(3) Given a sequence of n real numbers stored in an array, A[1], A[2], A[3], ..., A[n], we wish to find two numbers A[i] and A[j], where i < j, such that A[i] ≤ A[j] and their sum is the largest. Please design a divide-and-conquer algorithm to solve the above problem. Please analyze the complexity of your algorithm. The complexity of your algorithm must be $\mathcal{O}(n \cdot lgn)$ or better.

Pseudocode

Algorithm 1 Find max two such that:

$$i < j$$

and

$$A[i] \leq A[j]$$

```
1: function FindMaxSum(A, p, r)
2:     if p equals r then return −∞
3:     else
4:         mid ← ⌊(p + r)/2⌋
5:         i_1, j_1 ← FindMaxSum(A, p, mid)
6:         i_2, j_2 ← FindMaxSum(A, mid + 1, r)
7:         i_3, j_3 ← FindCrossingMax(A, p, mid, r)
8:         i_max, j_max ← {i, j | max_{1≤k≤3}(A[i_k] + A[j_k])}
9:         return i_max, j_max
10:    end if
11: end function
```

Algorithm 2 Perform work to actually find the max two for a given p and r.

```
1: function FindCrossingMax(A, p, mid, r)
2:     i ← −∞
3:     j ← −∞
4:
5:     for k from mid + 1 up to r do
6:         if A[j] < A[k] then
7:             j ← k
8:         end if
9:     end for
10:
11:    for k from mid down to p do
12:        if A[i] ≤ A[j] and A[i] < A[k] then
13:            i ← k
14:        end if
15:    end for
16:
17:    if i equals -∞ or j equals -∞ then
18:        return -∞
19:    else
20:        return i and j
21:    end if
22: end function
```