Week-9

1. graph, design an algorithm and implement it using a to implement Floyd - Warshall all pairs shortest path algorithm.

Alm :→ 1) START
2) Input n
3) if i>=n goto step 11
4) if j>=n goto step 10
5) input temp
6) if (temp != "INF")
7) graph[i][j] = stai(temp)
8) else graph[i][j] = 1e9
9) if j<n goto step 4
10) if i<n goto step 3
11) if k>=n goto step 19
12) if i>=n goto step 18
13) if j>=n goto step 17
14) if (graph[i][k] + graph[k][j] < graph[i][j])
15) graph[i][j] = graph[i][k] + graph[k][j]
16) if j<n goto step 13
17) if i<n goto step 12
18) if k<n goto step 11
19) To print shortest path matrix
20) if i>=n goto step 26
21) if j>=n goto step 25
22) if (graph[i][j]>= 1e9) print "INF"
23) else print graph[i][j]
24) if j<n goto step 21
25) if i<n goto step 20
26) STOP

value and weight. You have to design an algorithm and a program to find the list of the selected items such that selected content has weight w and has maximum value. fractions of items ie the items can be broken into smaller you have to carry only a fraction x_i of items i where $0 < x_i < 1$

1) START
2) input n
3) if i>=n goto step 6
4) input items
5) if i<n goto step 3
6) if i>n goto step 11
7) input val[i]
8) job.push_back {val[i] / items[i],
9) (double) (i+1)}
10) if i<n goto step 6
11) input k
12) sort (job.rbegin(), job.rend())
13) profit = 0
14) if i>=n goto step 23
15) if (job[i][1] >= k)
16) profit += k * job[i][0]
17) is.push_back(make_pair(k, job[i][2]))
18) break
19) else profit += job[i][1] * job[i][0]
20) Is.push_back(make_pair(job[i][1], job[i][2])
21) K = k - job[i][1]
22) Print profit
23) Print Item_weight
24) for every element it of Ls
25) Print it.second-it.first
26) STOP

of elements. Assume arr[i] represent the size of file i. Write a program to merge all these files into single file computations. For given two files A and B of size m and n of merging them is O(m+n).

1) START
2) input n
3) if i>=n goto step 6
4) input a[i]
5) if i<n goto step 3
6) i=0
7) if i>=n goto step 10
8) minheap.push(a[i])
9) if i<n goto step 7
10) ans = 0
11) while (minheap.size()>1)
   (i) e1 = minheap.top(), minheap.pop()
   (ii) e2 = minheap.top(), minheap.pop()
   (iii) ans = e1 + e2
   (iv) minheap.push(e1+e2)
12) print ans
13) STOP