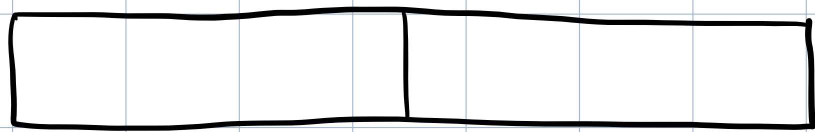
06 Jan 2025 – Algorithms – Week 02

Divide and Conquer



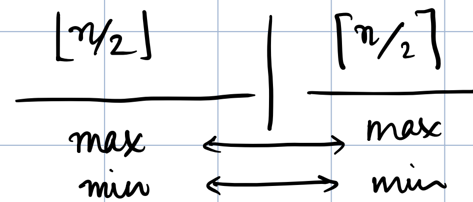
Input

1. divide into 2 or more pieces.
2. Solve on each piece.
3. combine the solution of each piece.

Eg 1: Binary Search

Eg 2: Max and min of n elements

$$T(n) = T\left(\lfloor n/2 \rfloor\right) + T\left(\lceil n/2 \rceil\right) + 2$$



$$T(n) = \frac{3n}{2} - 2 \quad \text{if } n \text{ is a power of } 2$$

Find max and min in $\sim 3n/2$ comparisions

# of comparisions

$\frac{n}{2}$ $\quad \underbrace{a_1 \quad a_2}_{\text{compare}} \;\Big|\; a_3 \quad a_4 \;\Big|\; \cdots \;\Big|\; \underbrace{a_{n-1} \quad a_n}_{\text{compare}} \;\Big|$

---

① Find minimum $\frac{n}{2}$ elements and maximum $n/2$ elements separately

② A = list of all min elements $\longrightarrow$ recursse

B = list of all max elements $\longrightarrow$

$$\frac{n}{2} + 2\left(\frac{n}{4} + \frac{n}{8} + \cdots \frac{n}{2^k}\right)$$

$$= \frac{3n}{2}$$

$$\frac{\cancel{n}}{\cancel{2}} \times \frac{2}{\cancel{n}} \frac{(1/4)\left((1/2)^{k-1} - 1\right)}{\cancel{(1/2 - 1)}}$$

Given an input I and an algorithm A,

$$T_A(I) = \#\text{ time steps taken to output}$$

Worst-case time complexity

$$T_A(n) = \max_{|I| = n} T_A(I)$$

$$\rightsquigarrow T(n)$$

Asymptotic Notation:

Insertion sort takes $\leq 4n^2 - 2n$ operations

$$T(n) \leq 4n^2 - 2n$$

$$= O(n^2)$$

$f(n) = O(g(n))$ if $f(n) \leq c\, g(n)$ for constant c and all n sufficiently large.

$$4n^2 - 2n + 100 = O(n^2)$$

$$4n^2 - 2n + 100 \leq 104\, n^2 \qquad \forall n \geq 1$$

$$4n^2 - 2n + 100 \leq 5n^2 \qquad \forall n \geq 10$$

Polynomials: $100\, n^2 = O(n^{2.5})$; $n^{2.5} = O(n^3)$

$n^c = O(n^{c'})$; $c < c'$

$\log^{100} n = O(n^{0.0001})$

$n^{100} = O(1.1^n)$

any logarithmic function is smaller than polynomial f^n

any polynomial function is smaller than exponential f^n

$100 \quad n\log n \quad vs \quad n^{1.5}$

$\equiv \quad \log n \quad vs. \quad n^{0.5}$

$100\, n \log n = O(n^{1.5})$

$(\log n)^n \quad vs \quad n \quad (= e^{\log n})$

$n \log \log n \quad vs \quad \log n$

$$f(n) = o(g(n))$$

if $$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

$$\lim_{n\to\infty} \frac{2}{n^{1.5}} = 0$$

$$f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$$

$$f(n) = O(g(n))$$

$$h(n) = O(g(n))$$

$$f(n) + h(n) = O(g(n))$$

$f(2n) = O(f(n))$

if f is polynomial or smaller

Eg: $(2n)^3 = O(n^3)$

$\log 2n = O(\log n)$

→ summation constant no. of times

$$\underbrace{1 + 1 + \cdots + 1}_{n \text{ times}} \neq O(1)$$

$$T(n) = \sum_{i=1}^{n} f(i)$$

$$T(n) \sim \int_1^n f(x)\, dx \quad , \text{ if } f \text{ is monotone}$$

$$\sum_{i=1}^{n} \frac{n}{i} = n \cdot \sum_{i=1}^{n} \frac{1}{i}$$

$$\approx n \int_1^n \frac{1}{x}\, dx$$

$$\sim n \log n$$

$$\underbrace{\sum_{i=1}^{n} \sqrt{i}}_{\leq n\sqrt{n}} \approx \int_1^n \sqrt{x}\, dx \sim n^{3/2}$$

