Can't sort faster than \( \Omega(n \log n) \) comparisons. Not do, more.

Sort \( n \) numbers in the range \([1...n]\).

Input \( x_1, x_2, ..., x_n \)

\( x_i \in [1...n] \)

Can sort using "count sort" count the \# 1's, \# 2's, ..., \# n's counters for \( i \leq i < n \) increment the counter correspondingly.

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & i & n \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{count} & 1 & 2 & 0 \\
\end{array}
\]

Output \[
\begin{array}{cccc}
5 & \\ 2 & \\ 44 & i \\
\end{array}
\]
\[
\text{count}[1], \text{count}[1] + \text{count}[2], \\
\text{count}[1] + \text{count}[2] + \text{count}[3]
\]

Support inputs are \(y_1, y_2, y_3, \ldots, y_n\)

0. \(y_1\)

0. \(y_1 + y_2\)

partial sums

\[
\sum_{j=1}^{i} y_j
\]

(6) \(y_1 + y_2 + \ldots + y_n\)

Parallel computation of partial sums
Using $\frac{n}{2}$ processors, we can sum $n$ numbers in $\log n$ rounds.

Total effort: $\#\text{processors} \times \#\text{rounds}$

Total work: $O(n \log n)$

Always compare with total sequential time.

Use $\frac{n}{\log n}$ processors.

Initially, each processor adds up $\log n$ number each sequentially.

Total time: $O(\log n)$

Subsequently, we have $\frac{n}{\log n}$ no.s and $\frac{n}{\log n}$ processes.
Use the tree based computation

$$
\Rightarrow \log \left( \frac{n}{\log n} \right) \text{ rounds } \frac{n}{\log n} \text{ processors}
$$

$$
\Rightarrow O(\log n) \quad \Rightarrow \quad O(n)
$$

2 logn parallel steps

For partial sums, if we employ different sets of processors for each term, then

$$
\Rightarrow \frac{n + n - 1}{\log n \log n} \ldots
$$

$$
= O\left(\frac{n^2}{\log n}\right)
$$

Total work \quad \frac{n^2}{\log n} \times \log n = O(n^2)
What is time

What is the # addition operations (adder circuits)

\[ T'(n) = T'(n/2) + 2 \]

\[ \implies T'(n) = 2 \log n \]

\[ S(n) = S(n/2) + n \]

\[ \implies S(n) \leq 2n \text{ i.e. } O(n) \]

Parallel prefix / scan operation

It generalises to any "associative" operator

\[ y_1 \circ y_2 \circ \cdots \circ y_n \]

associative