So there are total 3! = 6 possible orders of storage.

	Order	Total Retrieval Time	Mean Retrieval Time
1	1 2 3	2 + (2 + 5) + (2 + 5 + 4) = 20	20/3
2	1 3 2	2 + (2 + 4) + (2 + 4 + 5) = 19	19/3
3	2 1 3	5 + (5 + 2) + (5 + 2 + 4) = 23	23/3
4	2 3 1	5 + (5 + 4) + (5 + 4 + 2) = 25	25/3
5	3 1 2	4 + (4 + 2) + (4 + 2 + 5) = 21	21/3
6	3 2 1	4 + (4 + 5) + (4 + 5 + 2) = 24	24/3

It's clear that by following the second order in storing the programs, the mean retrieval time is least.

In above example, the first program's length is added 'n' times, the second 'n-1' times…and so on till the last program is added only once. So, careful analysis suggests that in order to minimize the MRT, programs having greater lengths should be put towards the end so that the summation is reduced. Or, the lengths of the programs should be sorted in increasing order.

That's the **Greedy Algorithm** in use – at each step we make the immediate choice of putting the program having the least time first, in order to build up the ultimate optimized solution to the problem piece by piece.

Conclusion:

Thus we have implemented optimal storage on tapes using Greedy Approach.

PROGRAM

```
#include<stdio.h>

int i,j,n,t=3; //i j for counters, n for number of programs, t for no of tapes

int p[30],l[30],temp,m,tape[10][10],tape1[30][30]; //p[] stores program no,and l[] stores program length

int itemcount[10]; //to store no of items in each tape

int c1=0,c2=0,c3=0;

float mrt[10];

void getval()

{

printf("Enter no of programs\n");

scanf("%d",&n);

printf("Enter number of tapes");

scanf("%d",&t);

for(i=0;i<n;i++)

{

printf("Enter length of program %d",i+1);

scanf("%d",&l[i]);

p[i]=i;

}
```

```
for(i=1;i<=t;i++) // to initialise tape matrix
{
for(j=0;j<30;j++)
tape[i][j]=0;
}
}
void sort()
{
for(i=0;i<n;i++)
{
for(j=0;j<n-1;j++)
{
if(l[j]>l[j+1])
{
temp=l[j];
l[j]=l[j+1];
l[j+1]=temp;
m=p[j];
p[j]=p[j+1];
p[j+1]=m;
}
}
}
for(i=0;i<n;i++)
```

```
{
printf("program %d\t",p[i]);
printf("length %d\n",l[i]);
}
}
void arrange()
{ int count=0;
int r=0;
for(i=0;i<10;i++)
itemcount[i]=0;
for(i=0;i<n;i++)
{
count++;
tape[count][r]=l[i];
itemcount[count]++;

if(count==t)
{
r++;
count=0;
}
}
}
void printtape()
```

```
{
int r=0;
for(i=1;i<=t;i++)
{ printf("\ntape %d",i);
while(tape[i][r]!=0)
{
printf("%d\t",tape[i][r]);
r++;
}
printf("\n");
r=0;
}
}
void calmrt()
{
int r=0,it,k;
float sum[10]; //to save sum of each tape
for(i=0;i<10;i++) //init sum
{
sum[i]=0;
}
j=0;
for(i=1;i<=t;i++)
{
```

```
for(j=0;j<itemcount[i];j++)
{
for(k=0;k<=j;k++)
{
sum[i]=sum[i]+tape[i][k];
}
r++;
it--;
}
r=0;
mrt[i]=sum[i]/itemcount[i];
printf("MRT of tape %d is %f\n",i,mrt[i]);
}
}
void finalmrt()
{ float final_mrt=0;
for(i=1;i<=t;i++)
{
final_mrt+=mrt[i];
}
final_mrt=final_mrt/t;
printf("final mrt is %f\n",final_mrt);
}
int main()
```

```
{
getval();
sort();
arrange();
printtape();
calmrt();
finalmrt();
return 0;
}
```

OUTPUT:

```
Enter no of programs
8
Enter number of tapes3
Enter length of program 12
Enter length of program 23
Enter length of program 34
Enter length of program 46
Enter length of program 51
Enter length of program 67
Enter length of program 75
Enter length of program 84
program 4       length 1
program 0       length 2
program 1       length 3
program 2       length 4
program 7       length 4
```

program 6 length 5

program 3 length 6

program 5 length 7

tape 11 4 6

tape 22 4 7

tape 33 5

MRT of tape 1 is 5.666667

MRT of tape 2 is 7.000000

MRT of tape 3 is 5.500000

final mrt is 6.055555

Process exited after 33.65 seconds with return value 0

Press any key to continue . . .