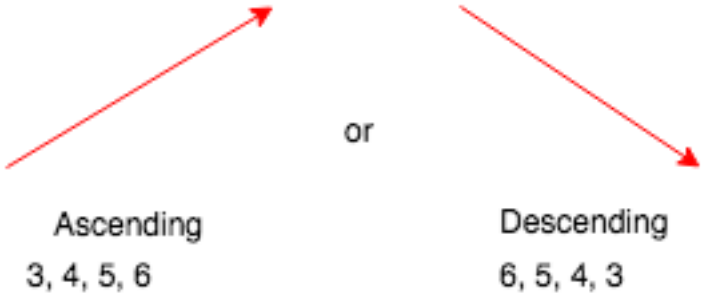
EE 382C/361C: Multicore Computing **Fall 2016**

Lecture 16: October 20

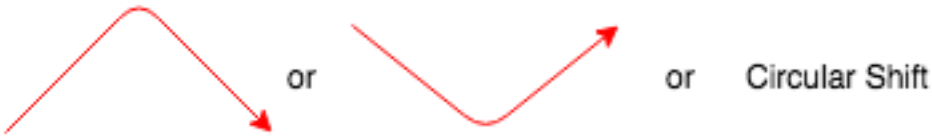
Lecturer: Vijay Garg *Scribe: Yanfeng Zhao*

16.1 Bitonic Sort

16.1.1 Monotonic Sequence



16.1.2 Bitonic Sequence



Def: $X_1 \dots X_n$ is a **bitonic sequence** if there exist k s.t. $X_1 \leq X_2 \leq \dots \leq X_k$ and $X_k \geq X_{k+1} \geq X_{k+2} \geq \dots \geq X_n$ or any circular rotation.

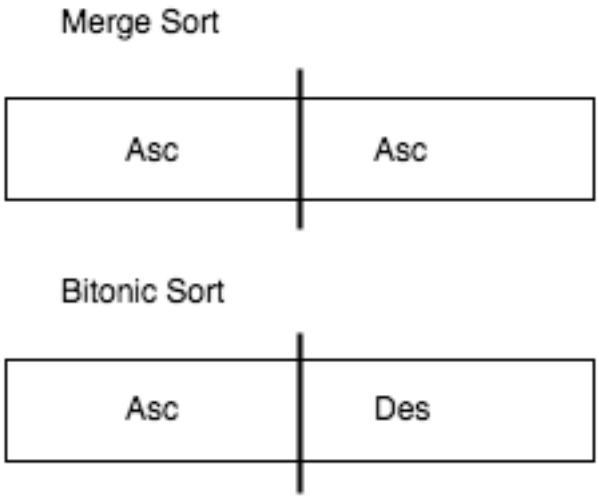
For example:

Bitonic sequence (k = 4): 1, 3, 5, 11, 9, 4, 2

Circular Shift Bitonic Sequence: 2, 1, 3, 5, 11, 9, 4. Notice that 2 was shifted to the front.

Any monotonic sequence is also bitonic sequence.

16.1.3 Why is this useful?



Bitonic Merge

$X_1 \dots X_{n/2} -> $ asc

$X_{n/2+1} \dots X_n$ -> des

Max

- $Y_1 = \max(X_1, X_{n/2+1})$
- $Y_2 = \max(X_2, X_{n/2+2})$
- ...

Min

- $Z_1 = \max(X_1, X_{n/2+1})$
- $Z_2 = \max(X_3, X_{n/2+3})$
- ...

Bitonic Sort Example:

2 7 11 13 9 5 3 1

2 5 3 1 < 9 7 11 15

2 1 < 3 5 < 9 7 < 11 13

1 < 2 < 3 < 5 < 7 < 9 < 11 < 13

16.2 Bitonic Sort Algorithm

BitonicSort (lo, hi) // assume size = 2^k for some k

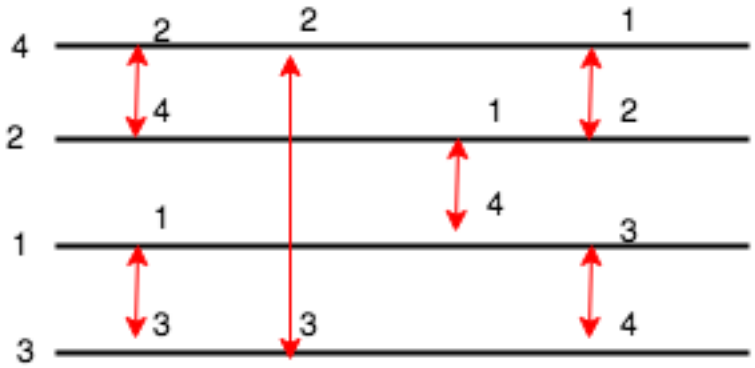
Base Case: If array Size == 1, then return

Recursion:

1. Bitonic Sort left side of the array (left half)
2. Bitonic Sort right side of the array (right half)
3. Compare and exchange if needed pairwise
4. Bitonic Merge (L)
5. Bitonic Merge (R)

Steps 3-5 are part of the bitonic merge.

Example:



16.3 Bitonic Sort Time Complexity Analysis

The algorithm in short:

- BitonicSort(L)
- BitonicSort(R)
- BitonicMerge(L, R)

Notice BitonicSort(L) and BitonicSort(R) can be done in parallel.

$T(n) = T(n/2) + log(n)$

$T(1) = O(1)$

$T(n) = T(n/2) + log(n)$

$= T(n/4) + log(n-1) + log(n)$ ... $= 1 + 2 + 3 + ... + log(n)$

$= O(log^2 n)$

References

[1] V. K. Garg, Introduction to Multicore Computing