$$= \Theta\left(n^{3/2}\right)$$

$$f(n) = O(g(n)) \quad \Big| \quad \leq$$

$$f(n) = \Omega(g(n)) \quad \Big| \quad \geq$$

$$f(n) \geq c \cdot g(n) \quad \forall \; n \geq n_0$$

$$f(n) = \Theta(g(n))$$

if $f(n) = O(g(n)) \wedge g(n) = O(f(n))$

$$4n^2 + n = \Theta(n^2)$$

$$\sum_{i=1}^{n} a^i \longrightarrow \text{if } a > 1 : \Theta(a^k)$$

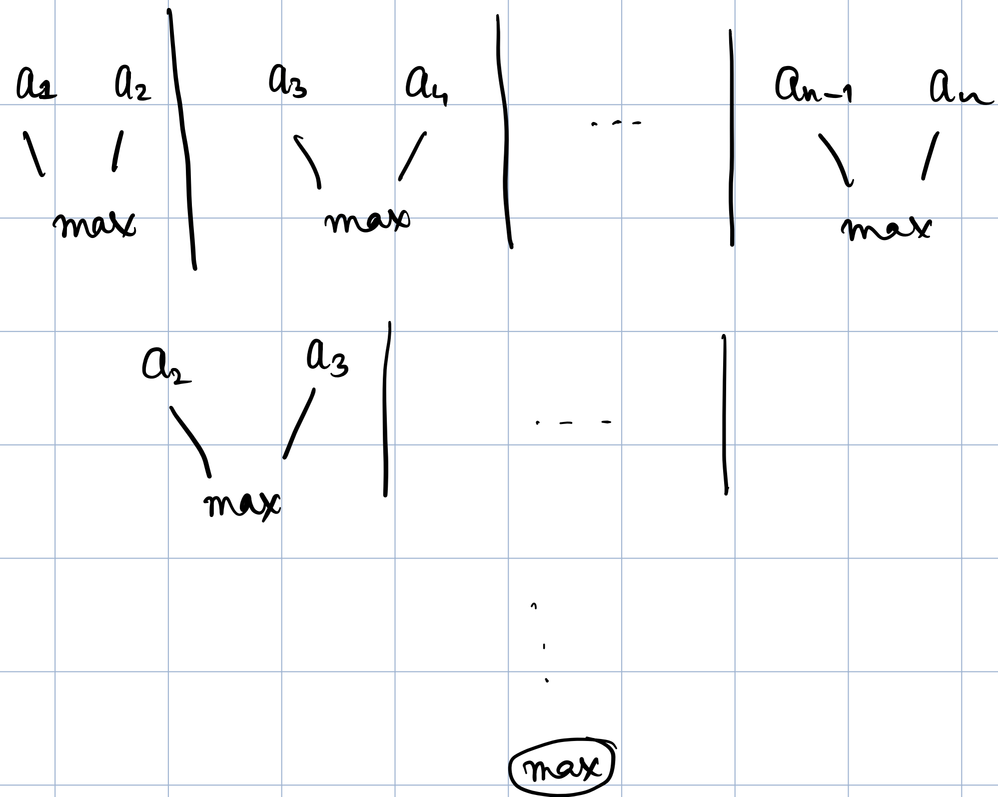
$$\frac{a^{k+1} - 1}{a - 1}$$

$\downarrow$

if $a < 1$: $\Theta(1)$

$$1 \leq 1 + \frac{1}{2} + \frac{1}{4} + \dots = 2$$

Find max and second max in $\leq n + \log_2 n + 1$ comparisions



Merge Sort

→ sort first and second halves

MergeSort (A, first, last)

mid = (first + last) / 2

1 MergeSort (A, first, mid)

2 MergeSort (A, mid+1, last)

Merge (1, 2)

