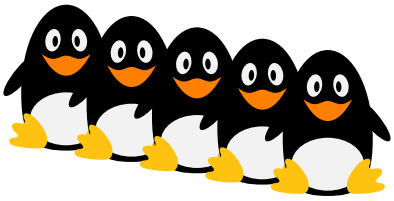
CS 130B Section 4

A divide and conquer correctness proof

Problem 3 (Penguins). You come across n penguins standing in a line. Each penguin has a distinct height. A local minimum is a penguin that is shorter than both its neighbors (or its one neighbor for the first and last penguin).

Algorithm. Let $A = [a_1, \ldots, a_n]$ be the input array. We will solve this problem recursively. For the base case, for an array of length one or two, check one or both elements, and return any local minimum. For an array of length $n \geq 3$, check whether the penguin at index $a_{\lceil n/2 \rceil}$ is local minimum, and if so, return it. If not, if penguin $a_{\lceil n/2 \rceil - 1}$ is shorter than penguin $a_{\lceil n/2 \rceil}$, recurse on the array $L = [a_1, \ldots, a_{\lceil n/2 \rceil}]$, and return its local minimum. Else recurse on the other half, $R = [a_{\lceil n/2 \rceil + 1}, \ldots, a_n]$.



(a) Local minimum exists.

Let's prove that a local minimum exists. We claim that if none of the first $n-1$ penguins are local minima, then the last penguin must be a local minimum.

Suppose that none of the first $n-1$ penguins are local minima. This means the first penguin must be taller than the second penguin. The second penguin is also not a local minimum, and yet the penguin just before it is taller, so the third penguin must be shorter than the second penguin. So far, we have seen that the first three penguins are in descending order. Continuing in this way (by induction), we can see that all of the penguins are in descending order. In particular, the penultimate ($(n-1)$th) penguin is taller than the last penguin, so the last penguin is a local minimum, as desired.

(b) Proof of correctness.

We proceed by induction on n, the number of penguins in the array. This will prove that the algorithm is correct on inputs of length n, for all n.

Base case: $n = 1$ and $n = 2$.

In this case, we manually check one or both elements and return whichever is a correct local minimum.

Induction step:

Suppose that for some $n \geq 3$, our algorithm returns a correct local minimum for all arrays of length $k < n$. We want to prove that the output is correct for arrays of length n.

Assume we have an array A of length n. If the penguin at index $a_{\lceil n/2 \rceil}$ is local minimum, then we certainly return a correct local minimum. If not, then we recurse on either the left or right half. By the induction hypothesis, since the length of each half is less than n, we know that our algorithm can find a correct local minimum for either half.

To recurse, we check if penguin $a_{\lceil n/2 \rceil - 1}$ is shorter than penguin $a_{\lceil n/2 \rceil}$, and if so, return the local minimum for the left half. We have chosen to split at $\lceil n/2 \rceil$ rather than $\lfloor n/2 \rfloor$ so that when $n \geq 3$, the left half will have at least two elements. This ensures that we don't get an out-of-bounds error when we check the penguins at indices $a_{\lceil n/2 \rceil - 1}$ and $a_{\lceil n/2 \rceil}$.

We need to check that the local minimum returned from the left half (call it m_L) is also a local minimum for all of A. The local minimum from the left half cannot be penguin number $a_{\lceil n/2 \rceil}$ (since penguin $a_{\lceil n/2 \rceil - 1}$ is shorter than penguin $a_{\lceil n/2 \rceil}$). Therefore, putting L and R together to form A doesn't give penguin m_L any new neighbors, so m_L is also a local minimum for all of A.

We reach the third case in the algorithm if the middle penguin is not a local minimum and penguin $a_{\lceil n/2 \rceil - 1}$ is taller than penguin $a_{\lceil n/2 \rceil}$. In this case, we need to check that the local minimum returned from the right half is also a local minimum for all of A.

We can only reach this case if penguin number $\lceil n/2 \rceil$ is not a local minimum and penguin $\lceil n/2 \rceil - 1$ is taller than penguin $\lceil n/2 \rceil$, so this implies that penguin $\lceil n/2 \rceil + 1$ is shorter than penguin $\lceil n/2 \rceil$. Therefore, even if the local minimum returned from the right half is at index $\lceil n/2 \rceil + 1$, it is still a valid local minimum for all of A. This shows that in all cases, our algorithm outputs a correct local minimum for the array A.

Therefore, by induction, our algorithm is correct for all n.

Fun fact:

We have found another way of proving that a local minimum exists. We have an algorithm that always outputs a local minimum, so this means it must exist!