6.006 Pset 1 Question 5

Proof of algorithm 4 as a correct algorithm.

An algorithm is correct if given a non-empty 2D matrix as input, it always returns a peak from the matrix, which always exists because given a finite set of numbers, there always exists a maximum of the set, for which the definition of a peak will be satisfied regardless of the arrangement of the values inside the matrix.

Claim 1: alg4 will always return a location given a non-empty 2D input matrix of dimensions mxn.

In each iteration of alg4, either None is returned if the input problem is empty, a location is returned, or alg4 will divide the map, choose a subproblem, and call alg4 on the subproblem. Since the division of the subproblem is done by dividing the current problem in half through a floor division on either the number of rows in the current problem or the number of columns, therefore the subproblem passed to a next iteration will never have a negative number of rows or columns. Therefore, the only way for alg4 to not return a location is if the input is empty.

Assume that in some cases, alg4 will not return a location, then by the above, it must eventually pass an empty subproblem into the next and last iteration, which returns None.

Define “the dividing dimension” to be the dimension, either row or column, which the current iteration divides in two in order to create two subproblems.

Such a situation can only occur if the dividing dimension of the second last iteration has 1 or 2 entries, as otherwise both subproblems created will have at least 1 entry in the dividing dimension. If there is only 1 entry, then alg4 will find the best divider location (BL) on the problem, and return it, because it either updates the BS to BL, or returns BL as BS since BS is already contained in the problem (see subclaim). If there are two entries, then,

Case 1: BS will stay on or be updated to BL. This means that this BS will be returned.

Case 2: BS will stay on or be updated to the only neighbour which gets compared in the getBetterNeighbour function, meaning alg4 will choose the non-empty subproblem to go into the next iteration. This reduces the problem to 1 entry in the dividing dimension, and if alg4 eventually reaches another iteration with this dimension as the dividing dimension, then alg4 will return a location as a 1 entry problem.

Therefore, alg4 will never reach an empty input matrix. Contradiction!

Thus, alg4 will always return a location given a non-empty input matrix.

Subclaim: The problem of the current iteration of alg4 will always contain the Best Seen (BS) location.

The first iteration of alg4 will assign BS to the returned location of getBetterNeighbour of the best location in the divider, which must be in the current problem or else getBetterNeighbour will return the best location in the divider, which is already in the problem.

In any iteration after the first iteration, if the subproblem received includes BS and alg4 recurses, then either BS updates to a neighbour of the divider’s best location, or the BS stays in the same position. Either way, the subproblem chosen to recurse into alg4 will contain BS since one of the subproblems must contain BS, and alg4 always selects the subproblem which contains BS. Thus, all iterations of alg4 will contain the Best Seen location.

Claim 2: If alg4 returns a value, then it is a peak of the matrix.

Assume that if alg4 returns a value, then sometimes it is not the peak of the matrix. Then for this case, the return location (d1, e1), will have a neighbour (d2, e1) where (d2, e1) > (d1, e1). d and e are arbitrary dimensions. As the algorithm returns, it will reach a subproblem i where it is no longer a peak. For this to happen, one side of the RL (return location) must be in the divider of subproblem i which exposes a location that satisfies the inequality for (d2, e1) as above. Name the first location which exposes the ascend from RL (d2, e1).

There are two cases for the subproblem when it was first encountered:

Case 1: BS was updated to a value neighbouring to subproblem i’s BL.

Subproblem i chooses to iterate on the subproblem containing (d1, e1), so there exists a j for which (d1, ej) > (d2, ej), where (d2, ej) >= all locations in subproblem i’s divider, and so (d1, ej) >= all locations in subproblem i’s divider. Since (d1, e1) < (d2, e1), j is not 1, and the new BS is not the RL. Since (d1, ej) > (d2, ej), (d2, ej) >= (d2, e1), and (d2, e1) > (d1, e1), by transitivity (d1, ej) > (d1, d1). Since BS will only update when a candidate location is greater than itself, (d1, e1) will never be BS. But RL will always be BS when it is returned. Contradiction!

Case 2: BS was not updated.

Subclaim: (d1, e1) is not the BS.

alg4 compares the max of the divider of subproblem i to BS. Since BS is not updated, all locations in the divider is <= BS. Since (d2, e1) > (d1, e1), (d1, e1) must not be the BS.

alg4 will only return the BS in its last iteration. Since (d1, e1) is not BS, at some point in a future iteration, BS must update to (d1, e1); therefore, the current BS must be < (d1, e1). However, (d2, e1), a value in the current divider, is greater than (d1, e1), and since BS is not updated, the max of subproblem i’s divider, (d2, ej) <= BS. Since (d1, e1) < (d2, e1) <= (d2, ej) <= BS, (d1, e1) < BS. So RL will never update to BS and will never be the return location. Contradiction!

No matter the case, we reach a contradiction if RL is exposed as a non-peak as alg4 returns through its subproblems.

Therefore, if a location is returned, it will be a peak.

Since a location will always be returned if alg4 receives a non-empty problem, alg4 will always return a peak to a non-empty problem.

Hence, alg4 is a correct algorithm.