

**Open Ended Problem**

**Subject :**  Analysis & Design Of Algorithms

**Subject code :** 2150703

**Department :** CO (EVE)

**Semester :** 5th

**Prepared By :** Group-04

Pankti Bhalodia 130420107504

Ankita Bhatti 130420107505

Dhruvi Gohil 130420107517

Shailee Naik 130420107532

Shreya Naik 130420107533

**Submit To :** Prof. Bintu Kadhiwala

Prof. Urvashi Mistry

**Aim :** Optimal Storage On Multiple Tapes

**Description :**

**Input**

* There are ‘n’ programs that are to be stored on a computer tape of length ‘l’.
* Associated with each program ‘i’ is a length li 1<i<n. Clearly, all programs can be stored on the tape iff and only iff the sum of the lengths of the programs is at most l. ∑1≤i≤n li ≤l

(l1+l2+……..+ln≤l)

**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.
* All six permutations n=3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |

|  |  |  |
| --- | --- | --- |
| 1 | 3 | 2 |

|  |  |  |
| --- | --- | --- |
| 2 | 3 | 1 |

|  |  |  |
| --- | --- | --- |
| 2 | 1 | 3 |

|  |  |  |
| --- | --- | --- |
| 3 | 1 | 2 |

|  |  |  |
| --- | --- | --- |
| 3 | 2 | 1 |

**MRT [Mean Retrieval Time]**

* If the programs are stored in the order I=i1, i2,………….,in , the time tj needed to retrieve program ij is proportional to ∑1≤ k ≤j lik.
* If all program are retrieved equally and every time head point to the front, then expected MRT is given by :MRT = (1/n) ∑ 1≤ k ≤j lik
* Minimizing the MRT is equivalent to minimizing D(I).
* D(I) = ∑ 1≤ j ≤n ∑ 1≤ k ≤j lik

**Example :**

Let n = 3 and (/ 1 , Ii, 1 3 ) = (5, 10, 3). There are n! possible orderings. These orderings and their respective D(I) values are:

|  |  |  |
| --- | --- | --- |
| no | Ordering I | D(I) |
| 1 | (1,2,3) | 5+(5+10)+(5+10+3) = 38 |
| 2 | (1,3,2) | 5+(5+3)+(5+3+10) = 31 |
| 3 | (2,1,3) | 10+(10+5)+(10+5+3) = 43 |
| 4 | (2,3,1) | 10+(10+3)+(10+3+5) = 41 |
| 5 | (3,1,2) | 3+(3+5)+(3+5+10) = 29 |
| 6 | (3,2,1) | 3+(3+10)+(3+10+5) = 34 |

**Greedy solution**

* + The greedy method simply requires us to store the programs in non decreasing order of their lengths.
  + **Case 1 : single tape**

Algorithm store()

{

                          \\Make tape empty.

                         For i=1 to n do

                                     grab the next shortest file put it next on tape

}

**Example :**

Let n=5 and l1=5 l2=7 l3=10 l4=20 l5=30

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

* + - Elements are stored in sorted order of its length.
    - Insert 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 |  |  |  |  |

Insert 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 7 |  |  |  |

Insert 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 7 | 10 |  |  |

Insert 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 7 | 10 | 20 |  |

Insert 30

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 7 | 10 | 20 | 30 |

**Case 2 : Multiple tapes**

Algorithm store(n,m)

\\ n is the number of programs and m is the number of tapes

{

j=1; // Next tape to store on.

For i=1 to n do

{

                     write(“Append program”, i ,”to permutation for tape”,j);

                     j=(j+1)mod m;

}

}

**Example :**

We want to store files of lengths ( in MB). {12, 34, 56, 73, 24, 11, 34, 56, 78, 91, 34, 45} three tapes. How should we store them on the tapes so that the mean retrieval time is minimized.

**Solution :**

Store files by non decreasing length. First sort the files in increasing order of length. For this we can use heapsort, merge sort or quicksort algorithm.

* + - Sorted elements are:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 12 | 24 | 34 | 34 | 34 | 45 | 56 | 56 | 73 | 78 | 91 |

* + - Insert 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tape 1 | 11 |  |  |  |
| Tape 2 |  |  |  |  |
| Tap2 |  |  |  |  |

* + - Insert 12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tape 1 | 11 |  |  |  |
| Tape 2 | 12 |  |  |  |
| Tap2 | 13 |  |  |  |

* + - Insert 24

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tape 1 | 11 | 14 |  |  |
| Tape 2 | 12 |  |  |  |
| Tap2 | 13 |  |  |  |

* + - We will insert file one by one. Finally it will become :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tape 1 | 11 | 34 | 45 | 73 |
| Tape 2 | 12 | 34 | 56 | 78 |
| Tap2 | 24 | 34 | 56 | 91 |

**Time complexity :**

* + - The greedy method simply requires us to store the programs in non decreasing order of their lengths. This ordering can be carried out in O(n log n) time using an efficient sorting algorithm.
    - Note that the permutation that yields an optimal solution is the one in which the programs are in non decreasing order of their length.

