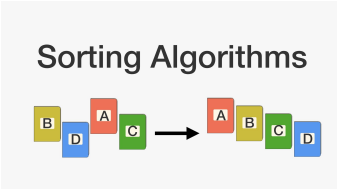
Final Exam Part 2 – A Hand-on Exam to implement different algorithmic requirements.
50 points
Due (via your git repo) no later than 12:00 p.m., Monday, 17th May 2021.

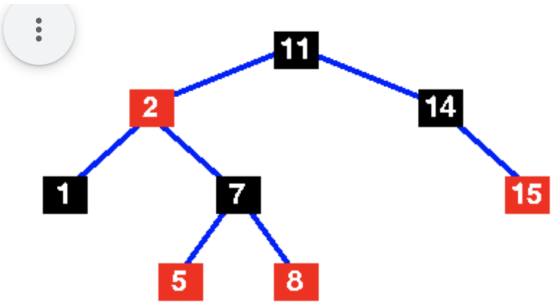
Part A (25 points):



1. Let us suppose that the patient details such as patient id's are given and the requirement is to sort the patients in ascending order based on their id's.
2. You can either implement **Quick** or **Merge** Sort in Python or Java, to complete this part of the exam.
   (a) A starter code in the files named `qs.py`, `ms.py`, `QS.java` and `MS.java` is provided in the repository. The starter code contains the implementation to generate the input dataset using random sampling. It is **not complete** and as it stands doesn't sort the input dataset.
   (b) If you had chosen to implement Quick Sort, complete the implementation of `main_sort` and `partition` methods. If you had chosen to implement Merge Sort, complete the implementation of `main_sort` and `merge` methods. Once these methods are completed, the starter code should be able to display the original unsorted data and the output sorted data on the console.
   (c) The algorithm for Quick and Merge sort is provided for your reference in the last section.

PS next page.

Part B (25 points):



1. Analyze the Red-Black tree provided above. Insert a new node 9 into the tree. Draw the tree and include all the steps while developing such a tree in a file named RB-1. The file should include a clear image of your drawing and should be uploaded through the GitHub exam repository. Clearly indicate the colors of a node using a Red and Black marker or using a written notation of R or B next to the nodes.
2. Delete the node 2 from the output tree in the previous step. Draw the tree and include all the steps while developing such a tree in a file named RB-2. The file should include a clear image of your drawing and should be uploaded through the GitHub exam repository. Clearly indicate the colors of a node using a Red and Black marker or using a written notation of R or B next to the nodes.
3. How is a red black tree different from a Binary Search Tree and 2-3 Trees? Provide your answer in the file named RB-Analysis.

Appendix

1. The algorithms for QuickSort and partition procedure are provided below:

Algorithm - Partition(A, p, r)

Input: an n-element un-sorted array A of integer values, a lower bound p of the array A, and a pivot r in the array A.

Output: new position of the pivot.

$x \leftarrow A[r]$
$i \leftarrow p - 1$
for j = p to r-1 **do**
  if A[j] $\leq$ x **then**
    $i \leftarrow i + 1$
    swap A[i] and A[j]
  end if
end for
swap A[i+1] and A[r]
return i+1

Algorithm - QuickSort(A, p, r)

Input: an n-element un-sorted array A of integer values, a lower bound p of the array A, and a pivot r in the array A.

Output: an n-element sorted array A of integer values.

if p $<$ r **then**
  $q \leftarrow$ Partition(A,p,r)
  QuickSort(A, p, q-1)
  QuickSort(A, q+1, r)
end if

2. The algorithms for MergeSort and merge procedure are provided below:

Algorithm - Merge(A, p, m, r)

Input: an n-element un-sorted array A of integer values, a lower bound p of the array A, a midpoint m, and a pivot r in the array A.

Output: an n-element merged array A of integer values.

$n_1 \leftarrow m - p$
$n_2 \leftarrow r - m$
Initialize Two Arrays L of size $n_1 + 1$ and R of size $n_2 + 1$
for i = 0 to n_1 **do**
  L[i] $\leftarrow$ A[p+i]
end for
for j = 0 to n_2 **do**
  R[j] $\leftarrow$ A[m+j]
end for
$L[n_1 + 1] \leftarrow \infty$ and $R[n_2 + 1] \leftarrow \infty$
Initialize $i, j \leftarrow 0$
for k = p to r **do**
  if L[i] $\leq$ R[j] **then**
    A[k] $\leftarrow$ L[i]
    i $\leftarrow$ i+1
  else
    A[k] $\leftarrow$ R[j]
    j $\leftarrow$ j+1
  end if
end for

Algorithm - MergeSort(A, p, r)

Input: an n-element un-sorted array A of integer values, a lower bound p of the array A, and a pivot r in the array A.

Output: an n-element sorted array A of integer values.

if p $<$ r **then**
  $m \leftarrow$ Floor $(p + r)/2$
  MergeSort(A, p, m)
  MergeSort(A, m+1, r)
  Merge(A, p, m, r)
end if

Submission Details

For this part of the exam, please submit the following to your GitHub repository by using the link shared to you by the Professor:

1. Modified source code of one of the following: `qs.py`, `QS.java`, `ms.py`, and `MS.java`.
2. Red Black Tree image files using the files named `RB-1` and `RB-2`.
3. Red Black Tree Analysis using the file named `RB-Analysis`.
4. A document to reflect your experience in this exam in a file named `reflections.txt`.
5. A document with the honor code pledge signed in a file named `honor-code.txt` document.
6. It is highly important, for you to meet the honor code standards provided by the college and to ensure that the submission is completed before the deadline. The honor code policy can be accessed through the course syllabus.