the fill freq array using 2-pointer

ind = [2,4,1,0,3]

10/22/23

CLASS - 10

i/p	o/p
25	5
64	8

Given a number find the sqrt

B.F

↑ Not needed (Code & check)

```
for(int i=1; i<=N; i++)
{ if (i*i==N)
  { return i;
  }
}
```

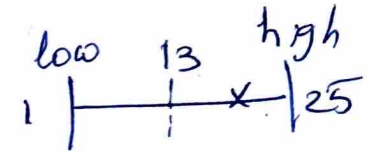
O(√N, 1)

Binary Search

Time Space.

O(logN)(1).

lo	hi	mid	
1	25	13	13×13 > 25
1	(high = mid-1) 12.	6	6×6 > 36
1	5	3	3×3 < 25
4	5	4	16 < 25
5	5	5	25 == 25 → return mid



Code

```
low=0, high=N
while (low<=high)
{ mid=(low+high)//2
  if (mid*mid==N):
      return mid
  if (mid*mid>N)
      high=mid-1
  else
      low=mid+1
}
```

Cube root → mid * mid * mid

p^{th} root → (mid * mid * --- p times)

↙ Pocen (mid, P) → $\log_2 P$

Better to use internal Lnc

p^{th} root → Binary Search → log n

Total comp ⇒ $\log_2 P * \log_2 N$

	$\sqrt{N}$	logN
4	2	2
16	4	4
10^4	10^2	$\log_2 10^4 = \log_2 10 + \log_2 2^{10}$ (13) $\cong 3 + 10 \cong 13$
10^{18}	10^9	$\log 10^{18} = \log 10^9 + \log 10^9$ $= \log 2^{30} + \log 2^{30}$ $= 60$.
	(10^9 iterations)	(60 iterations)

$a^6 = a^{6/2} \cdot a^{6/2} = a^3 + a^3 = a^6$

$a^5 = a^{5/2} \cdot a^{5/2} = a^2 \cdot a^2 = a^{4+1}$ $a^4 \cdot a$.

So

even	$a^{n/2} \cdot a^{n/2}$
Odd	$a \cdot a^{n/2} \cdot a^{n/2}$

PROBLEM

	0	1	2	3	4	5	6
a_N :	5	12	-6	24	19	10	7

$k = 29$

$a + b == k$

$i \neq j$

After Sorting

-6	5	7	10	12	19	24
↑ P_1						↑ P_2

$-6 + 24 = 18 < 29$

$\Rightarrow \uparrow P_1$ (2-ptr app)

Soln

1) Sort, 2 ptr

$N \log N + N, 1$

2) B.F

$(N^2, 1)$

3) B.S

$N \log N + N \log N$

Binary Search

→ fix an element

→ Search for $(k - a)$ in the array in range i (i^{st}-fixed) start 2nd from $i+1$

```
for(int i=0; i<N; i++)
  { int b = k - arr[i];
    if (bs(arr, i+1, N-1, b))
        return true;
  }
return false;
```

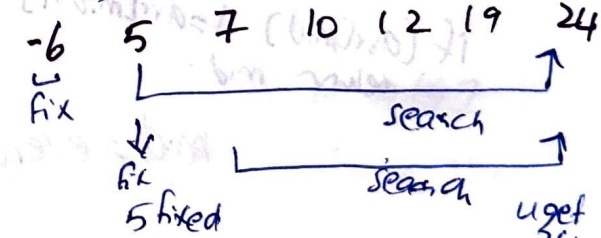
Complexity

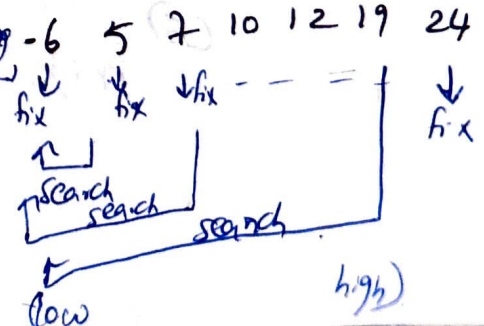
$N \log N + N \log N$

[Sort + loop B.S.]

(or)

```
for(int i=1; i<N; i++)
  { int b = k - arr[i];
    if(bs(arr, 0, i-1, b))
        return true
  }
return false
```





PROBLEM

ARR : 2 2 5 5 12 12 17 24 24 30 30

Given a sorted array, everything is repeated twice except one.

Solutions

1) B.F $N^2, 1$

2) XOR $N+N, 1$

3) ~~B.S $\log N, 1$~~

3) $N/2, 1$ (i+=2)

for 11 elements we are only checking 5 elements at max even if ele is at the end.

③
```
i = 0
while (i < N)
if (arr[i] == arr[i+1])
        i += 2
else return arr[i].
```

④ Binary Search

0	1	2	3	4	(5)	6	7	8	9	10
2	2	5	5	12	(12)	17	24	24	30	30

if (ar[mid] != ar[mid-1] && ar[mid] != ar[mid+1])
return mid.

mid = even

- arr[i] == arr[i+1] → low = mid+2
- arr[i] == arr[i-1] → high = mid-2

⑤ mid = odd

- arr[mid] == ar[mid-1] → low = mid+1
- arr[mid] == ar[mid+1] → high = mid-1

Tracing

low=0, high=10, mid=5

mid=5=odd → arr[5] == a[4]. → low = mid+1=6.

mid=(6+10)/2=8=even → arr[8] == arr[7]. → high = mid-1 = 8-1=7.

mid = 6+7/2 = 13/2 = 6 → arr[6] != arr[7] != a[8] → return arr[mid]

Initial Check

if mid == 0

mid == N-1

return.

