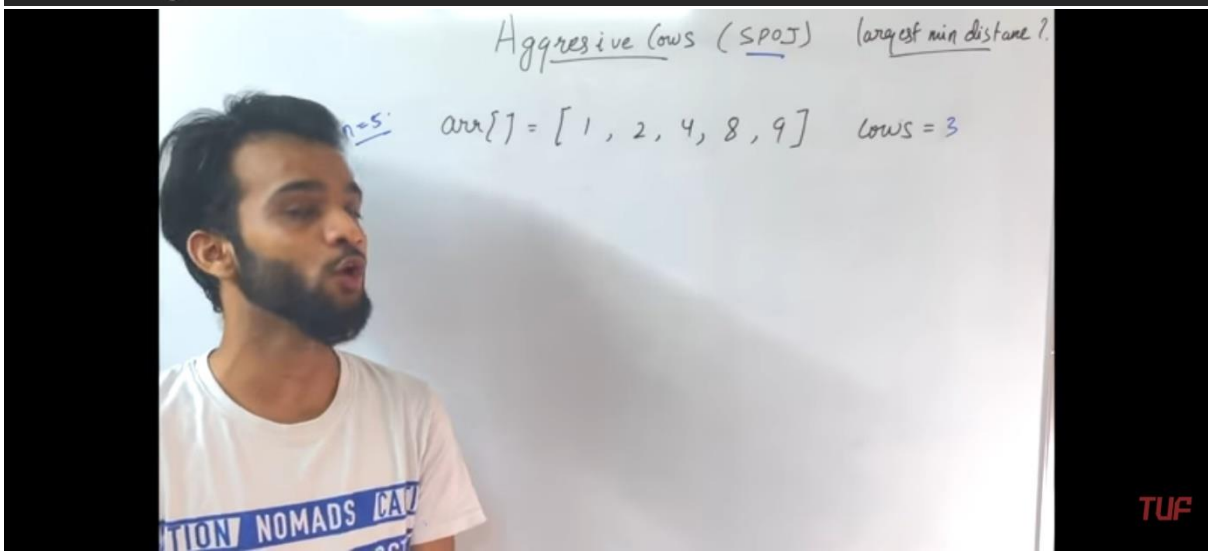
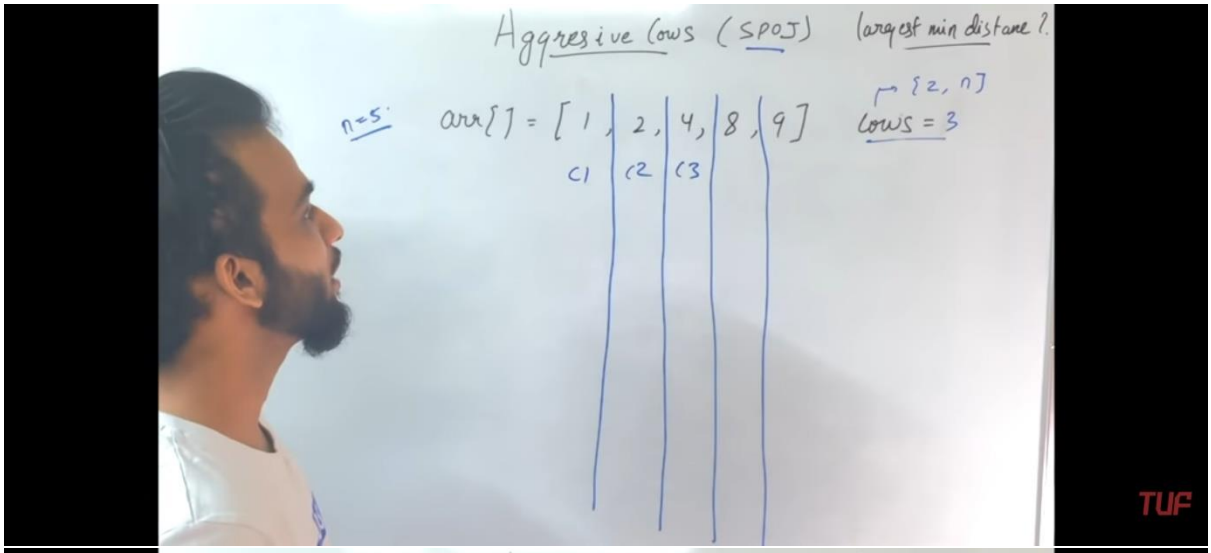
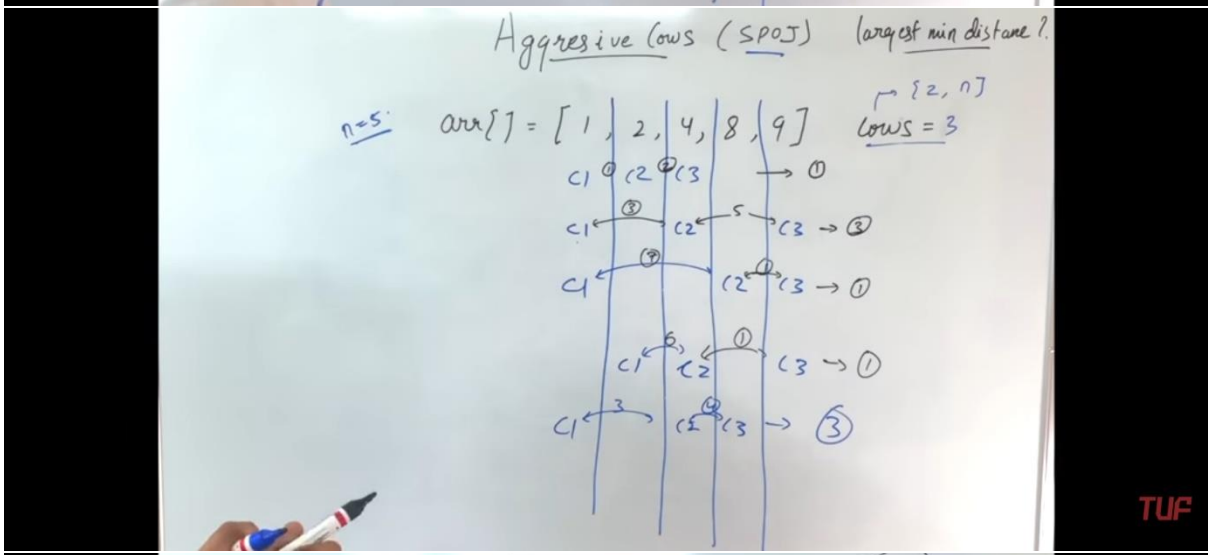
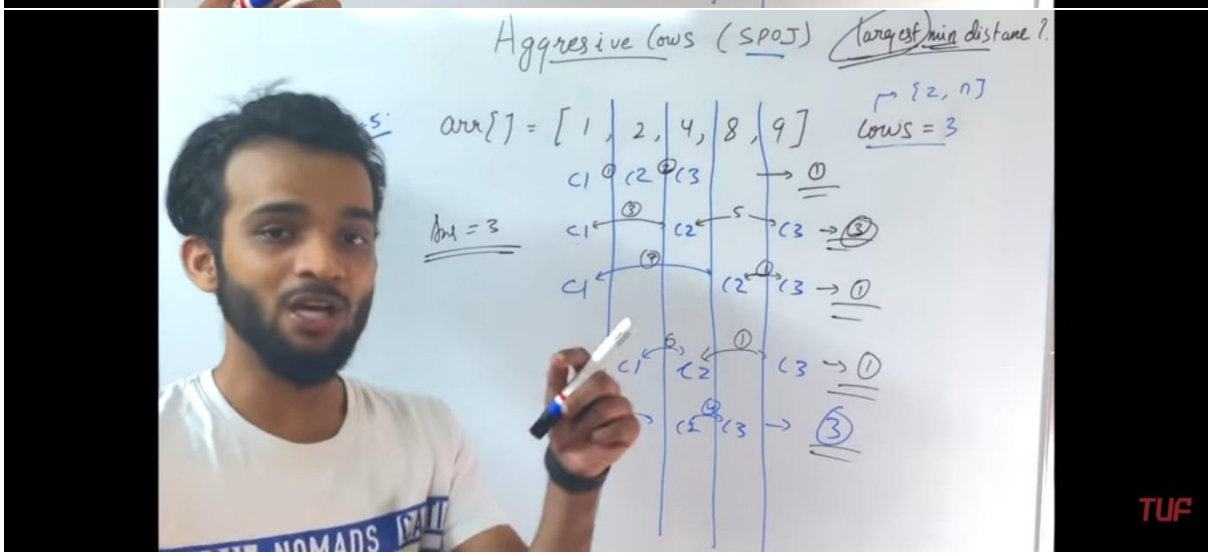
similar question on leetcode ->

