Question 5 - Local Minimum Algorithm

Jaan Tollander de Balsch

2018-09-30

Given an array $A[1...n]$, we say that entry $A[i]$ is *locally minimum* if

$$A[i] = \min(A[i-1], A[i], A[i+1]);$$

for $i = 1$, it is locally minimum if $A[i] < A[i+1]$ and for $i = n$, when $A[i] < A[i-1]$.

Describe an algorithm that finds a local minimum in an array A in time $O(\log n)$. Explain your algorithm either in clear English or a clear pseudocode.

The local mimimum algorithm (1)

```
function local_minimum_search(arr, low, high, n)
    mid = low + div(high - low, 2)
    if (mid == 1) || (mid == n) ||
        (arr[mid] <= arr[mid-1] && arr[mid] <= arr[mid+1])
        return mid
    elseif arr[mid-1] < arr[mid]
        return local_minimum_search(arr, low, mid-1, n)
    else
        return local_minimum_search(arr, mid+1, high, n)
    end
end


# Returns the index of the local minimum
function local_minimum(arr)
  n = length(arr)
  if n == 1
      return 1
  elseif n == 2
      if arr[1] <= arr[2]
          return 1
      else
          return 2
```

```
    end
  else # n >= 3
    return local_minimum_search(arr, 1, n, n)
  end
end
```

Input: An array $A = \langle a_1, ..., a_n \rangle$.

Output: An index i such that element a_i meets the requirement of being *locally minimum*. Being locally minimum is defined as

$$\begin{cases} a_i < a_{i+1} & i = 1 \\ a_i < a_{i-1} & i = n \\ a_i = \min(a_{i-1}, a_i, a_{i+1}) & \text{otherwise} \end{cases}$$

It should be noted that $a_i = \min(a_{i-1}, a_i, a_{i+1})$ is equivalent to $(a_i \leq a_{i-1}) \wedge (a_i \leq a_{i+1})$.

Local Minimum Algorithm:

1) In the trivial case $n = 1$ local minimum $i = 1$.
2) In the trivial case $n = 2$, if $a_1 \leq a_2$ then $i = 1$, otherwise $i = 2$.
3) Otherwise when $n \geq 3$, the algorithm uses *divide and conquer* strategy. It works similarly to a binary search. First it finds the middle index *mid* from the current slice, from index *low* to *high*, of the array A. The initial slice is

$$\begin{aligned} low &:= 1 \\ high &:= n \end{aligned}.$$

   The middle index is calculated

$$mid := low + \lfloor (high - low)/2 \rfloor$$

   a) *Base Case*: If any of the following conditions is true a local mimimum will be at index $i = mid$.
      1) If $mid = 1$ the search has moved the first element and due to the *recursive case 1* the element must be a local mimimum $(a_1 < a_2)$.
      2) If $mid = n$ the search has moved to the last element and due to the *recursive case 2* the element must be a local minimum $(a_n < a_{n-1})$.
      3) If $(a_{mid} \leq a_{mid-1}) \wedge (a_{mid} \leq a_{mid+1})$ then the element is local mimimum by definition.

   b) *Recursive Case*: If the base case is not local minimum then we'll divide the search space and evaluate the recursive case with the new parameters.

1) If $a_{mid-1} < a_{mid}$ then set new parater values and start again from calculating the middle index:

$$\begin{aligned} low &:= low \\ high &:= mid - 1 \end{aligned}.$$

2) If $a_{mid+1} < a_{mid}$ then set new parater values and start again from calculating the middle index:

$$\begin{aligned} low &:= mid + 1 \\ high &:= high \end{aligned}.$$

References

1. Agarwal R. Find a local minima in an array - geeksforgeeks. Available at: https://www.geeksforgeeks.org/find-local-minima-array/ [Accessed October 1, 2018]