[1] A is an $n \times m$ matrix. The ith column of A has weight w_i. Design an algorithm to pick a subset of linearly independent columns of maximum total weight. Assume you have a subroutine to determine if a given subset of columns is linearly independent. How many calls will your algorithm make to this subroutine? Prove the correctness of your algorithm. (2+3)

Algorithm: Maintain separate lists for each linearly independent subset

For each column c_i where $1 \le i \le m$, we make subroutine calls with all the present number of lists to check is subset remains linearly independent on putting the column c_i.

Case-I) No present subset in linearly Independent then create a new list with c_i in it.

Case-II) There are linearly independent subsets present but weight of $c_i \le 0$ Discard c_i

Case III) There are linearly independent subsets present & $wt.(c_i) > 0$ then say these subsets of list that are linear Independent with c_i are $(S_1, S_2, \ldots S_k)$ then duplicate these lists and add c_i to them.

0

At the end we take the subset with max- sum.

We are duplicating at each step because there maybe a case when adding c_i to the set may hinder other higher weight columns to get added. Hence, duplicating to maintain 1 original setting.

Number of routine calls $\Rightarrow 1 + 2 + 4 + 8 + \ldots 2^m$ in worst case when we have to duplicate everything in whole list

$1 + 2 + \ldots 2^m = (2^{m+1} - 1)$ calls.

Correctness: At any moment we maintain all possible linearly Independent subsets. At the end we output the max. wt. subset and hence the solution although exponential will give correct result.

[2] Devise a divide and conquer algorithm to find the minimum and maximum element of a set of n numbers using at most $3n/2$ comparisons. (4)

A-2) Assume ~~let~~ n be even.

then make pairs of 2 ~~with~~ ~~of~~ all elements.
This gives us $\frac{n}{2}$ pairs.

- In each pair find out min and max. This is $\frac{n}{2}$ comparisom. (1)
- Now, we have $n/2$ minimums & $\frac{n}{2}$ maximums.
- Minimum will lie in the set of $\frac{n}{2}$ minimums for sure & maximum will lie in set of $\frac{n}{2}$ maximums.
- Traverse $\frac{n}{2}$ minimums maintaing global minimum which take $\frac{n}{2}$ comparison (2)
- Traverse $\frac{n}{2}$ max. maintaing global max. which will a take $\frac{n}{2}$ comparison (3)

So, by (1) + (2) + (3) = $\frac{3n}{2}$ comparison, we have a global max & global min.

If n was odd, that would pair up (n-1) elements to give $\frac{(n-1)}{2}$ pairs, and we would compa ~~[illegible]~~ las ~~[illegible]~~ element ~~with e both min & max. [illegible] to give~~

~~a total of $\left(\frac{(n-1)}{2} + \left(\frac{n-1}{2}+1\right) + \left(\frac{n-1}{2}+1\right)\right)$ in [illegible] steps~~

~~$\frac{3(n-1)}{2}+2 = \frac{(3n+1)}{2}$ comparison~~

4

with the list of $\frac{n}{2}$ min & max to give total of

$\frac{3(n-1)}{2} + 2 = \frac{3n+1}{2}$ comparison

Scanned with CamScanner

An α-pseudomedian of a list of n distinct values (where $0 < \alpha < 1$) is a value that has at least n^α list elements larger than it, and at least n^α elements smaller than it. The following is a divide-and-conquer algorithm for computing a pseudomedian (that is, an α-pseudomedian for some value of α to be determined later). Assume n is power of 3. If $n = 3$, then simply sort the 3 values and return the median. Otherwise, divide the n items into $n/3$ groups of 3 values. Sort each group of 3, and pick out the $n/3$ medians. Now recursively apply the procedure to find a pseudomedian of these values.

Let $T(n)$ be the number of comparisons used by the preceding algorithm for computing the pseudomedian. Write a recurrence relation for $T(n)$, and solve it. (3)

$T(n) = T(\frac{n}{3}) + \frac{cn}{3}$ [$\frac{cn}{3}$ is due to c time taken to sort each 3 element set)

(n) $\frac{cn}{3}$

$(n/3)$ $cn/9$

$(n/9)$ $cn/27$

$(n/3^i)$ $cn/3^{i+1}$

So Total time is

$\frac{cn}{3}\left[1 + \frac{1}{3} + \frac{1}{3^2} + \cdots \frac{1}{3^{\log_3 n}}\right]$

as $\frac{1}{3} < 1$ we have converging series & to compute order we can extend series till ∞, we have

$\frac{cn}{3}\left[1 + \frac{1}{3} + \text{———} \infty\right]$

$\frac{cn}{3} \times \frac{1}{2/3} = \frac{cn}{2}$

$\boxed{T(n) = \frac{cn}{2}}$

3

Let $E(n)$ be the number of values that are smaller than the value found by the preceding algorithm. Write a recurrence relation for $E(n)$, and hence prove that the algorithm does return a pseudomedian. What is the value of α? (3)

$E(n) = E(\frac{n}{3}) + 1$ (1 because, at each step we can surely put 1 max & 1 min element)

Solving this recurrence we have $E(n) = \log_3 n$

$n^\alpha = \log_3 n$

$\alpha = \log_n(\log_3 n)$

0

[4] Let E be a set of m linear equations of the form $x_i = x_j + c_{ij}$ over the variables $x_1, \ldots, x_n (c_{ij} \in \mathbb{Z}$ for all $1 \le i, j \le n)$. Devise an $O(m)$ algorithm for determining whether the equations in E are consistent, that is, whether an assignment of integers can be made to the variables so that all of the equations in E are satisfied. (5)

~~Divide up the set of m equations into $m/2$ each. Then find recursively whether these set of $m/2$ systems are consistent. Then Take any 1 equation from left & 1 equation from right and check if they are consistent. If yes, then the system in E is consistent, else no.~~

~~Time complexity: $T(m) = 2T(\frac{m}{2}) + C$~~ ~~[Assuming checking consistency in 2 equations is constant C time]~~

Write the eq.^{n} as $x_i - x_j = c_{ij}$

Sum up all the equations.

If L.H.S is 0 and RHS $\neq 0$ then inconsistent else a solution is possible.

Summing up all equations takes $O(m)$ time.

Proof: ~~If LHS ≠ RHS then~~ There are no coefficients attached to x_i & x_j due to which the solution is possible.

If LHS(0) ≠ RHS the clearly system of equations is inconsistent but if LH.S is not 0 then L.H.S has an equation left.

Then all those pairs that satisfy the equa[tion] must satisfy the system.

0