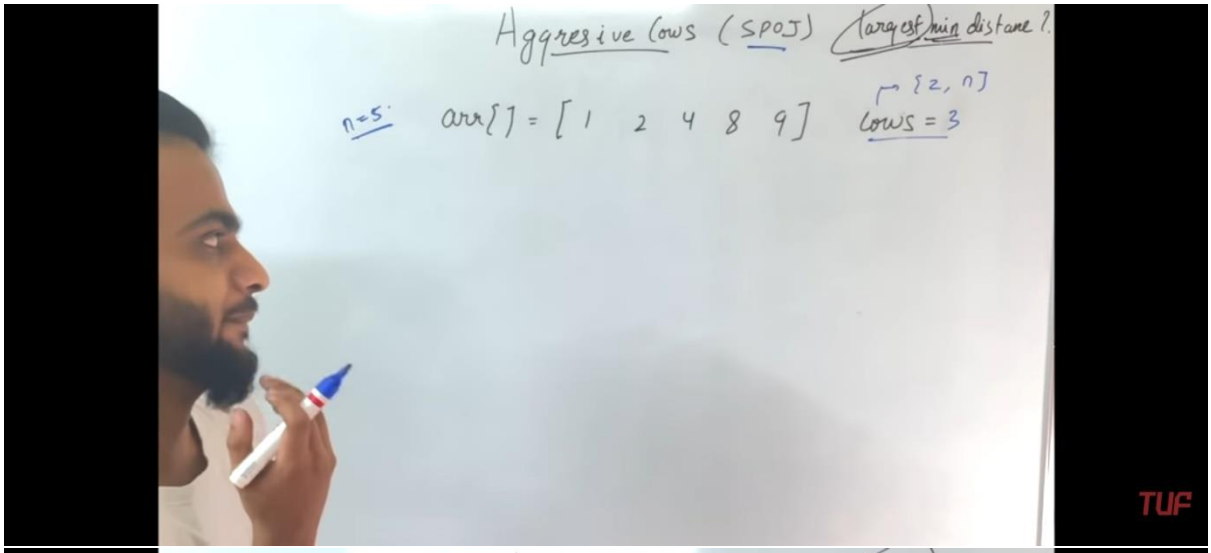
1552. Magnetic Force Between Two Balls
1870. Minimum Speed to Arrive on Time
875. Koko Eating Bananas
1011. Capacity To Ship Packages Within D Days
1283. Find the Smallest Divisor Given a Threshold
1482. Minimum Number of Days to Make m Bouquets
2064. Minimized Maximum of Products Distributed to Any Store

