Odd-even Mergesort

➤ The main idea revolves around the Odd-Even Merge operation which merges two sorted sequences A, and B.

Notation:
Let $A = [a_0, ..., \ldots, a_{n-1}]$ and $B = [b_0, ..., \ldots, b_{n-1}]$ two sorted arrays. Define

$$E(A) = [a_0, a_2, \ldots, a_{n-2}]; \quad O(A) = [a_1, a_3, \ldots, a_{n-1}]$$

and similarly for B. We will also use the notation $\overline{a, b}$ to denote the sorted version of a, b.

Issue: how to sort the union of A and B into one sorted array M.

➤ Here is the algorithm to produce M:

ALGORITHM : 1. *OEmerge*

0. **Input:** A, B *two sorted arrays of length* n
   Output: *Sorted array* M *of union of* A *and* B
1. *Obtain* $C = OEmerge\{E(A), O(B)\} \equiv [c_0, ..., c_{n-1}]$
2. *Obtain* $D = OEmerge\{O(A), E(B)\} \equiv [d_0, ..., d_{n-1}]$
3. $M = [\overline{c_0, d_0}, \overline{c_1, d_1}, \ldots, \overline{c_{n-1}, d_{n-1}}]$

➤ Steps 1 and 2 are recursive and they are also parallel.

Example: Consider the two sorted sequences

$$A = [2, 4, 6, 7] \qquad B = [1, 3, 5, 8]$$

Then:

$$E(A) = [2, 6], \quad O(A) = [4, 7]$$
$$E(B) = [1, 5], \quad O(B) = [3, 8]$$

➤ Step 1 merges (recursively) [2, 6] with [3,8]

Result: $C = [2, 3, 6, 8]$

➤ Step 2 merges (recursively) [4,7] with [1,5]

Result: $D = [1, 4, 5, 7]$

➤ Step 3 interleaves the two arrays

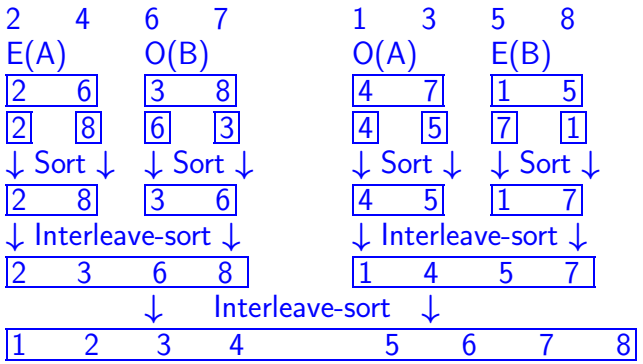
Result: $[2, 1, 3, 4, 6, 5, 8, 7]$

➤ Finally, rearrange unordered pairs (c_i, d_i):

Result: $M = [1, 2, 3, 4, 5, 6, 7, 8]$

Cost analysis: Unfolding the recursive calls

➤ See how the recursive calls unfold (previous example)



Odd-Even Mergesort: The algorithm

Q: How can we use this to sort a sequence of numbers?

A: Build sorted sublists bottom up - starting with lists of size 2, and merging sublists into bigger ones

Odd-Even Mergesort:

- Sort arrays $[a_0, a_1]$, $[a_2, a_3]$ etc.. $[a_{n-2}, a_{n-1}]$
- Merge $[a_0, a_1]$, $[a_2, a_3]$ into $[a_0, a_1, a_2, a_3]$, $[a_4, a_5]$, $[a_6, a_7]$ into $[a_4, a_5, a_6, a_7]$, etc...
- Continue merging larger and larger subsets until the whole array is sorted..

A complete example from start

```
 Bottom level:

7  6  2  4   5  3  8  1  --> sort all pairs:
----  ----   ----  ----

6  7  2  4   3  5  1  8
--A-  -B--   -A--  --B-  --> Merge by OE Merge

EA OB OA EB  EA OB OA EB
6  4  7  2   3  8  5  1  --> Sort each set (merge
----  ----   ----  ----      pairs of singletons)
A  B  A  B   A  B  A  B

--C-- --D--  -C--  -D--
4  6  2  7   3  8  1  5  --> Interleave:
----  ----   ----  ----

2  4  6  7   1  3  5  8
---- A----   ---- B ---    --> OE Merge sort ...
```



```
---- A----      ----- B ----  --> recursion
2  4  6  7     1   3   5  8  OEmerge(A,B)

--EA-  -OB--    --OA-- --EB-
2   6  3  8     4   7   1  5   recursion
-----  ----     -----   ----   OEmerge

EA  OB  OA EB    EA OB OA EB
2   8   6  3     4  5  7  1
-----   -----    -----  ----   Sort
2   8   3  6     4  5  1  7
--C--   --D--    --C--  --D-   Interleave->

2   3   6   8    1  4   5  7
-----C-------    ---D-------   Interleave->

1  2    3   4    5  6   7  8   <-- final array
```


Complexity

✍D1 How many steps are there?

✍D2 What is the number of sequential operations?

The Odd-even mergesort algorithm consists of sorting larger and larger arrays - using the merge operation. Start with arrays of length 2. Then merge pairs of arrays of length 2 into sorted arrays of length 4. etc. The total number of operations is $O(\log_2^2(n))$.

