**ALGORITHMS & DATA STRUCTURES**

**Lahcen Ouarbya**

**TOPIC 2: RECURSION**

**LAB. SUBMISSION**

In this activity, you will work **individually** analysing recursive proposals. Remember that you can discuss ideas with your classmates, but you cannot see other’s work neither others can see your individual work.

After finishing the activities presented in this document, attempt the LAB SUBMISSION QUIZ. The grade you obtain in that quiz will be your grade for this lab submission. The quiz will open on after the lecture on week2 of topic 2 to give you time to work and reflect on your work before trying to answer the quiz questions.

Enjoy recursion! Enjoy recursion! Enjoy recursion!... (sorry, no base case)

|  |
| --- |
| **DISCOVER, ANALYSE & IMPLEMENT** |

In the following, you will be given different recursive algorithms. For each:

1. Describe what is the task performed by the algorithm
2. Find the recurrence equation that describes the running time of the algorithm
3. Use the recurrence equation to find the time complexity of the algorithm, either by expanding the recurrence equation or using the Master Theorem
4. Implement it (either in Java or C++)

|  |  |
| --- | --- |
| **ALGORITHM 1**  A: 1D array of N integer numbers  B: 1D array of N integer numbers  N: number of elements of arrays A and B  function ALG1(A,B,N):  if(N==0):  return 0  return A[N-1]\*B[N-1]+ALG1(A,B,N-1) | |
| Task | Multiples items in 2 arrays together |
| Recurrence equation | C + (N-1) |
| Time Complexity | Θ(N) |
| Return value for :  A: 1D array made of the first X even numbers (starting at number 2)  B: 1D array made of the first X odd numbers (starting at 1) | Return value for X=100: 1343300  Return value for X=200: 10706600  Return value for X=300: 36089900 |

|  |  |
| --- | --- |
| **ALGORITHM 2**  A: 1D array  low: lowest index  high: highest index  function ALG2(A,low,high):  if(high-low==1):  if(A[low]<A[high])  return high  else  return low  if (low==high)  return high  mid=low+floor((high-low)/2)  a=ALG2(A,low,mid)  b=ALG2(A,mid+1,high)  if(A[a]>A[b])  return a  return b | |
| Task | Returns the highest value |
| Recurrence equation (in terms of N, number of elements of A) | (n/2) (n/2) + C |
| Time Complexity | Θ(NlogN) |
| Return value for  A: 1D array made of the first X even numbers (starting at number 2) | Return value for X=100: 99  Return value for X=200: 199  Return value for X=300: 299 |
| Return value for  A: 1D array made of the first X odd numbers (starting at 1) | Return value for X=100: 99  Return value for X=200: 199  Return value for X=300: 299 |

Hint for ALG2: When finding the recurrence equation, please notice that the constants multiplying the running time of smaller size instances **ARE** important. For example, the time complexity of T(N)=T(N-1)+C is **VERY** different from that of T(N)=2T(N-1)+C.

|  |  |
| --- | --- |
| **ALGORITHM 3**  A: square matrix (2D array of N rows and N columns)  N: Number of rows/Number of columns  function ALG3(A,N)  if(N==0)  return 0  return A[N-1,N-1]+ALG3(A,N-1) | |
| Task |  |
| Recurrence equation | C + (N-1) |
| Time Complexity | Θ(N) |
| Return value for  A: 2D array where the value stored in element [i,j] is equal to i+j | Return value for N=50:  Return value for N=100:  Return value for N=X=150: |

|  |  |
| --- | --- |
| **ALGORITHM 4**  A: square matrix (2D array of N rows and N columns)  B: square matrix (2D array of N rows and N columns)  N: Number of rows (and number of columns) of A and B  def ALG4(A,B,N)  if(N==0)  return 0  return A[N-1][N-1]+B[N-1][N-1]+ALG4(A,B,N-1)+ALG4(A,B,N-1) | |
| Task (an equation here it is easy) | *In this case, instead of explaining in words what this recursive algorithm does, please write the equation of what is calculated for 3x3 matrices.* |
| Recurrence equation | C1 + C2 + (N-1) + (N-1) |
| Time Complexity | Θ(2^n) |
| Return value for  A=[0 1 2  1 2 3  3 4 5]  B=[1 1 1  2 2 2  3 3 3] |  |
| Return value for  A: 2D array where the value stored in element [i,j] is equal to i+j  B: 2D array where the value stored in element [I,j] is equal to i\*j | Return value for N=10:  Return value for N=15:  Return value for N=20:  Return value for N=30: |