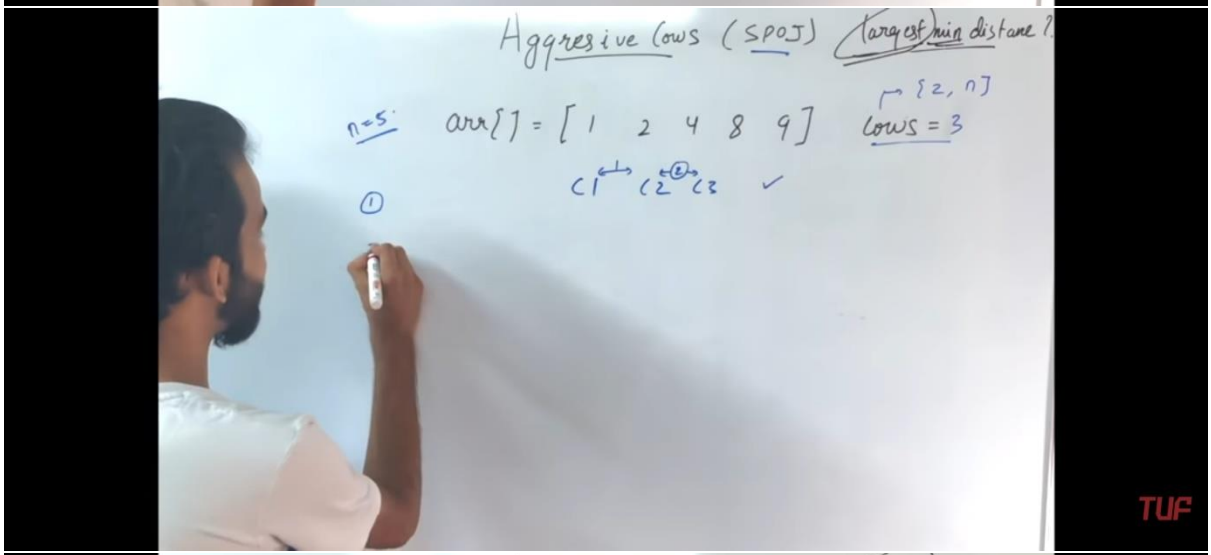


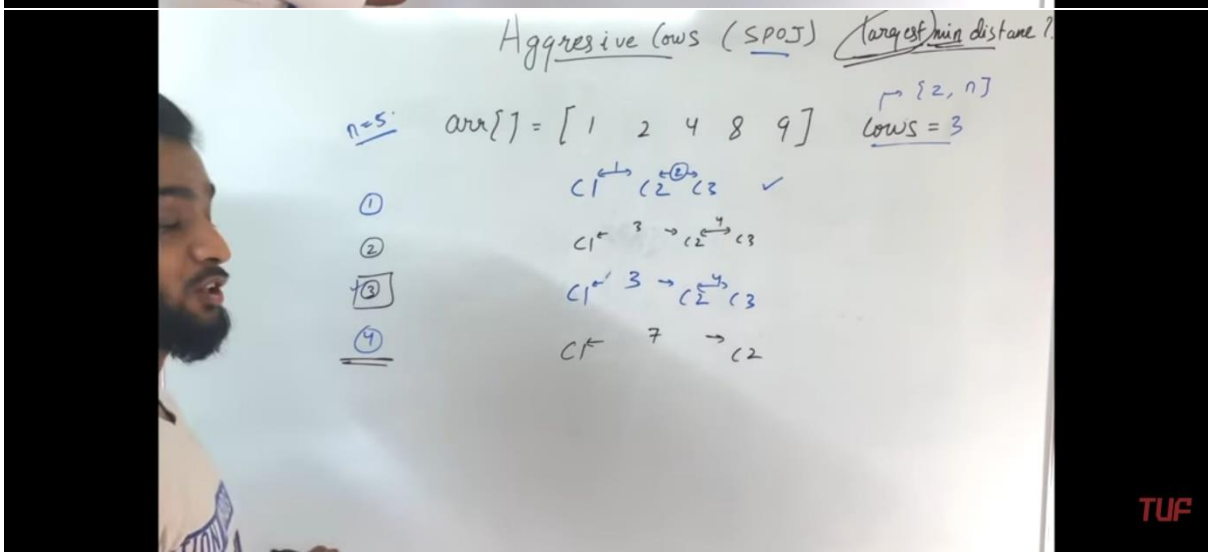

Aggresive Cows (SPOJ)
largest min distance ?
n=5
arr[] = [1, 2, 4, 8, 9]
cows = 3
[2, n]
C1 C2 C3
TUF

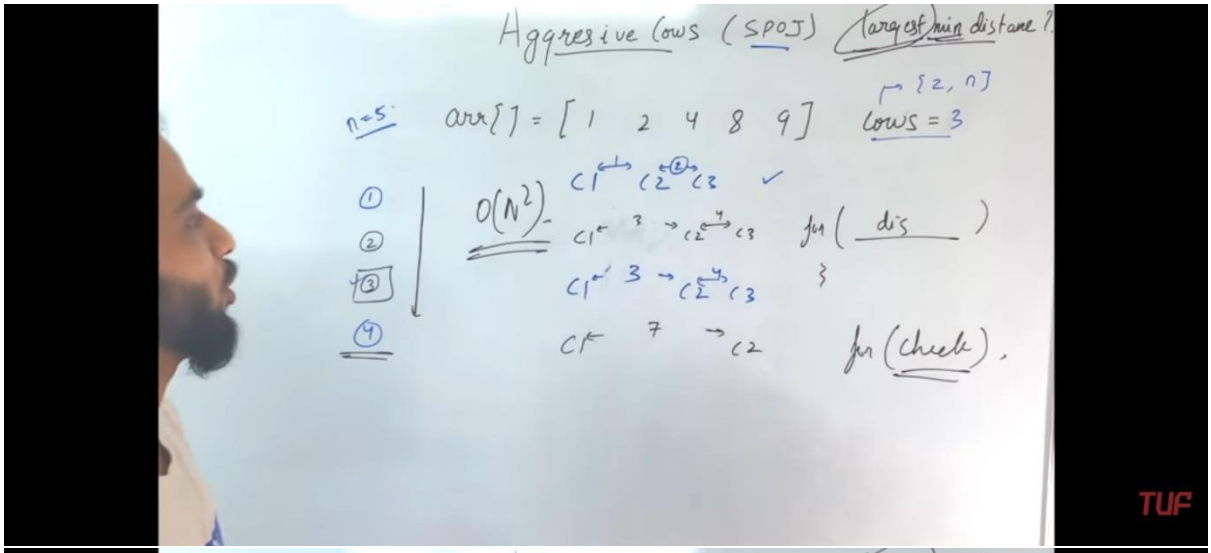

Aggresive Cows (SPOJ)
largest min distance ?
n=5
arr[] = [1, 2, 4, 8, 9]
cows = 3
[2, n]
C1 C2 C3
TUF

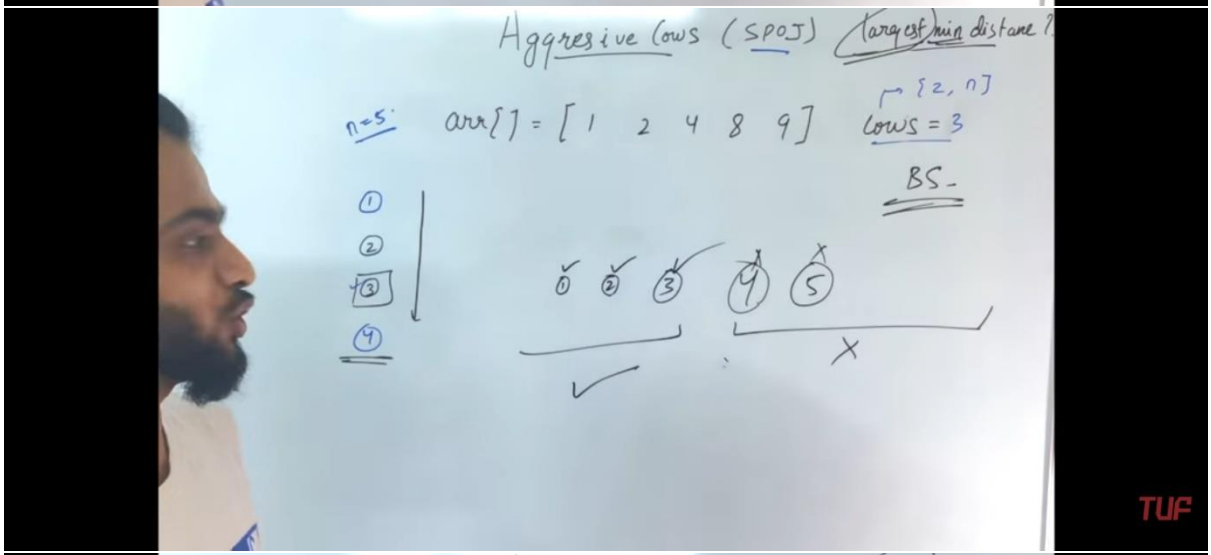

Aggresive Cows (SPOJ)
largest min distance ?
arr[] = [1, 2, 4, 8, 9]
cows = 3
[2, n]
ans = 3
C1 C2 C3
TUF