A	2	4	5	7	∞

B	1	3	6	9	11	∞

→ In-place merge sort

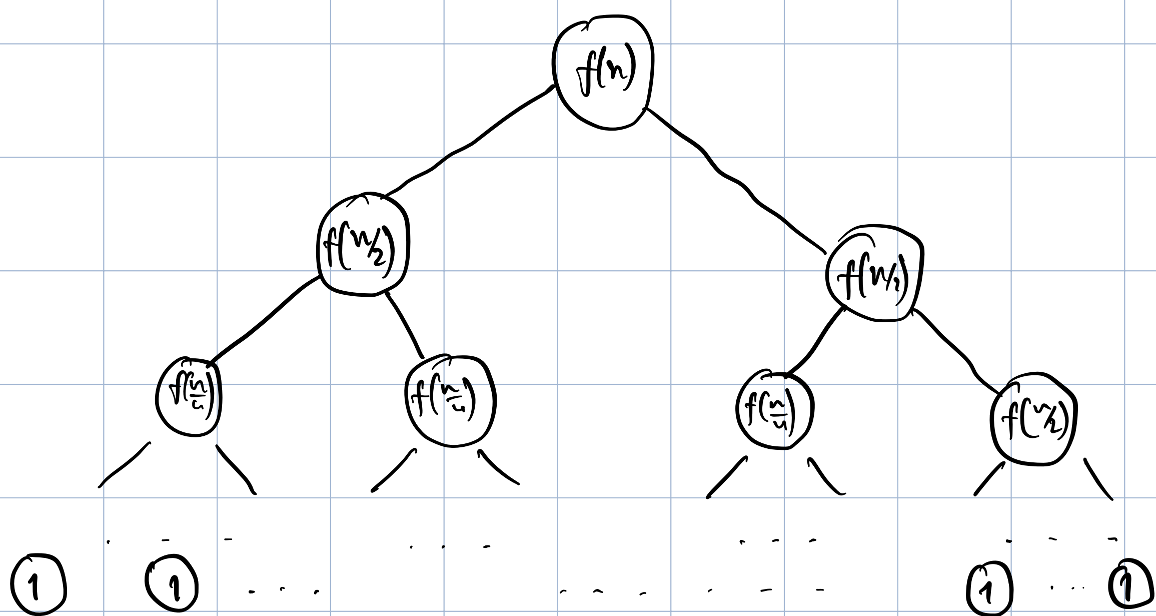
Merge Sort Analysis:

$O(n)$

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

time to combine n solutions

Method 1: Recursion tree $T(n) = 2T(n/2) + f(n)$



instructions at level 1 = $f(n)$

level 2 : $2f(n/2)$

⋮

$k = \log n$

$$T(n) = f(n) + 2f(n/2) + \cdots + 2^k f\left(n/2^k\right)$$

if $f(n) = \Theta(n)$ $= n \cdot k$

$= n \log n$

n

n

Ex 1 : $T(n) = 2T(n/2) + \sqrt{n}$

$$T(n) = \sqrt{n} + 2\sqrt{\frac{n}{2}} + \cdots + n\sqrt{\frac{n}{n}}$$

$$= \sqrt{n} + \sqrt{2n} + \sqrt{4n} + \cdots + \sqrt{nn}$$

$$= \sqrt{n}\,\frac{(\sqrt{2})^{\log n} - 1}{\sqrt{2} - 1} = \sqrt{n}\,(\sqrt{n} - 1)(\sqrt{2} + 1)$$

$$T(n) = 8\,T(n/2)$$
$$= 8^2\,T(n/4)$$
$$= 8^3\,T\left(n/2^3\right)$$
$$= 8^k\,T\left(\frac{n}{2^k}\right) = 8^{\log n} = 2^{3\log n} = n^3$$

$$= \sqrt{n}\left(1 + \sqrt{2} + \cdots + \sqrt{2^k}\right)$$
$$= \Theta\left(\sqrt{n} \cdot \sqrt{2^k}\right)$$
$$= \Theta(n)$$

$$T(n) = a\,T(n/b)$$
$$= a^{\log_b n}$$
$$= n^{\log_b a}$$

Method 2: Master Theorem *

$$T(n) = a\,T(n/b) + f(n) \quad , \quad a, b > 1 \qquad n \in \mathbb{N}$$

compare $n^{\log_b a}$ and $f(n)$. If one of them is larger, that's the solution.

If both are equal, the solution is $T(n) = O(f(n) \log n)$

$$T(n) = 3T(n/2) + 5n$$

$\underset{\text{Larger}}{n^{\log_2 3}}$ vs n

1. $T(n) = 3T(n/5) + O(\sqrt{n})$

2. Divide into 3 equal pieces, sort and merge.
Write recurrence and analysis

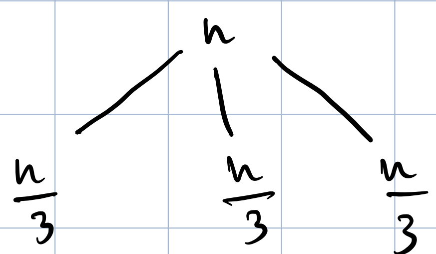
1. $n^{\log_5 3}$ vs. $n^{1/2}$

$\log_5 3 > 1/2$

$3 > 5^{1/2}$

$9 > 5$

2.



$T(n) = 3 \cdot T\left(\frac{n}{3}\right) + O(n)$

$n^{\log_3 3}$ vs n $\Big\}$ equal $O(n \log n)$

Doubts

① Insertion sort : $4n^2 - 2n$ comparisions

9 Jan 2025

* Master Theorem : Caveat

Ratio of larger to smaller function grows atleast polynomially.

