Assignment 6: Data Structures

CS 69001: Computing Lab 1

Introduction:

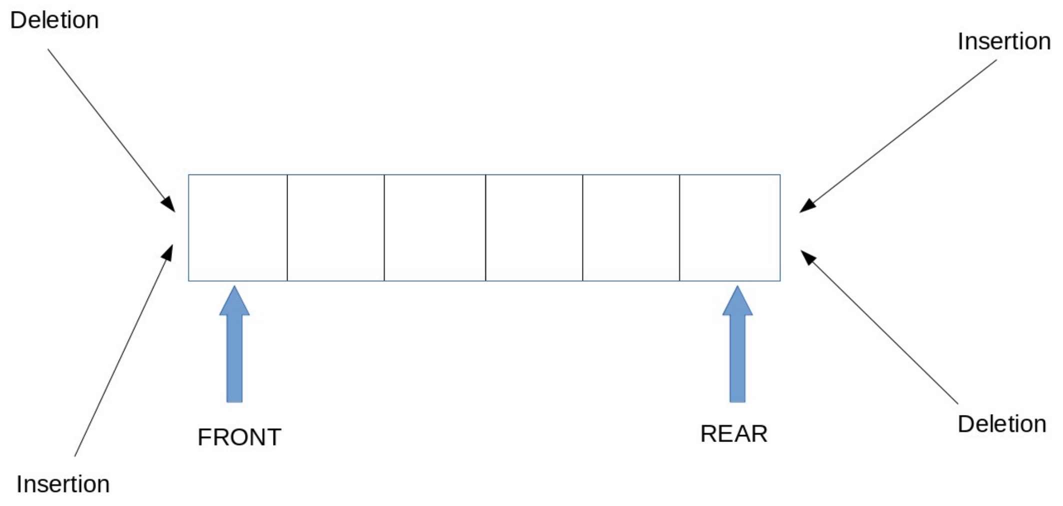
In this assignment you have to implement the external quick sort algorithm using double-ended priority queue (DEPQ).

Objective:

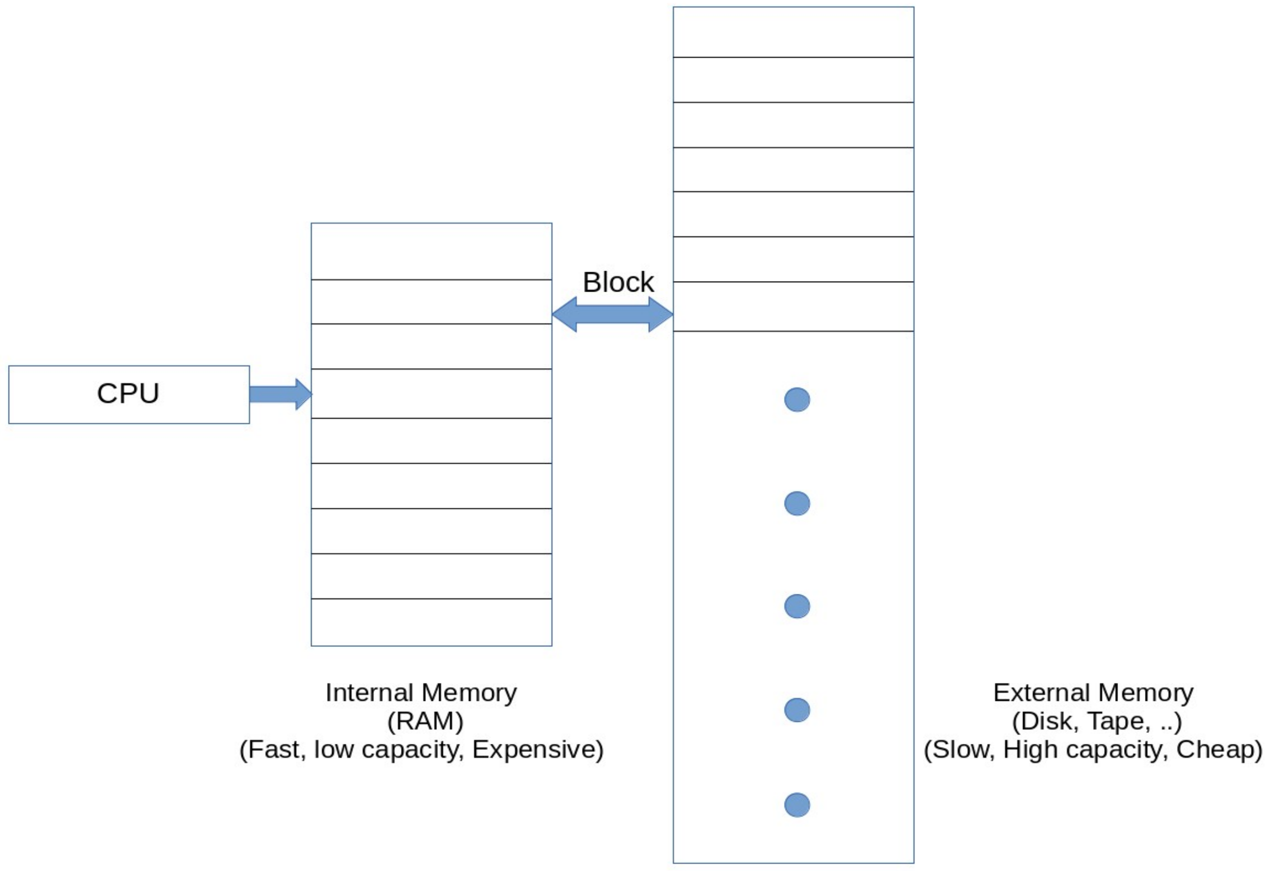
The aim of this assignment is to get acquainted with the DEPQ data structure and external sorting.

