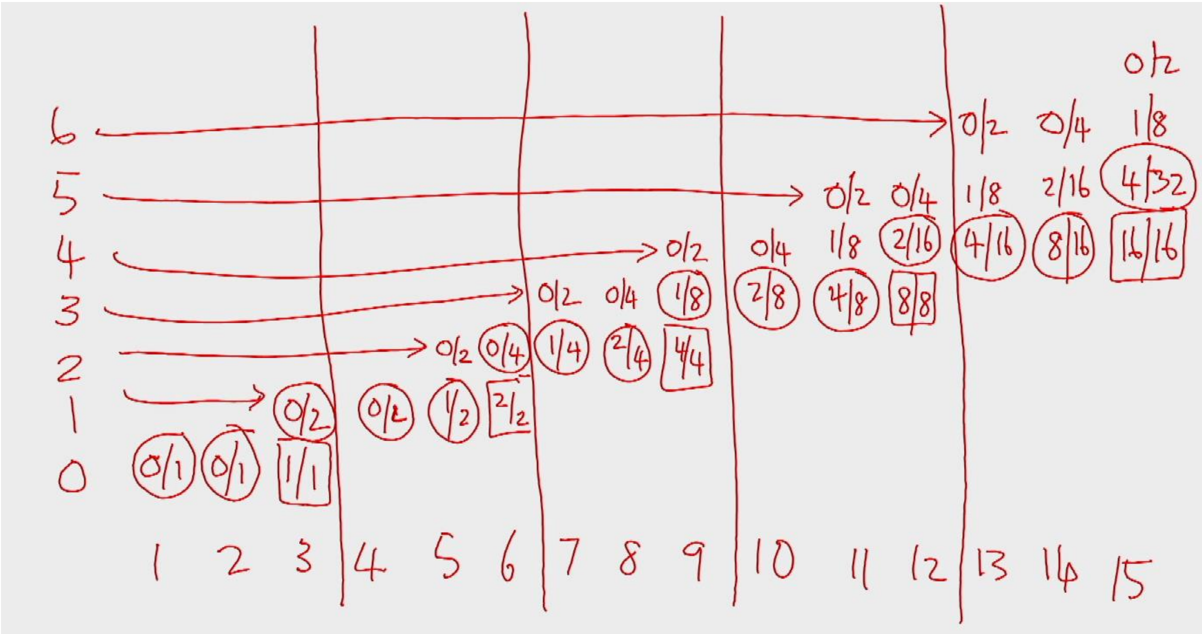


In stages 3r

$\longrightarrow n/64$ ⋮

$\longrightarrow n/8$

$\longrightarrow n$

15) $32 \to 8 \to 2$

Total cache size at the parent : $\frac{1}{8}$th cache size at children

$$n + \frac{n}{8} + \frac{n}{64} + \cdots < \frac{8n}{7} = O(n)$$

In stages that are $3r+2$

$$1 \to \frac{1}{4} \to \frac{1}{4} \to$$

$$\frac{1}{2} \to \frac{1}{8} \to \frac{1}{8} \to \frac{1}{8} \to \frac{1}{8}$$

$$n \to \frac{n}{2} \to \frac{n}{16} \to \frac{n}{128} \text{ —}$$

$$n + \frac{n}{2} + \frac{n}{16} + \cdots < \frac{11n}{7} \in O(n)$$

The total # elements in all the sample arrays is $O(n)$

Formal proof

Cole's Merge sort

runs in $O(\log n)$ time

using n processors

On ~~CREW~~ PRAM

EREW

Can we do faster?

Comparison Complexity of

Sorting $\Omega(\log n)$ time

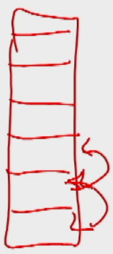
with n processors

A Lower Bound for Comparison Based Sorting

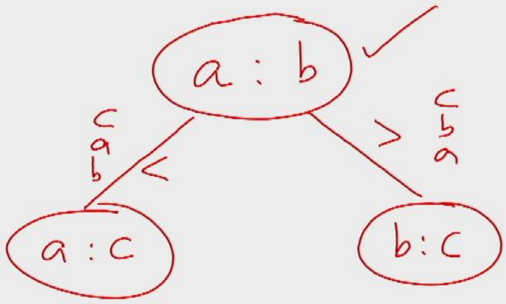
LB: $\Omega(n \log n)$ time

Decision trees.