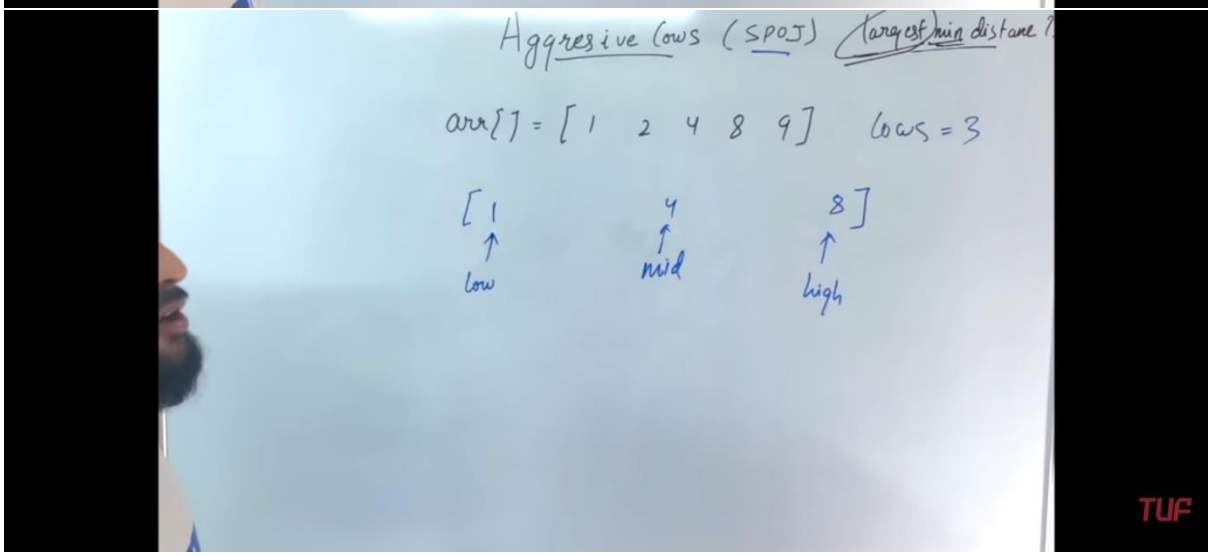

Aggresive Cows (SPOJ) largest min distance?
n=5 arr[] = [1 2 4 8 9] cows = 3
[2, n]
TUF

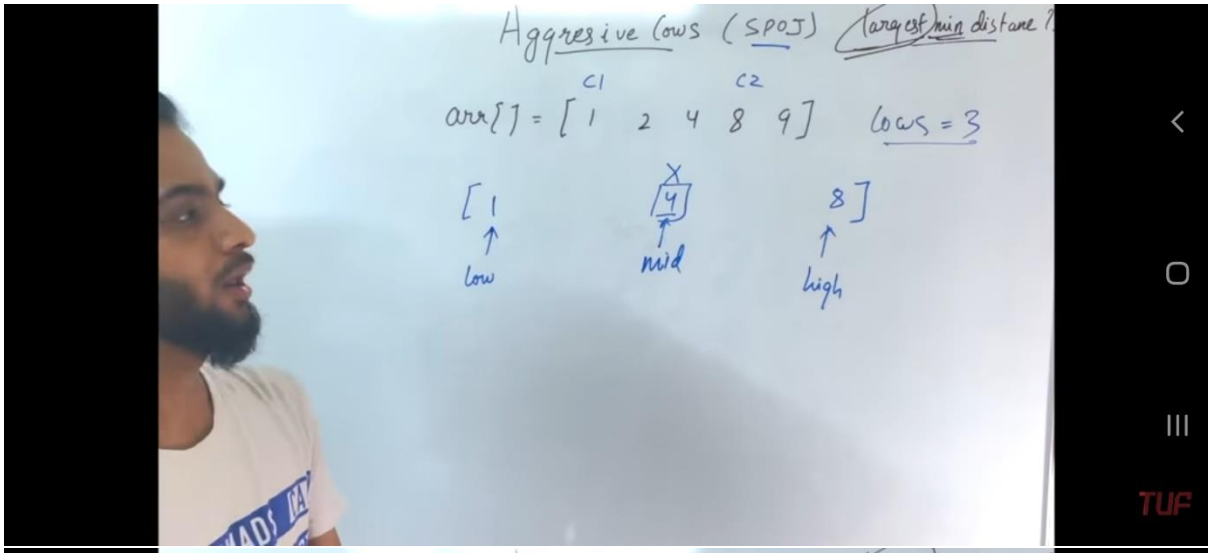

Aggresive Cows (SPOJ) largest min distance?
n=5 arr[] = [1 2 4 8 9] cows = 3
[2, n]
① C1 C2 C3 ✓
TUF