Problem Description:

double-ended priority queue
A DEPQ is a data structure similar to a priority queue or heap, but allows for efficient removal of both the maximum and minimum, according to some ordering on the keys (items) stored in the structure. Every element in a DEPQ has a priority or value. In a DEPQ, it is possible to remove the elements in both ascending as well as descending order. It is efficiently implemented using double-ended heaps.



External Sorting
External sorting is a term for a class of sorting algorithms that can handle massive amounts of data. External sorting is required when the data being sorted do not fit into the main memory of a computing device (usually RAM) and instead they must reside in the slower external memory (usually a hard drive). External sorting typically uses a hybrid sort-merge strategy. In the sorting phase, chunks of data small enough to fit in main memory are read, sorted, and written out to a temporary file. In the merge phase, the sorted subfiles are combined into a single larger file.



The quick sort technique can also be applied for sorting large data which cannot be stored together in the main memory. The external quick sort is efficiently implemented using the DEPQ (a benchmark algorithm) as follows:

1. The set of elements can be divided into three groups – a left group, a middle group and a right group.

2. Read as many elements as will fit into an internal DEPQ. The elements in the DEPQ will eventually be the middle group (pivot) of elements.

3. Read in the remaining elements. If the next element is less than or equals to the smallest element in the DEPQ, output this next element as a part of the left group. If the next element is greater than or equals to the largest element in the DEPQ, output this next element as part of the right group. Otherwise, remove either the max or min element from the DEPQ (the choice may be made randomly or alternately); if the max element is removed, output it as part of the right group; otherwise, output the removed element as part of the left group; insert the newly input element into the DEPQ.

4. Output the elements in the DEPQ, in sorted order, as the middle group.

5. Sort the left and right groups recursively.

Problem Statement

Task 1 (40 marks, +10 if the implementation is optimal)
You have to implement a DEPQ using the *list* data structure in Python. Your implementation should supports the following operations:

1. isEmpty(): Checks if DEPQ is empty and returns true if empty.

2. Size(): Returns the total number of elements present in the DEPQ.

3. GetMin(): Returns the element having least priority.

4. GetMax(): Returns the element having highest priority.

5. put(*x*): Inserts the element *x* in the DEPQ.

6. BuiltDEPQ(*FileName*): Built the DEPQ with all the words in the input file. The input file is of the following format:

   <# words in the text file>
   word1
   word2
   .
   .
   .
   word n

7. RemoveMin(): Removes an element with minimum priority and returns this element.

8. RemoveMax(): Removes an element with maximum priority and returns this element.

9. IsContain(*x*): Whether a given element is there inside the DEPQ. This function should take execution time in O(1).

The elements in the heap will be words in English of any size. Given a sequence of words, the priorities of the words are considered in the alphabetical order, i.e, the first word in the alphabetical order is of highest priority and the last word in the order is of the lowest priority.

Task 2 (40 marks, +10 if the implementation is optimal)
Government collects huge amount of data by registering citizens for Adhaar Card. Suppose you have been given the job to implement a functionality where for a given set of names of the citizens, your function can sort the names in alphabetical order. You have to implement a external quick sort algorithm to solve the problem. The algorithm described above is the benchmark algorithm, you are free to implement a better version with less execution time to gain extra credits.

For simplicity assume that the input is give in a text file as described previously. Assume that the size of DEPQ as 100 (it can stores 100 strings of any arbitrary length), and the files as well as the words can be of any length. **You should not use any other data structure to store the data from the files.** You should read the data from the file and directly store it to the DEPQ.

The language to be used for the implementation has to be Python. Your code should output the sorted list of words in a file and the execution time required for the operation.

Important Note
You can not import any package in your code. Marks will be not be given if you do so.

Deliverables:

You will submit your Python programs and a note explaining your implementation and the time complexity of the algorithms used in a single rollno_a6.tar.gz file in the moodle submission link. It is mandatory that your code should follow proper indentation and commenting style. There will be deductions in the awarded marks, if you fail to do so.