2017-09-28

- Hanoi: $T(n) = \begin{cases} 1 \text{ if } n = 1 \\ 2T(n-1) + 1 \text{ if } n > 1 \end{cases}$ $\therefore T(n) = 2^n - 1 = O(2^n)$.
- Merge sort: $T(n) = \begin{cases} O(1) \text{ if } n = 1 \\ 2T(\frac{n}{2}) + O(n) \text{ if } n > 1 \end{cases}$ $\therefore T(n) = O(n \log n)$.
- Bitonic champion problem: $\Theta(\log n)$
  - Lower bound: any comparison-based algorithm needs $\Omega(\log n)$ time in the worst case.
  - Upper bound by divide and conquer: $T(n) = \begin{cases} O(1) \text{ if } n = 1 \\ T(\frac{n}{2}) + O(1) \text{ if } n > 1 \end{cases}$ $\therefore T(n) = O(\log n)$.
- Maximum subarray problem: $\Theta(n)$
  - Lower bound: $\Omega(n)$.
  - Upper bound by divide and conquer: $T(n) = \begin{cases} O(1) \text{ if } n = 1 \\ 2T(\frac{n}{2}) + O(n) \text{ if } n > 1 \end{cases}$ $\therefore T(n) = O(n \log n)$.
  - Upper bound by dynamic programming: $O(n)$