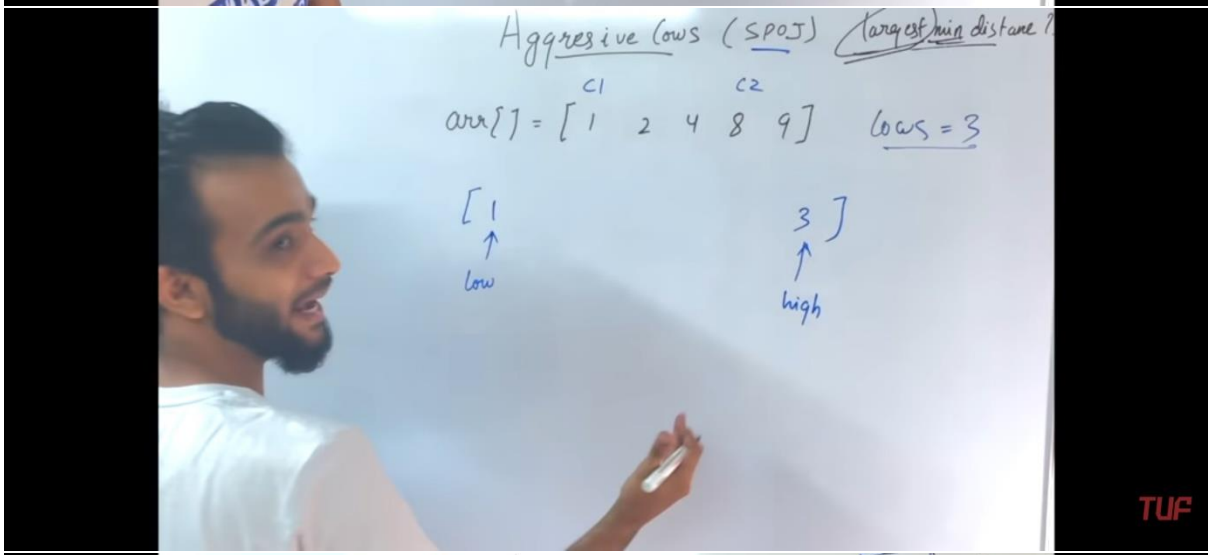

Aggresive Cows (SPOJ) largest min distance?
n=5 arr[] = [1 2 4 8 9] cows = 3
[2, n]
① C1 C2 C3 ✓
② C1 3 C2 C3
③ C1 3 C2 C3
④ C1 7 C2
TUF

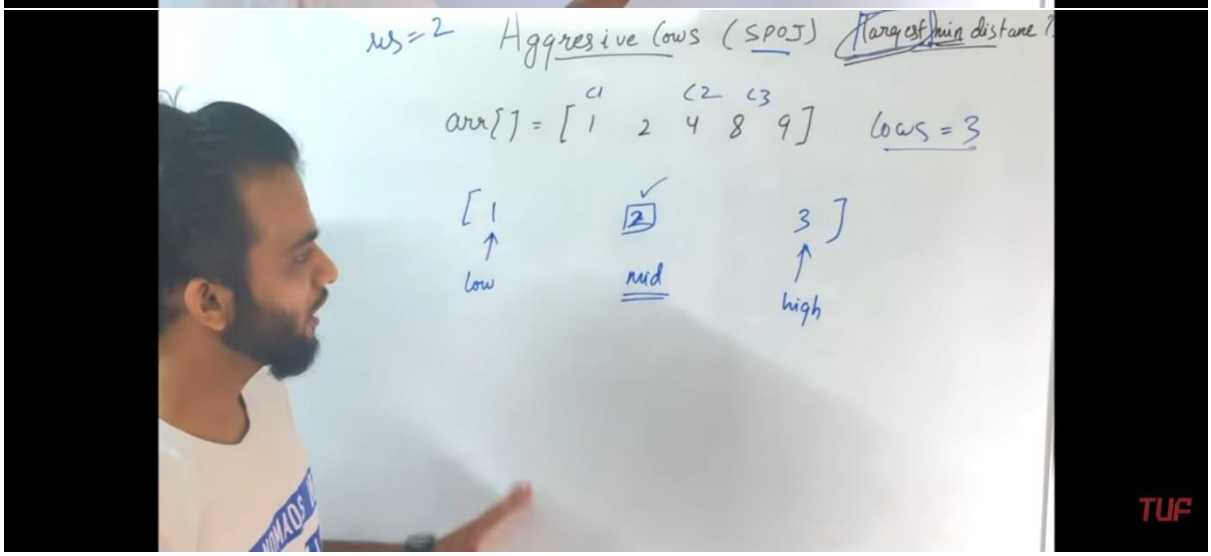

Aggresive Cows (SPOJ) (largest min distance)
n=5: arr[] = [1 2 4 8 9] cows = 3
[2, n]
1
2
3
4
O(N^2)
C1 C2 C3
C1 C2 C3
C1 C2 C3
C1 C2
fun(dis)
{
fun(check).

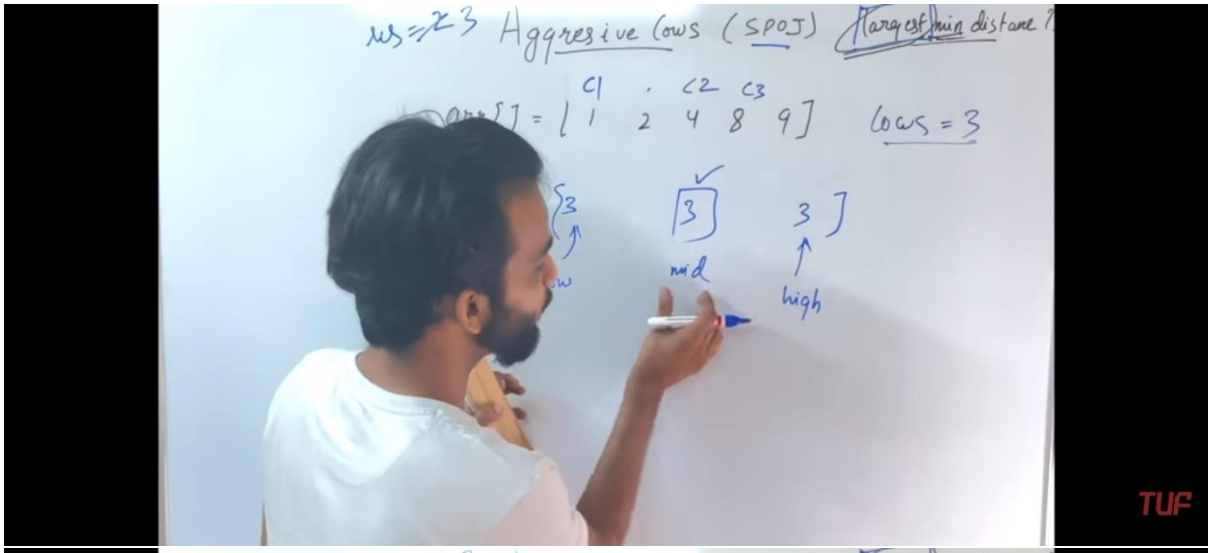

Aggresive Cows (SPOJ) (largest min distance)
n=5: arr[] = [1 2 4 8 9] cows = 3
[2, n]
BS-
1
2
3
4
0 2 3 4 5
X

