CS7960 L4 : Basic I/O data structures + sorting

Disk <---I/O---> RAM <--> CPU
N = size of problem
B = block size
M = size of memory
T = size of output
I/O = block move between disk + memory

Basic Data structures:

Stack: FILO
```
 <-> [][][][ .... ]
```
Maintain push/pop blocks in RAM

Queue: FIFO
```
push ->  [][....][][]  -> pop
```
Maintain push and pop blocks in RAM

O(N/B) push/pop operations

Sorting :

< M/B sorted lists (queues) merged in O(N/B) I/Os

```
[....][] -> | |
[....][] -> | |
 ...     -> | | -> [][....]
[....][] -> | |
```

Unsorted list (queue) distributed w/ < M/B splits in O(N/B) I/Os

```
             | | [][.....]
[.....][] -> | | [][.....]
             | | [][.....]
             | | [][.....]
```

Merge Sort: <how to do it?>
 - create N/M (size M) sorted lists
 - merge list together O(M/B) at a time

```
                              <number>
[ ...                   ... ]  1
[M][M][M] [M] ...      ...[M]  N/M  (sorted v)
```

```
 [M^2/B] ...          [M^2/B]  (N/M) / (M/B)
  [M^3/B^2] ...                (N/M) / (N/B)^2
            ....               ...
   [N ....              ...]   1
```

O(log_{M/B} (N/M) rounds, using O(N/B) I/Os each
-> O((N/B) log_{M/B} (N/B)) I/Os

Do you use Merge sort in internal memory?
- *quick*, heap, bucket?

Selection

Find median in O(N/B) time.
http://www.ics.uci.edu/~eppstein/161/960130.html

Median(D,k=N/2)
Input: Data set D, size N.
(1) Partition D into sets of size 5. Find median of each -> M size N/5.
(2) m = Median(M,|M|/2)
(3) L items l in D w/ l < m
    R items r in D w/ r < m
(4) - if |L| = N/2-1 return m
    - if |L| > N/2 return Median(L,k)
    - else return Median(R,k-|L|-1)

What is runtime T(N)?
- Step (1)+(3) in O(N/B) I/Os
- Step (2) in time T(N/5)
- Step (4) in time at most T(N(7/10))

T(N) = O(N/B) + T(N/5) + T(N(7/10)) = ???
     = O(N/B) I/Os

[Generalizes to any k]

____

Quick Sort ("Distribution Sort")

(1) Compute Theta(M/B) splitting elements
   O(M/B) * O(N/B) = O(MN/B^2)
(2) Compute O(M/B) unsorted lists of equal size
(3) Recur on each list
    T(N) = O(N/B * (M/B)) + (M/B) T(NB/M)
         = O(???)

= O((M/B) * N/B log_{M/B} (N/B))

Extra (M/B) term -- Any ideas?
A: Find sqrt{M/B} elements in O(N/B) I/Os
- partitions lists into size at most (3/2) N/sqrt{M/B}
O((N/B) log_{sqrt{M/B}} (N/M)) = O((N/B) log_{M/B} (N/B))

Sorting Lower Bound:
Omega((N/B) log_{M/B} (N/B))

even stronger, permuting takes Omega((N/B) log_{M/B} (N/B)).
Takes Theta(N) in internal memory.