|  |
| --- |
| Median (arr1[1…n], arr2[1…n])  N ← array length  Return median(arr1[1..n], 0, N-1, arr2, 0, N-1) |
| Median(arr1[1..n], l1, r1, arr2, l2, r2)  n← r1- l1 + 1  mid1 ← ⌈(r1+ l1)/2⌉  mid2 ← (r2+ l2)/2  m1← median(arr1[1..n], n)  m2← median(arr2[1..n], n)  if (n=1)  return (arr1[0] + arr2[0])/2  if(n=2)  return (Max(arr1[l1], arr2[l2]) + Min(arr1[r1], arr2[r2]))/2  if (m1=m2)  return m2  else if (m1 > m2)  return median (arr1[1..n], l1, mid1, arr2[1..n], mid2, r2)  else  return median (arr1[1..n], mid1 r1, arr2[1..n], l2, mid2) |
| Median (arr[1..n], int n)  {  if (n % 2 = 0)  return (arr[n / 2] + arr[n / 2 - 1]) / 2;  else  return arr[n / 2];  } |

**Basic operation**: number of comparing

**Setting and solving the recurrence relation:**

𝑇(𝑛) = 𝑇(𝑛/2) + 1 level 1

|  |  |
| --- | --- |
| = [𝑇((𝑛/2)/2) + 1] + 1  = 𝑇(𝑛/4) + 2 level 2  = [𝑇((𝑛/4)/2) + 1] + 2  = 𝑇(𝑛/8) + 3 level 3  :  = 𝑇 𝑛 2𝑘 + 𝑘  :  = 𝑇 1 + log2 𝑛 level log2n (leaves)  = Θ(log2𝑛) | 1 = 𝑛  2𝑘  𝑛 = 2𝑘  log2 𝑛 = 𝑘 |