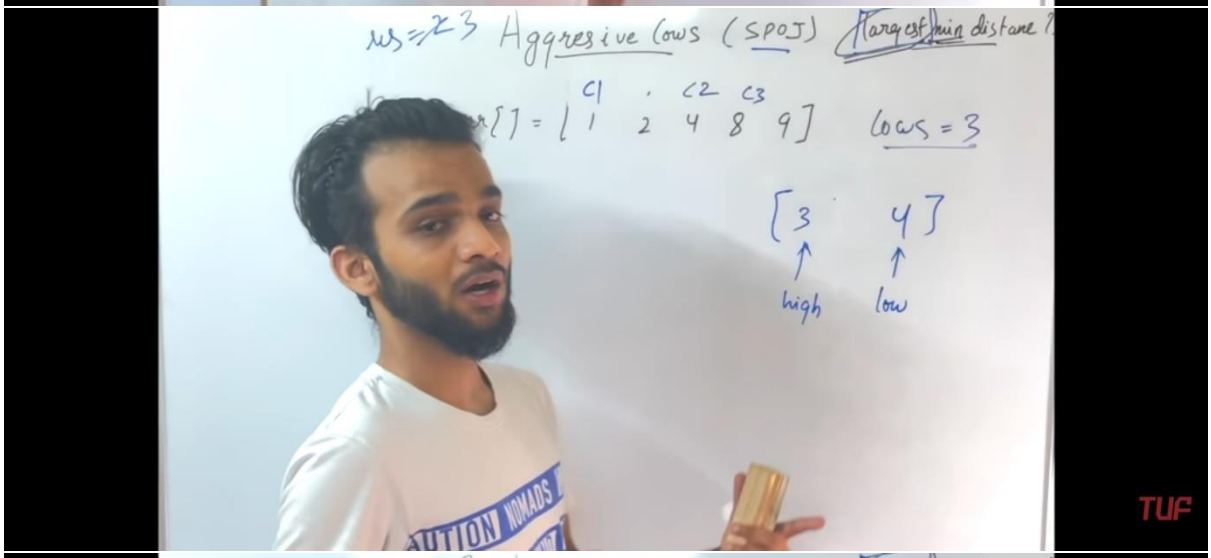

Aggresive Cows (SPOJ) (largest min distance)
arr[] = [1 2 4 8 9] cows = 3
[1 4 8]
low
mid
high

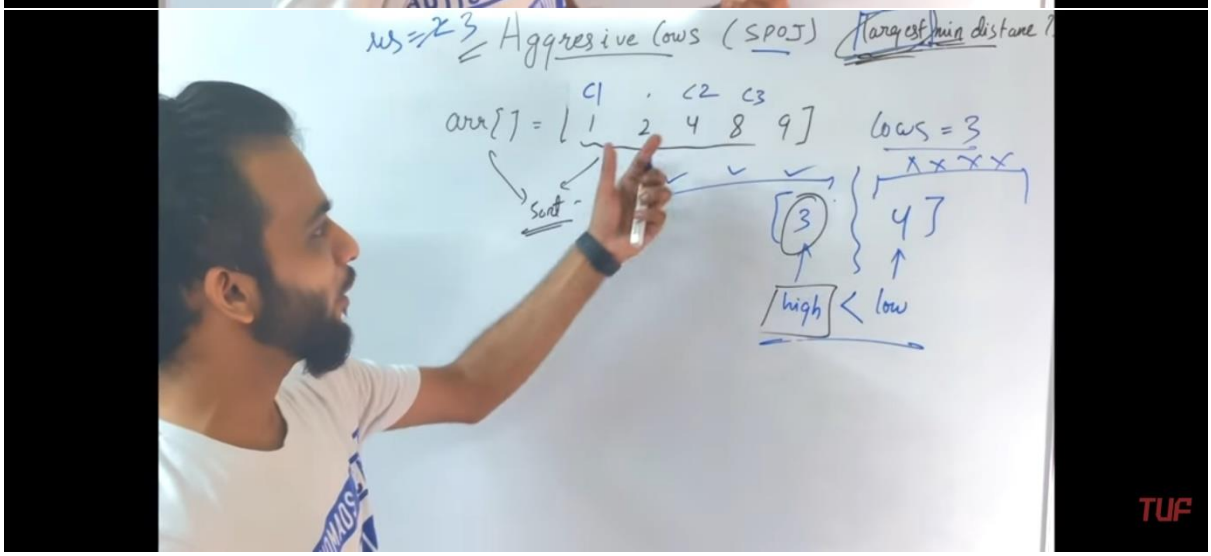

Aggresive Cows (SPOJ) largest min distance
arr[] = [1 2 4 8 9] cows = 3
c1
c2
[1 4 8]
low
mid
high

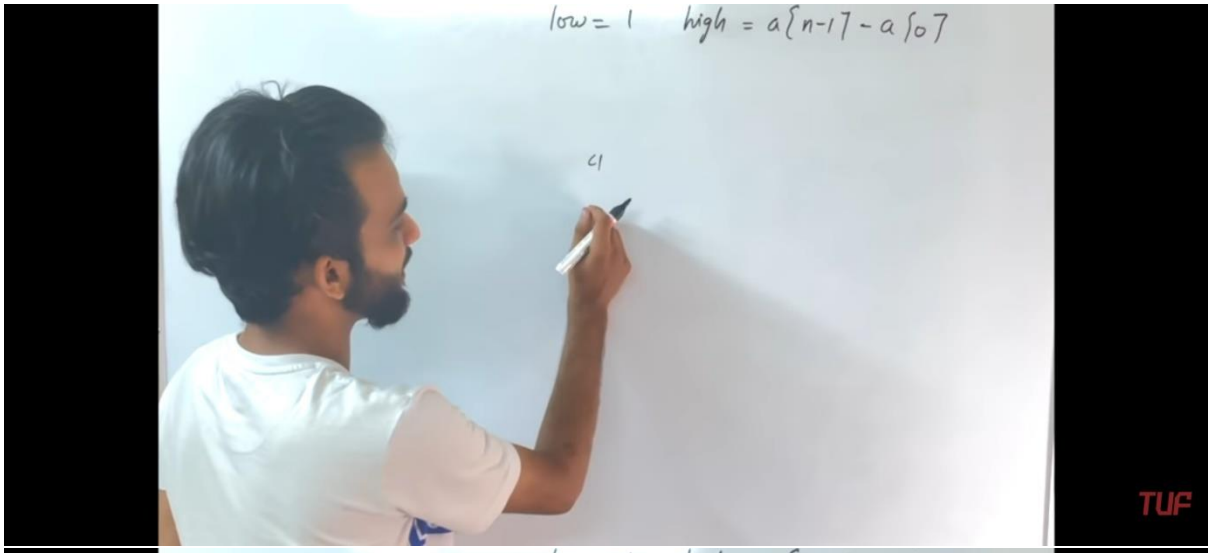

Aggresive Cows (SPOJ) largest min distance
arr[] = [1 2 4 8 9] cows = 3
c1
c2
[1 3]
low
high