**Program :**

//Optimal Storage On Tapes

import java.util.\*;

import java.io.\*;

//sorting of prgrms

public class HeapSort

{

private static int[] a;

private static int[] p;

private static int n;

private static int left;

private static int right;

private static int largest;

public static void buildheap(int []a)

{

n = a.length-1;

for(int i=n/2; i>=0; i--)

{

maxheap(a,i);

}

}

public static void maxheap(int[] a, int i)

{

left = 2\*i;

right = 2\*i+1;

if(left <= n && a[left] > a[i])

{

largest = left;

}

else

{

largest = i;

}

if(right <= n && a[right] > a[largest])

{

largest = right;

}

if(largest != i)

{

exchange(i,largest);

maxheap(a, largest);

}

}

public static void exchange(int i, int j)

{

int t = a[i];

a[i] = a[j];

a[j] = t;

int t1= p[i];

p[i] = p[j];

p[j] = t1;

}

public static void sort(int []a0,int[] q)

{

a = a0;

p = q;

buildheap(a);

for(int i=n;i>0;i--)

{

exchange(0, i);

n = n-1;

maxheap(a, 0);

}

}

}

//main class to store sorted prgrms

class store

{

public static void main(String args[])

{

Scanner d = new Scanner(System.in);

System.out.println("Enter no. of Programs:");

int n = d.nextInt();

System.out.println("Enter no. of tapes:");

int nt = d.nextInt();

int [] p = new int[n];

int [] a = new int[n];

double ans = n/(double)nt;

double f = Math.ceil(ans);

int te = (int) f;

int[][] tapes = new int[nt][te];

int i;

System.out.println("Enter the programs with length:");

for(i=0;i<=n-1;i++)

{

System.out.print("Program"+(i+1)+":");

p[i]=d.nextInt();

System.out.println();

a[i]=i+1;

}

HeapSort.sort(p,a);

//printing sorted prgrms

System.out.println("The Best Ordering is");

for(i=0;i<=n-1;i++)

{

System.out.println("Program"+a[i]+":"+p[i]);

}

//method to store in tapes

int k=0;

int x = 0;

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

{

if(x==nt)

{

x = 0;

k++;

}

tapes[i%nt][k] = p[i];

x++;

}

//printing final prgrm in given tapes

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

{

System.out.print("Tape " + (i+1) + " : ");

for(int j = 0;j<te;j++)

{

System.out.print(" "+tapes[i][j]);

}

System.out.println();

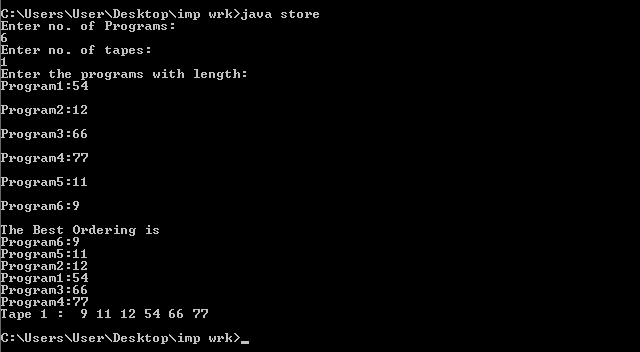
}

}

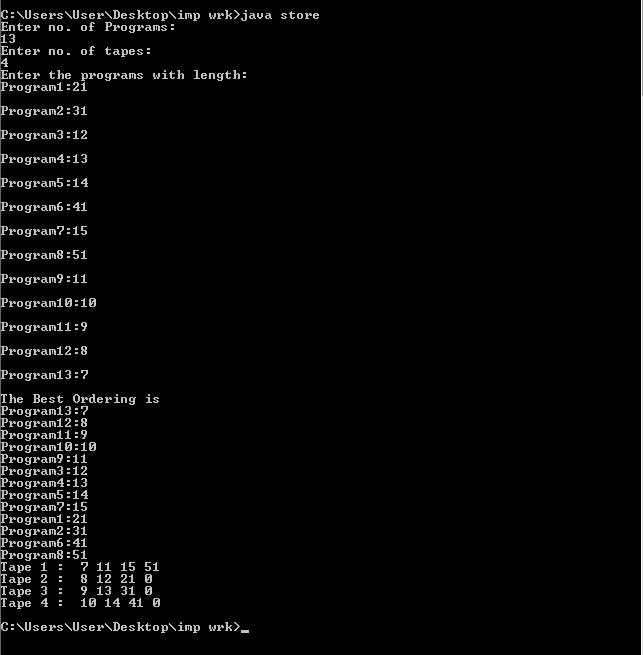
}

**Output :**

\*\* Output for single tape

****

\*\* Output for Multiple Tapes

****