Tecnológico
de Monterrey

Tarea 1 Divide y Venceras

Análisis y diseño de algoritmos avanzados

Gpo 607

David Mireles Gutierrez **A00836010**

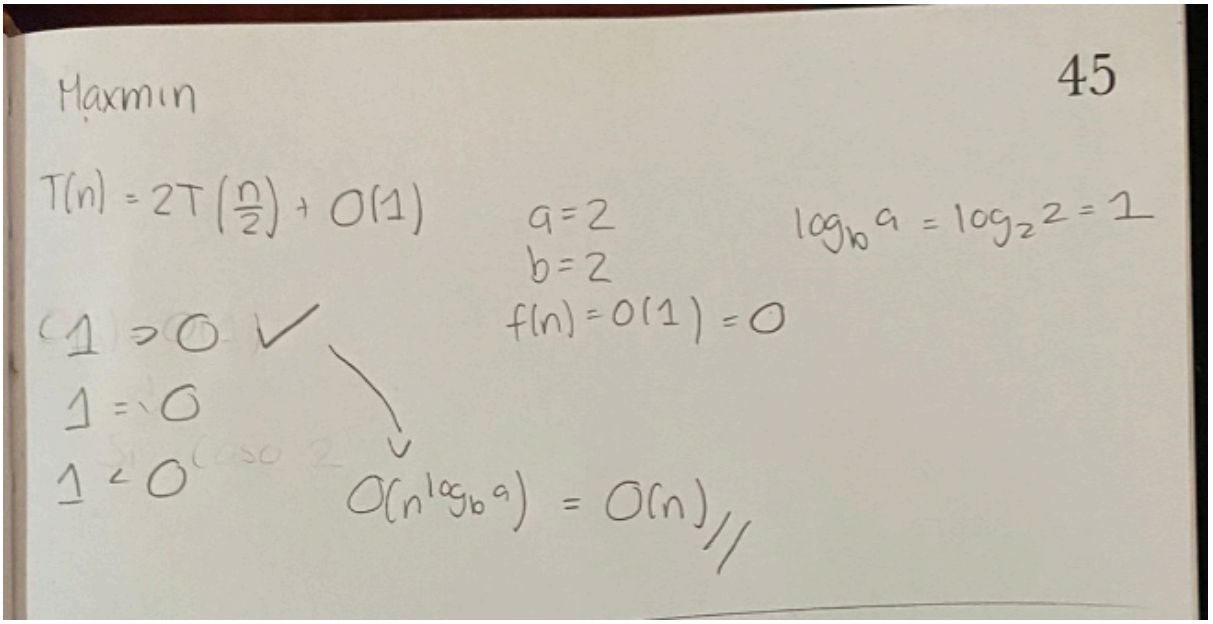
Complejidad Algoritmo Maximin Teorema Maestro

C/C++

```
Tuple maximin(int *A, int low, int high){
    if (low == high) {
        return Tuple(A[low], A[low]);
    }

    if (high == low + 1) {
        if (A[low] < A[high]) {
            return Tuple(A[low], A[high]);
        } else {
            return Tuple(A[high], A[low]);
        }
    }
    int mid = (low + high) / 2;
    Tuple left = maximin(A, low, mid);
    Tuple right = maximin(A, mid + 1, high);
    int min_val = min(left.get_min(), right.get_min());
    int max_val = max(left.get_max(), right.get_max());

    return Tuple(min_val, max_val);
}
```



Complejidad Algoritmo Maximum Subarray Teorema Maestro

C/C++

```
Result find_max_crossing_subarray(int *A, int low, int mid, int high) {
    int leftSum = numeric_limits<int>::min();
    int sum = 0;
    int maxLeft = mid;

    for (int i = mid; i >= low; i--) {
        sum += A[i];
        if (sum > leftSum) {
            leftSum = sum;
            maxLeft = i;
        }
    }
    int rightSum = numeric_limits<int>::min();
    sum = 0;
    int maxRight = mid + 1;

    for (int i = mid + 1; i <= high; i++) {
        sum += A[i];
        if (sum > rightSum) {
            rightSum = sum;
            maxRight = i;
        }
    }
    return Result(maxLeft, maxRight, leftSum + rightSum);
}

Result find_maximum_subarray(int *A, int low, int high) {
    if (low == high) {
        return Result(low, high, A[low]);
    } else {
        int mid = (low + high) / 2;
        Result leftSide = find_maximum_subarray(A, low, mid);
        Result rightSide = find_maximum_subarray(A, mid + 1, high);
        Result crossed = find_max_crossing_subarray(A, low, mid, high);

        if (leftSide.get_sum() >= rightSide.get_sum() && leftSide.get_sum() >=
crossed.get_sum()) {
            return leftSide;
        } else if (rightSide.get_sum() >= leftSide.get_sum() &&
rightSide.get_sum() >= crossed.get_sum()) {
```

```
            return rightSide;
        } else {
            return crossed;
        }
    }
}
```