Justin Mendiguarin

Professor Ouyang

CSC 130

9/30/15

Assignment 1 - Maximum Subsequence Product

**Design:**

Divide and Conquer:

* Split the problem into two equal sub problems
* Solve the sub problems recursively
* Combine together the two solutions of the sub problems, possibly doing a small amount of additional work, to arrive at a solution for the whole problem

We must find the subsequence whose product is the largest. When finding the left center maximum sequence and right center maximum sequence, the subsequence must at least contain their center point (center and center + 1) respectively.

Solving for the maximum center product containing the two center points:

* I began by creating two separate (For) loops, one for solving the max left center product and one for the max right center product. To find the maximum center product containing the two center points, I multiplied the max left center product and max right center product. Thus, I divided the problem into two, solved the sub-problems separately, and then combined the two solutions of those two sub problems to arrive at a solution for the max center product.

**//Left Max Center Product:**

**for** (**int** i = center - 1; i >= start; i--) {

leftProductofCrossinMiddle \*= ARRAY[i];

**if** (leftProductofCrossinMiddle > maxLeftCenterProduct) {

maxLeftCenterProduct = leftProductofCrossinMiddle;

}

}

//Right Max Center Product:

**for** (**int** i = center + 2; i <= end; i++) {

rightProductofCrossinMiddle \*= ARRAY[i];

**if** (rightProductofCrossinMiddle > maxRightCenterProduct) {

maxRightCenterProduct = rightProductofCrossinMiddle;

}

}

// Max Center Product:

// Combine the solution from the left and right max center product to arrive at a

// Solution for the max center product.

**int** maxCenterProduct = maxLeftCenterProduct \* maxRightCenterProduct;

* For example, given a list of integers: [4, -3, 5, -2, 1, 2, 6, -2].

|  |  |
| --- | --- |
| 4, -3, 5, -2 | 1, 2, 6, -2 |

* We divide the array in middle such that
* We now split the problem into two, we solve for the left maximum center product containing the center point [-2]. We do the same for the right maximum center product containing the center point [1].

Solving for the minimum center product containing the two center points:

* We take into account for the minimum center product, when the left and right center product are both negative, there is possibility that it could give off the maximum center product value, that is when multiplied could result a positive value.

**//Left Max Center Product & Left Min Center Product.**

**for** (**int** i = center - 1; i >= start; i--) {

leftProductofCrossinMiddle \*= ARRAY[i];

**if** (leftProductofCrossinMiddle > maxLeftCenterProduct) {

maxLeftCenterProduct = leftProductofCrossinMiddle;

}

**else if** (leftProductofCrossinMiddle < minLeftCenterProduct) {

minLeftCenterProduct = leftProductofCrossinMiddle;

}

}

// Right Max Center Product & Right Min Center Product.

**for** (**int** i = center + 2; i <= end; i++) {

rightProductofCrossinMiddle \*= ARRAY[i];

**if** (rightProductofCrossinMiddle > maxRightCenterProduct) {

maxRightCenterProduct = rightProductofCrossinMiddle;

}

**else** **if** (rightProductofCrossinMiddle < minRightCenterProduct){

minRightCenterProduct = rightProductofCrossinMiddle;

}

}

// Max Center Product & Min Center Product:

// Combine the solution from the left and right max center product to arrive at a

// Solution for the max center product.

**int** maxCenterProduct = maxLeftCenterProduct \* maxRightCenterProduct;

//Combine the solution from left and right minimum center product to solve for minimum center product.

**int** minCenterProduct = minLeftCenterProduct \* minRightCenterProduct;