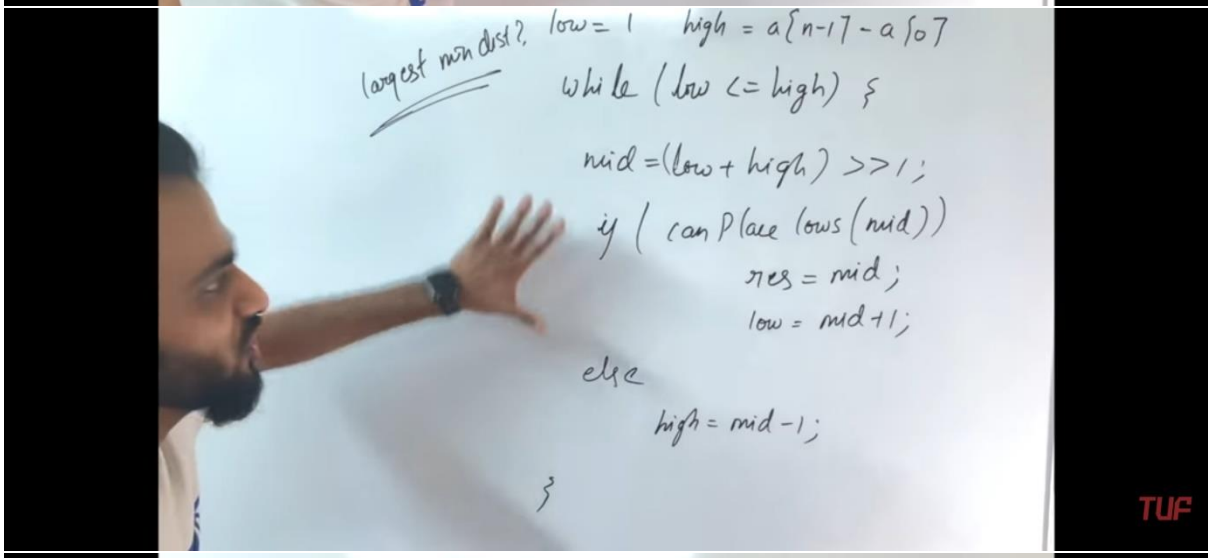

ans = 2
Aggresive Cows (SPOJ) largest min distance
arr[] = [1 2 4 8 9] cows = 3
c1
c2
c3
[1 2 3]
low
mid
high


Aggresive Cows (SPOJ)
largest min distance
c1 c2 c3
arr[] = [1 2 4 8 9]
cows = 3
[3 3]
mid
high
low
TUF

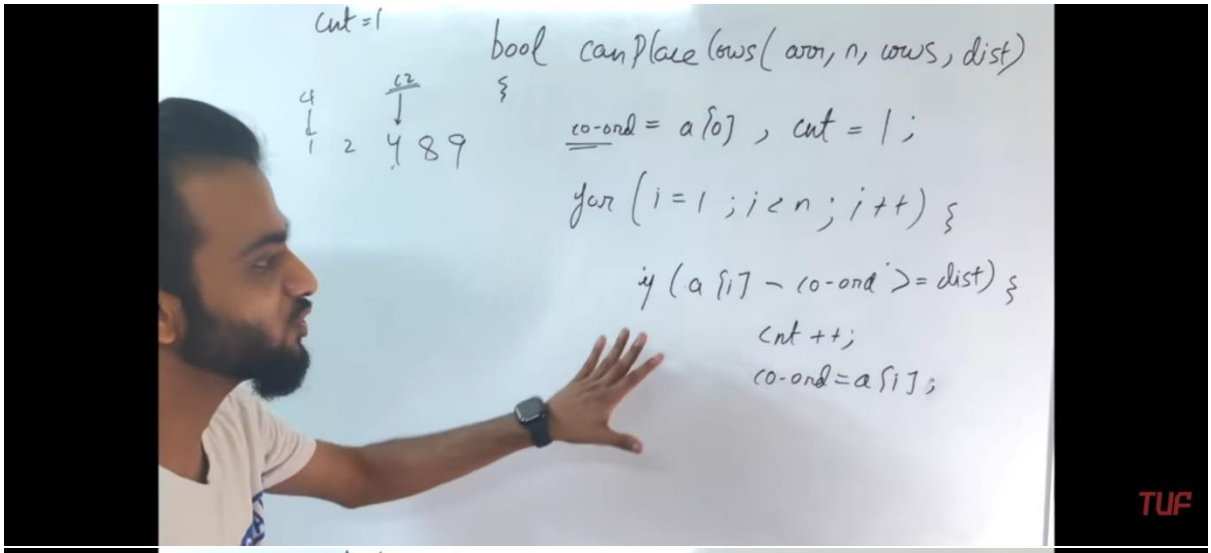

Aggresive Cows (SPOJ)
largest min distance
c1 c2 c3
[1 2 4 8 9]
cows = 3
[3 4]
high
low
TUF

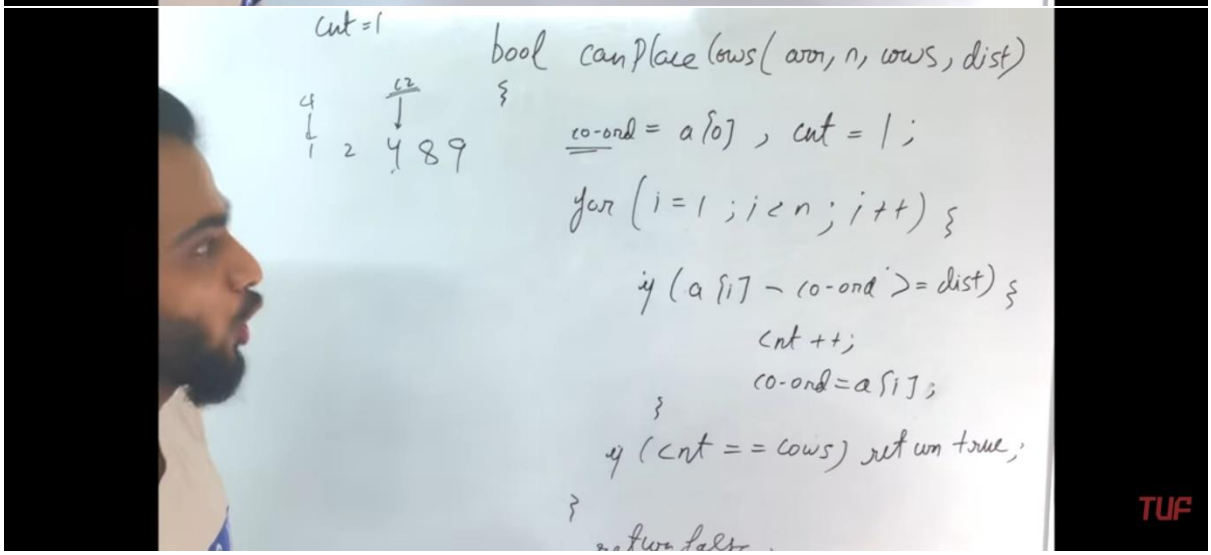

Aggresive Cows (SPOJ)
largest min distance
c1 c2 c3
arr[] = [1 2 4 8 9]
cows = 3
Sort
[3 4]
high < low
TUF