A slightly simpler implementation

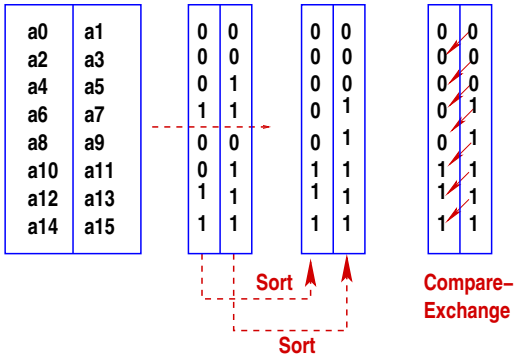
- Use a different version of OEmerge
- Formulated *in-place*

1. OEmerge even part of A: $a_0, a_2, \cdots, a_{n-1}$
2. OEmerge odd part of A: $a_1, a_3, \cdots, a_n$
3. Sort pairs: $A = \{a_0, \overline{a_1, a_2}, \overline{a_3, a_4} \cdots \overline{a_{n-2}, a_{n-1}}, a_n\}$

- See matlab implementation (sequential)

```
a0 a2 a4 a6 sort --> a0 a2 a4 a6
                      /  /  /  <--compare exchange
a1 a3 a5 a7 sort --> a1 a3 a5 a7
```


Proof of correctness : use induction



Bitonic Mergesort. Definition of bitonic sequences

A bitonic sequence has two sub-sequences, one increasing and one decreasing. For example,

$a_0 < a_1 < \cdots < a_{i-1} < a_i > a_{i+1} > \cdots > a_{n-2} > a_{n-1}$

for some $i (0 < i < n)$. ➤ A sequence is also bitonic if shifting the sequence cyclically (left or right) gives a sequence with above property.

Example: Sequences: [4 6 8 9 3 1] and [8 9 3 1 4 6] are bitonic

✍D3 Find an easy way to recognize a bitonic sequence..

Property of Bitonic Sequences: Performing a compare-exchange operation on a_i with $a_{i+n/2}$ for all i , yields *two* bitonic sub-sequences.

Example:

```
          <----Initial Bitonic sequence------->
          3    5    8    9    7    4    2    1
Compare-  ^--------------------^
exchanges      ^--------------------^
                    ^--------------------^
                         ^--------------------^
          3    4    2    1    7    5    8    9
          Bitonic sequence  |   Bitonic sequence
```

✍D4 What do you observe regarding these sub-sequences? [Hint: compare largest entry of one with smallest of other]

Bitonic Mergesort- Algorithm

➤ The algorithm will exploit one nice property which makes it easy to sort bitonic sequences.

Property: After compare-exchange operations with stride $n/2$, keys of the resulting left bitonic subsequence are all smaller than those of the right bitonic subsequence.

➤ See previous example

Sorting a bitonic sequence:

➤ Given a bitonic sequence, recursively perform compare-exchange operations on smaller and smaller sets [strides $n/2$, $n/4$, ...]

Example: Sorting a bitonic sequence – Process produces smaller and smaller bitonic sequences such that entries of left ones of a pair are smaller than entries in right one.

```
                2   4   7   8  |  6   3   1   0
Comp-exch       ^---------------^
stride= 4           ^-----------------^ etc..

                2   3  | 1   0  |  6   4  |  7  8
Comp-exch       ^-------^
stride = 2          ^--------^ etc..

                1 | 0 | 2 | 3 | 6  | 4 | 7 | 8
Comp-exch       ^---^
stride = 1              ^---^ etc..

Sorted list     0   1   2   3   4    6   7   8
```


Bitonic sorting Uses a bottom up approach.

1. Build adjacent pairs of numbers that ↑ and ↓
a_1, a_2: ↑, a_3, a_4 ↓, etc.

3. Use previous idea of sorting to sort each pair of pairs into increasing numbers and them decreasing numbers, so now a_1, a_2, a_3, a_4 is ↑, and a_5, a_6, a_7, a_8 is ↓, etc.

4. Repeat this process. Bitonic sequences of larger and larger lengths are obtained.

5. In the final step, a single bitonic sequence is sorted into a single increasing sequence.

➤ Total cost is similar to OE-Mergesort: $O(\log^2(n))$ using $n/2$ processors.

A complete example

Phase 1

```
---unsorted sequence---     --> sort all pairs
7  6  2  4   5  3  8  1     --> into Up and Down
----  ----   ----  ----
6  7  4  2   3  5  8  1     <-- Result
--->  <---   --->  <---
```

Phase 2

```
6  7  4  2  |  3  5  8  1   --> Comp/exch on each part
--->  <---  |  --->  <---   stride=2. Up for 1st half
                           Down for 2nd half
4  2  6  7  |  8  5  3  1   --> Repeat with stride=1:
2  4  6  7  |  8  5  3  1   <-- Result
---------->    <---------
```

Phase 3

```
2  4  6  7   |   8   5   3  1  <-- Compare exchange
--------->       <-----------  Stride=4

2  4 | 3  1  |  8  5  |  6  7  <-- Compare exchange
                                 Stride=2

2 | 1 | 3 | 4 | 6 | 5 | 8 | 7  <-- Compare exchange
                                 Stride=1

1   2    3   4   5   6   7   8 <-- Final Result
```

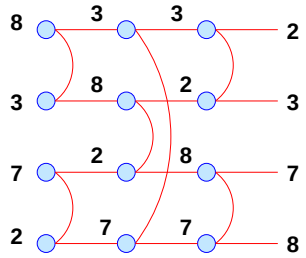

Sorting networks

Odd-even merge-sort and bitonic mergesort belong to a class of sorting algorithms based on "sorting networks".

- Multistage Networks designed specifically for the task of sorting.

Example: Sequence $\boldsymbol{A} = [\mathbf{8}, \mathbf{3}, \mathbf{7}, \mathbf{2}]$. Arrows indicate direction of comparator Min (down) or Max (up)]

Odd-even Mergesort



bitonic Mergesort