MTH-681 Analysis and Design of Algorithms
Assignment (2): Divide and Conquer

Mostafa Hassanein

15 March 2025

2-5

Suppose you have a row of $n > 2$ grayscale pixels. An array $I[1, \ldots, n]$ contains the intensities of the respective pixels. The intensity of a pixel is an integer between 0 (black) and 255 (white). Suppose that the first and last pixels are black. We say that a pixel, other than the first and the last, stands out if its intensity is greater than or equal to the intensities of its two neighbors.

a. Describe a $O(\log(n))$-time algorithm to find an index $2 \leq j \leq n-1$ such that the j^{th} pixel stands out. (Note that you need to argue that your algorithm runs in $O(log(n))$.)

b. Prove that your algorithm is correct.

Solution:

a.

```java
int search(int[] intensities) {
  return searchHelper(intensities, 1, intensities.length);
}

int searchHelper(int[] intensities, int low, int high) {
  int mid = (low + high) / 2;

  // Base case.
  if (isStandOutPixel(intensities, mid)) {
    return mid;
  }

  // Recursive step.
  int forwardDifferenceAtMid = intensities[mid + 1] - intensities[mid];
  if (forwardDifferenceAtMid > 0) {
    return searchHelper(intensities, mid, high);
  }
  else {
    return searchHelper(intensities, low, mid);
  }
}

boolean isStandOutPixel(int[] intensities, int index) {
  return intensities[index] >= intensities[index - 1]
      && intensities[index] >= intensities[index + 1];
}
```

Theorem 1. $T(n) = \Theta(\log(n))$

Proof. (By Substitution)

Assuming n is a power of 2 to simplify the analysis, then:

$$\begin{aligned} T(n) &= T(n/2) + \Theta(1) \\ &= T(n/2) + 2\Theta(1) \\ &= T(n/4) + 3\Theta(1) \\ &\vdots \\ &= T(n/2^k) + k\Theta(1) \\ &= T(1) + \lg(n) * \Theta(1) \\ &= \Theta(1) + \lg(n) * \Theta(1) \\ &= \Theta(\lg(n)). \end{aligned}$$

□

b.

Before we prove correctness, we first state a useful invariant.

Lemma 2. *The forward difference at the lower index (forwardDifferenceAtLow) and the backward difference at the higher index (backwardDifferenceAtHigh) are always non-negative.*

Proof.

TO BE DONE □

Theorem 3. *The algorithm is correct.*

Proof. (By Strong Induction)

Base case ($n = 3$):

Since both pixels at i=1 and i=3 have an intensity value of zero and the pixel between them (midPixel) has an intensity value between 0..255, then the intensity of midPixel must be greater than or equal to the pixels at i=1 and i=3 and therefore midPixel is a stand out pixel.

The algorithm correctly returns the index of midPixel.

Inductive step ($n > 3$):

Suppose that midPixel is not a stand out pixel and forwardDifferenceAtMid > 0.

Since forwardDifferenceAtMid > 0 and backwardDifferenceAtHigh > 0 (by Lemma 2), then we must have a local maxima in the interval between *mid..high*.

Since a standout pixel is a local maxima, then we have a standout pixel in the interval between *mid..high*.

Since the interval size of *mid..high* is smaller than *low..high*, then by the induction hypothesis the algorithm will return the correct result.

On the other hand, if midPixel is not a standout pixel and forwardDifferenceAtMid $\not> 0$, then it must be the case that the intensity at index $mid - 1$ is higher than that at *mid*. Therefore, backwardDifferenceAtMid > 0.

Since the forwardDifferenceAtLow > 0 (by Lemma 2), then we have a local maxima in the interval between *low..mid*, which is a standout pixel.

Since the interval size of low..mid is smaller than low..high, then by the induction hypothesis the algorithm will return the correct result.

□