This lecture continues our theme of cache-oblivious data structures.

First, we'll finally cover the black box we used last lecture to obtain cache-oblivious B-trees: maintain an ordered file with $O(1)$-size gaps in $O(\lg^2 N)$ moves per insert/delete in the middle of the file. As an extra bonus, we'll see how to maintain a linked list subject to order queries in $O(1)$ time per insert/delete, which is a black box we used back in Lecture 1 to linearize the version tree when implementing full persistence.

Second we'll cover cache-oblivious priority queues, supporting insert and delete-min in what's usually sub-constant time per operation (!), $O((1/B) \log_{M/B} (N/B))$. This will be the first time we see how to adapt to the cache size M, not just the block size B, in a cache-oblivious data structure.

6.851 Advanced Data Structures

Spring 2012