Minimum cost to remove all elements

Given N array elements, at every step, minimum you can remove any element from array cost to remove any element is sum of all the elements

{1, 1, 2, 3}

cost 2: {1, 1, 2} → 1+1+2 = 7

cost 1: {1, 1, 3} → 1+1 = 2

cost 1: {1, 3} → 1 = 1

Ans = 7 + 5 + 1 = 13

TC:

Cost 4: {1, 1, 2, 3} = 7

Cost 2: {1, 1, 3} = 3

Cost 1: {1, 3} = 1

Ans = 11

optimal

{3, 5, 1, -3} Ans = 2

Cost 5: {3, 5, 1, -3} = 3+5+1-3 = 6

Cost 3: {3, 5, 1, -3} = 3+1-3 = 1

cost 1: {1, -3} = -2

cost -3: {-3} = -3

7 - 5 = 2

{a, b, c, d}

d → a+b+c+d

c → a+b+c

b → a+b or 1st

a → a

Max found + min of the

= min of 1st & 2

$d_4 = 3b + 2c + a$ (Minimum)

A → [ ] find Max ×1

one time

A → [ ] find ×2

TC: $O(n^2)$

[1, 2, 3, 4, 5]

a[0] ... a[n-1]

sort (A) ← increasing order

cost = 0; M

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

cost += M*A[i]

M -=;

cost = 0

sort N * A[0]

cost = N*A[0] + (N-1)*A[1]

cost = N*A[0] + (N-1)*A[1] + (N-2)*A[2]

TC: $O(N \log N)$

SC: $O(1)$

Count of unique Triplets

Given N array elements, calculate no of unique triplets that is equal to the number of elements < a[i]

{ count of unique Triplets }

Ex: {7, -5, 3, 0, -10, 8}

count = 3

CEIL: {-5, 1, 0, 0, 0, 0}

output: 3 {-5, 3, 5, 4}

arr[i]: {-10, -5, 2, 1, 3, 4, 0, 5, 8}

CEIL: {0, 1, 3, 2, 5, 4, 1, 6, 7}

output: 4

CEIL = {0}

for i=1; i<N; i++

for j=i-1; j>=0; j--

if (Ceil[j] < arr[i])

CEIL[i] += 1;

count = 0

for i=1; i<N; i++

if (CEIL[i] == arr[i])

count ++

TC: $O(n^2)$

SC: $O(n)$

Ex. arr: {-10, -5, 0, 3, 3, 3, 5} i=3

CEIL: 0 1 2 3 3 3 5 i

INPUT ARR

CEIL = {0,1,2,3}

1) Sort array

2) iterate through array & find all element

Array [i] = A[i]

sort (A)

count = 0

for i=0; i<N; i++

if (i == A[i])

count += 1;

return count

main () {

count = No of count value;

cout << find count(A);

}

Ex. CEIL: {0, 0, 2, 2, 4, 4, 6}

Ind: 0 1 2 3 4 5 6

CEIL[i] != indexes

{-3, 0, 1, 5, 5, 5, 5, 5, 8, 8, 8, 11}

CEIL

Ind

output = 7

{-10, 1, 3, 7, 8, 8}

→ CEIL {0 1 2 3 4 5}

TC: $O(N)$ SC: $O(N)$

{-10, 1, 1, 1, 3, 3, 7, 7, 14}

Ind {0, 1, 2, 3, 4, 5, 6, 7, 8}

CEIL {0, 1, 1, 1, 4, 4, 7, 7, 8}

1) CEIL value for all repeated occurrency are same

2) for third occurrency for first time CEIL[i] == i

1) A[i] != A[i-1] // not repeated

A[i] == A[i-1] → use a while

count

2) A[i] != A[i-1] (occurrency for first time)

sort (A)

CEIL = 0

for (i=1; i<N; i++){

if (a[i] != A[i-1])

if (CEIL == A[i])

count += 1;

}

if (CEIL[0] == 0)

count += 1;

return count

{-10, 1, 1, 1, 3, 3, 7, 7, 14}

i=1 CEIL = 1 , arr[1] = 1 , count = 1

i=2 CEIL = 1 , arr[2] = 1 , count = 2

i=3 CEIL = 1 , arr[3] = 1 , count = 3

i=4 CEIL = 4 , arr[4] = 3 , count = 3

i=5 CEIL = 4 , arr[5] = 3 , count = 5

i=6 CEIL = 7 , arr[6] = 7 , count = 6

i=7 CEIL = 7 , arr[7] = 7 , count = 7

i=8 CEIL = 8 , arr[8] = 14 , count = 8

return 7

i =

CEIL =

Ans = 10

Q) sort array on the basis of number of factors

If number of factors are equal sort them on the value

→ 1 3 5 6 9 12 10

1 2 2 3 3 6 4

1, 3, 5, 6, 9, 10, 12 output

12 → 1, 2, 3, 4, 6, 12

10 → 1, 2, 5, 10

6 → 1, 3, 2

factor

1 3 5 9 6 15 12

f 1 2 2 3 4 4 6

21 → 1, 3, 7, 21

12 → 1, 2, 3, 4, 6, 12

1, 2, 7, 7, 14

14 → 1, 2, 7, 14

{8, 12, 15, 16, 14, 21, 24}

f: 4 6 4 5 4 4 8

output: {8, 14, 15, 21, 16, 12, 24}

15 15

→ sort (A.begin, A.end, mycomp);

Compare before

15 13

compare after

13 < 15

fa = 4

fb = 4

return false;

bool mycomp (int a, int b) {

int fa = factors(a);

int fb = factors(b);

if (fa < fb) {

return true;

}

else if (fa == fb && a < b)

return true;

else

return false;

}

int fa = 4

int fb = 6

if (fa < fb)

return true

else

return false

$O(\log(A.D))$

$O(N \log N)$

sort A.begin(), A.end(), mycomp);

return A;

TC: $O(N \log N * \text{sqrt}(A.D))$

for (i=1; i<=sqrt(A); i++) {

if (A%i == 0) {

i, A/i both are factor

if (i == A/i)

fact ++;

else factor += 2;

}

}

A.sort() ;

int

contiguous memory

4 4 9

A.begin() A.end()

Q) Sort element on the number of digits that are present, if digits are same the bigger value should come first

10 , 12 , 123 , 1 , 111

2 2 3 1 4

output 1 , 12 , 10 , 123 , 1111