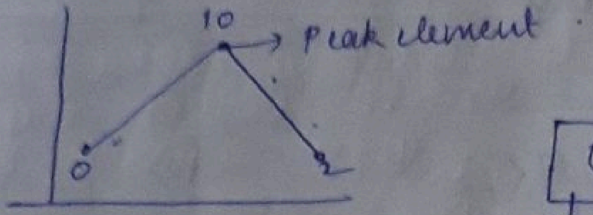
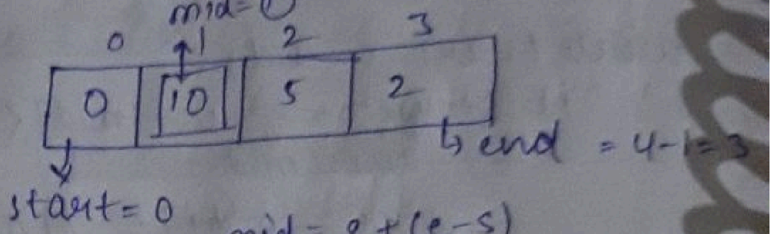
④ Peak element in a Mountain Array

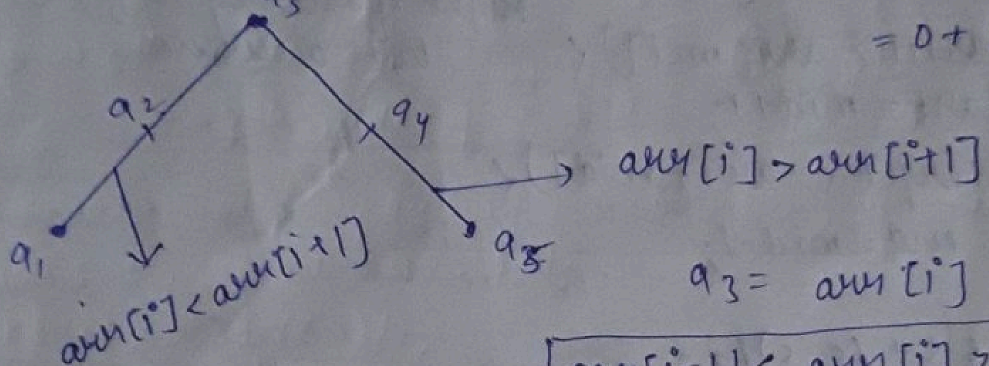
0	1	2	3
0	10	5	2

Peak element is 10





$$mid = s + \frac{(e-s)}{2}$$
$$= 0 + 1 = 1$$



a_3 = arr [i]

$\boxed{arr[i-1] < arr[i] > arr[i+1]}$

```
peakelement (vector <int> arr) {
    int start = 0; end = arr.size() - 1.
    int mid = start + (end - start)/2;
    while ( end < start) {
        if (arr[mid] < arr(mid+1]) {
            start = mid+1.
        }
        else if (arr[mid] > arr[mid+1]) {
            end = mid;
        }
        mid = start + (end - start)/2;
    }
    return ~~mid~~ start;
}
```

If start = mid+1
or
end = mid-1
~~so~~ we can use
while (start <= en

If start = mid or
end = mid
we will use
while (start < e

(5) Binary Search in Decending Order

```
same all just
(target> arr[mid]) {
    end = mid-1;
} (target < arr[mid]) {
      start = mid+1;
  }
```

→ The array must be sorted.

(6) square root of any no. using Binary

```
int sqrtfind (int num){
int s=0; e = num;
int mid = s+ (e-s) /2;
int ans = -1;
while (s<=e) {
  if (mid* mid == ~~ta~~num){
        return mid;
  else if (mid*mid <num) {
        ans = mid;
        start = mid+1;
  }
  mid = s+(e-s)/2;
 }
return ans;
}
```

```
int main() { int sqrt = sqrtfind(num
  // everything same just
   decimal part
double finalAns = sqrt
int decimal;
 cin >> decimal;
double step ~~sp~~ = 0.1;
for (int i=0; i<decimal;
                 i++);
    for(double j = finalAns;
            j*j<=num;
              j=j+step)
     finalAns = j;
  }
  step = step/10;
}
cout << finalAns;
```

(7) Binary search in 2D Array:

```
bool findbinary2d (vector<vector<int>> arr, int ~~a~~target) {
int rows = arr.size(); int cols = arr[0].size();
int s=0, e= rows* cols -1;
int mid = s+ (e-s)/2;
while (s<=e) {
  int rowIndex = mid/ cols;
  int colIndex = mid% cols;
  if( arr[rowIndex] [colIndex] == target) {
    cout << " Found at" << rowIndex << " and " << colIndex << endl
    return 0;
  else if( element < target) {        else if (element > target) {
    s= mid+1;          →                 e= mid-1;
  }                                  } m = s+(e-s)/2;
                                   return false;
```