Bubble Sort for size n



c
~~b~~ a
~~a~~ b



The first comparison is hardcoded in the algorithm

Bubble sort: Two bottom most

Insertion sort: Two left most

Merge Sort: Two left most

Quick Sort (Left most = pivot)

pivot, the leftmost of the remaining

Parallel Comparison Model

only comparisons count
every other op : 0 cost
0 time

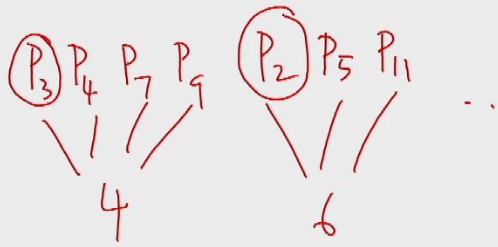
PCM $\geq$ PRIORITY CRCW PRAM

---

one step of PRIORITY

0 time / Radix Sort { Let us sort the processors on the addresses they want to write

(address, pr. no)



$c(t,n)$: # comparisons that any algorithm that sorts n nos in t comparison steps must necessarily perform

Claim $\quad c(t,n) > \dfrac{t\,n^{1+1/t}}{e} - tn$

Proof Basis

✓ $t=1 \quad c(1,n) = \dfrac{n(n-1)}{2} > \dfrac{n^2}{e} - n$ ✓

✓ $t \geq 1 \quad c(t,1) = 0 > \dfrac{t}{e} - t$

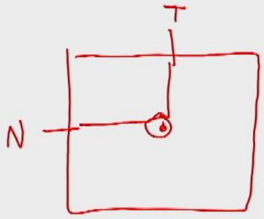
Hypothesis

$\forall\, t < T$ and $n \leq N$

or

$t \leq T$ and $n < N$

$$c(t,n) > \frac{t n^{1+1/t}}{e} - tn$$



$N \times T$

---

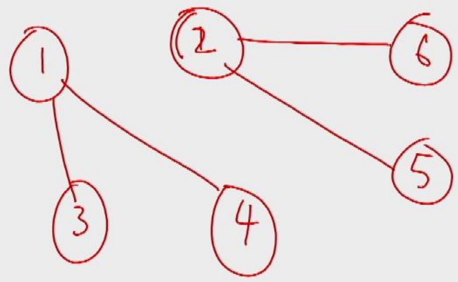
an input of size N

your algorithm sorts this input in T steps

Consider the 1st step of your algorithm

Construct a graph

vertices are the input positions



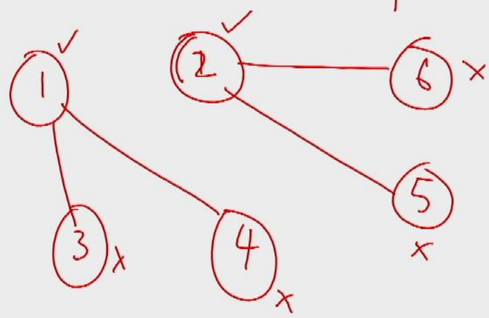
Find a maximal independent set of this graph

IS: a set of vertices no two of which are adjacent

MIS: an IS that is maximal

Construct a graph

vertices are the input positions



$\{1, 2\}$

Cook up an input

to make your algo. do badly.

Take N elements from a linearly ordered set

$|MIS| = x$

Place the largest x on the MIS positions

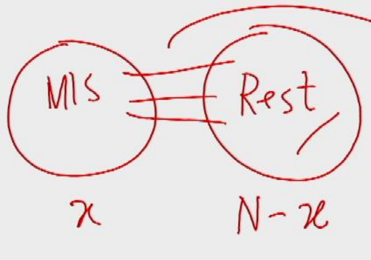
The remaining posns are filled anyhow

Run the algorithm on this input.

In the 1st step,

nothing is done for sorting the elements on MIS

at least $\underline{\underline{N-x}}$



1st step

× sorting the MIS elements

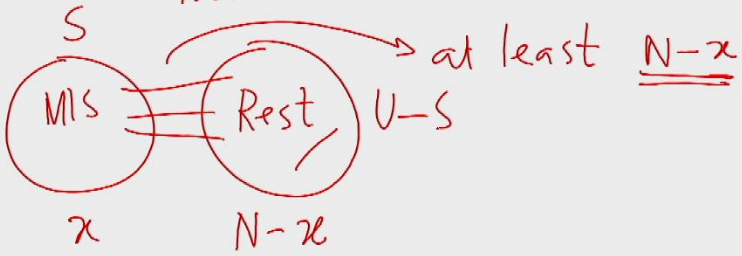
× comparison (MIS elems, other)

do not contribute to the sorting of either set

Run the algorithm on this input.

In the 1st step,

nothing is done for sorting the elements on MIS



Benefit of doubt

$U-S$ is on its way to being sorted in the best possible way

$$c(N, T) \geqslant c(T, N-x) + (N-x) + c(T-1, x)$$

Sort U, $|U| = N$

S, $\underbrace{U-S}$

$$c(T, N) > T\left[\frac{(N-x)^{1+1/T}}{e} - (N-x)\right] + (N-x) + (T-1)\left[\frac{x^{1+1/(T-1)}}{e} - x\right]$$