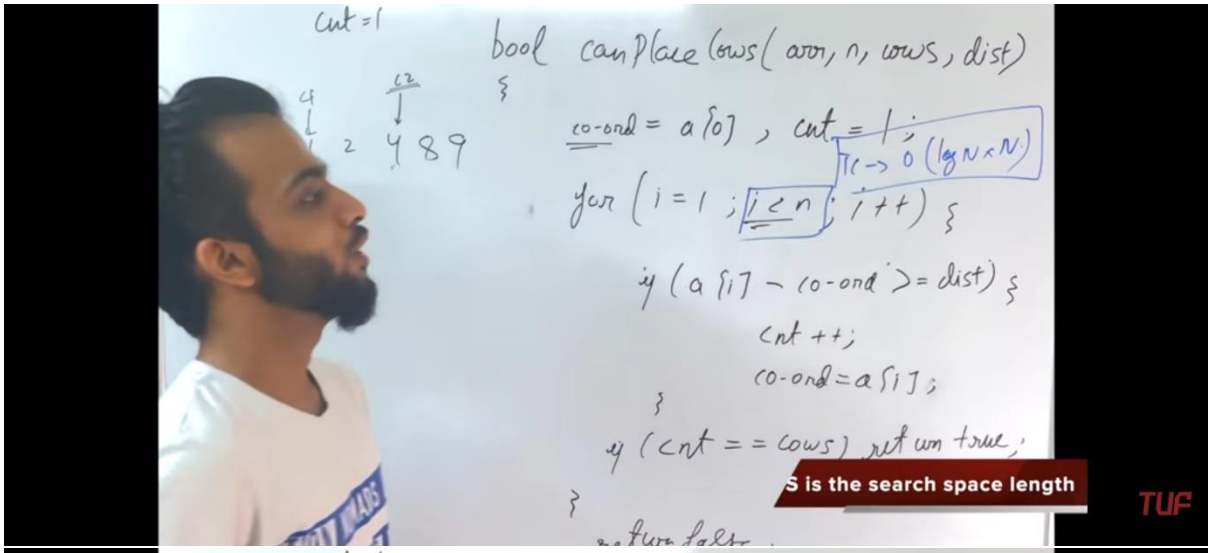

low = 1 high = a[n-1] - a[0]
cl
TUF

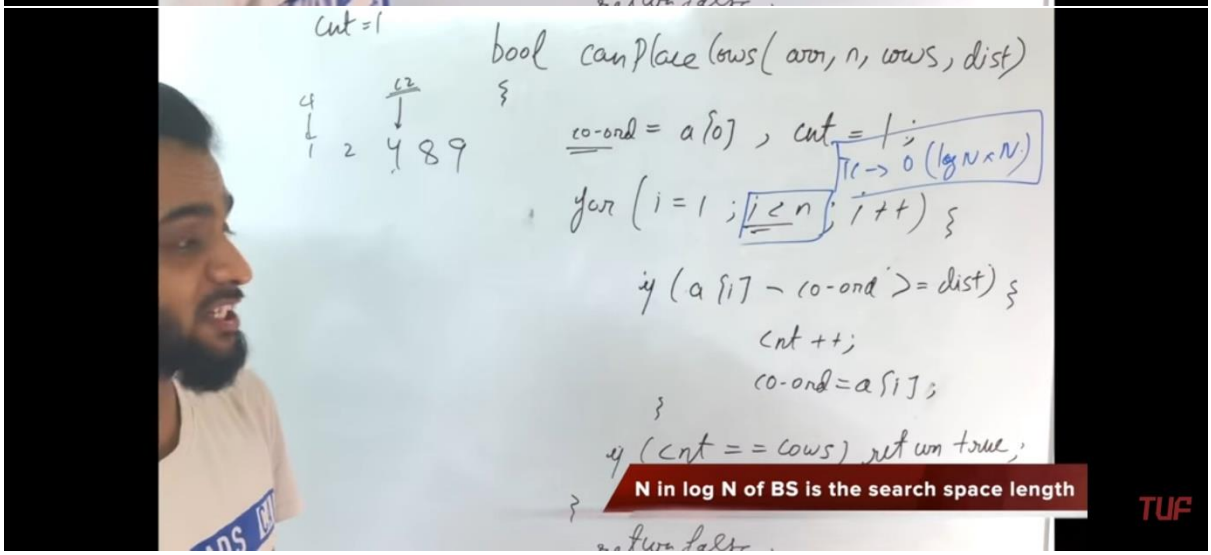

largest min dist?
low = 1 high = a[n-1] - a[0]
while (low <= high) {
mid = (low + high) >> 1;
if (canPlace(cows(mid))
res = mid;
low = mid+1;
else
high = mid-1;
}
TUF


bool canPlace(cows(arr, n, cows, dist)
TUF


cnt = 1
bool canPlace(ows(arr, n, ows, dist)
{
co-ord = a[0], cnt = 1;
for (i = 1; i < n; i++) {
if (a[i] - co-ord >= dist) {
cnt++;
co-ord = a[i];
TUF


cnt = 1
bool canPlace(ows(arr, n, ows, dist)
{
co-ord = a[0], cnt = 1;
for (i = 1; i < n; i++) {
if (a[i] - co-ord >= dist) {
cnt++;
co-ord = a[i];
}
if (cnt == cows) return true;
}
TUF


cnt = 1
bool canPlace(ows(arr, n, ows, dist)
{
co-ord = a[0], cnt = 1;
for (i = 1; i < n; i++) {
if (a[i] - co-ord >= dist) {
cnt++;
co-ord = a[i];
}
if (cnt == cows) return true;
}
return false;
i -> 0 (log N x N)
S is the search space length
TUF


cnt = 1
bool canPlace(ows(arr, n, ows, dist)
{
co-ord = a[0], cnt = 1;
for (i = 1; i < n; i++) {
if (a[i] - co-ord >= dist) {
cnt++;
co-ord = a[i];
}
if (cnt == cows) return true;
}
return false;
i -> 0 (log N x N)
N in log N of BS is the search space length
TUF