PROBLEM (MIN-MAX PROBLEM) → (Minimizing the maximum result)

1) Contiguous tasks — N-1 places to put partition, partitions

2) Minimum time — K people we do p-1 partitions.

N-1 C K-1

10 | 20 | 5 27 40 60 15 → max = 147

we 10 and 20 first

147

10 | 20 5 | 27 40 60 15 → max = 142.

142

10 | 20 5 | 24 |

```
int ans = INT_MAX
void partitioning (int arr[], int N, int k, int idx,
                                              int cans)
  if (k==1) || idx==n)
  {
    for (int i=idx; i<N; i++)
        y+=a[i].
    cans = max(cans, y)
    ans = min(ans, cans)
    return;
  }
  int x=0.
  for(int i=idx; i<N; i++)
    {
      x+= a[i]
      partitioning (arr, N, k-1, i+1, max(cans,x))
    }.
```

Complexity : $N-1 C_{k-1} * N$.

2nd Approach

10 20 5 27 40 60 15

low = max(arr) , high = sum(arr)
60 177

mid = $\frac{177+60}{2} = \frac{237}{2} = 118$.

Sum = 0+10 < 118 → 10+20=30 < 118 → 35 < 118 → 62 < 118 <

ans = 118. high = mid − 1

low	high	mid	
60	117	88	→ True
60	88	74	→ True.
60	87	73	→ False.

10 20 5 27 | 40 | 60 15

10 20 5 27 | 40 | 60 15

sum = 40 60.

74 87 80 → true

74 75 74 → x

75 75 75 → true

```
int low = max(arr,n),   high = sum(arr,n)
while (low <= high)
{
    mid = (low + high)/2
    if (valid( arr, N, k, mid ))
    {
        ans = mid;
        high = mid - 1;
    }
    else { low = mid + 1;
    }
}
return ans;
```

No ↓ log N
coz. it not sorted
log(high - low + 1)
log(sum - max + 1)

```
bool valid (arr, N, k, mid)
{
    sum = 0   c = 1
    for(int i=0; i < N; i++)
    {
        if (sum + arr[i] < mid)
        { sum += arr[i]
        }
        else{
            sum = arr[i];
            c += 1;
        }
    } if (k >= c)
    { return T;
    } else { return F;
    }
```

O(N)

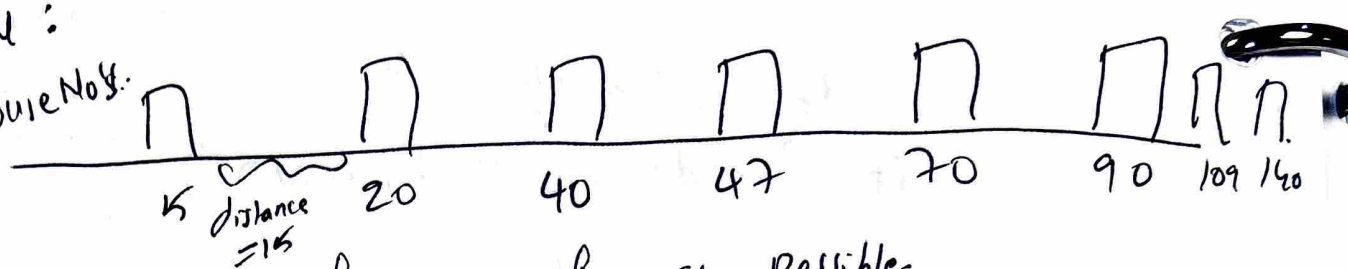
} N

Total Complexity
N * log(high - low + 1)
(or)
N * log(sum - max + 1)

PROBLEM organizing some party

K = 4 (4 families to assign houses to)

ARR:



we want to give houses as far as possible.

maximize the distance bhw the houses

min

↑ Calculating

Giving out: (minimum).

If these are distances bhw houses & if app ask at what dist they are

35, 30 20 → u say 20

So this is Maximizing the Minimum result

we are giving 2 houses.

$N-2C_{k-2}$ ↑ K, 1

Solutions

1) Backtracking → Ask in the B.O room

2) Binary Search

B.S

arr.sort()

low = min(arr[i] - arr[i-1]), high = arr[N-1]

↓ minimum the adjacent houses

```
while (low <= high)
    if (valid(arr, N, K, mid))
    {   ans = mid;
        low = mid+1;
    }
    else
    {   high = mid-1;
    }
}
```

$O(\log(h-l+1))$

$O(\log(\text{arrE}$

$(N * O(\log(M-m+1))$

max min.

```
void valid(arr, N, k, mid)
{
    diff = arr[0], C = 1
    for(int i = 0; i < N; i++)
    {
        if(abs(diff - arr[i]) ≥ mid)
            diff = arr[i];
            C += 1
    }
    if C ≥ k.
        return T
    else return F
```

O(N)