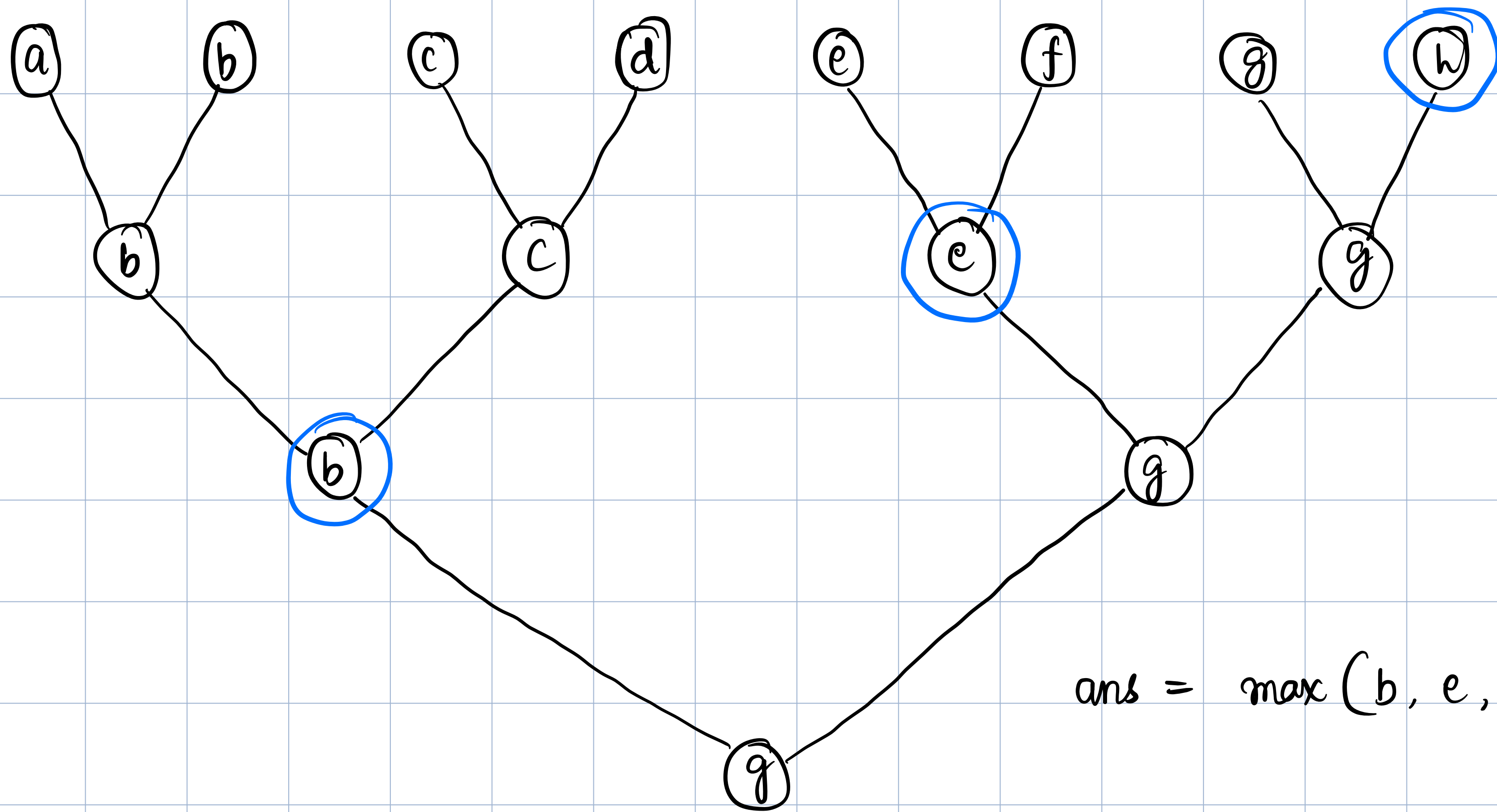
e.g.: $n^{2.5}$ and n, n^2 and $n \log n$

non-examples: n and $n \log n$

$$T(n) = a\,T(n/b) + f(n)$$

* $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{3}\right) + n$ } DON'T solve separately

Second - max element puzzle



ans = max (b, e, h)

Merge step:

$$\left.\begin{array}{l} A\,[1\ldots m] \\ B\,[1\ldots n] \end{array}\right\} \rightarrow \text{merge}$$

$\forall\ 1 \leq k \leq m+n$

Claim: The first k elements placed in C are the k smallest elements in $A \cup B$ in non-decreasing order

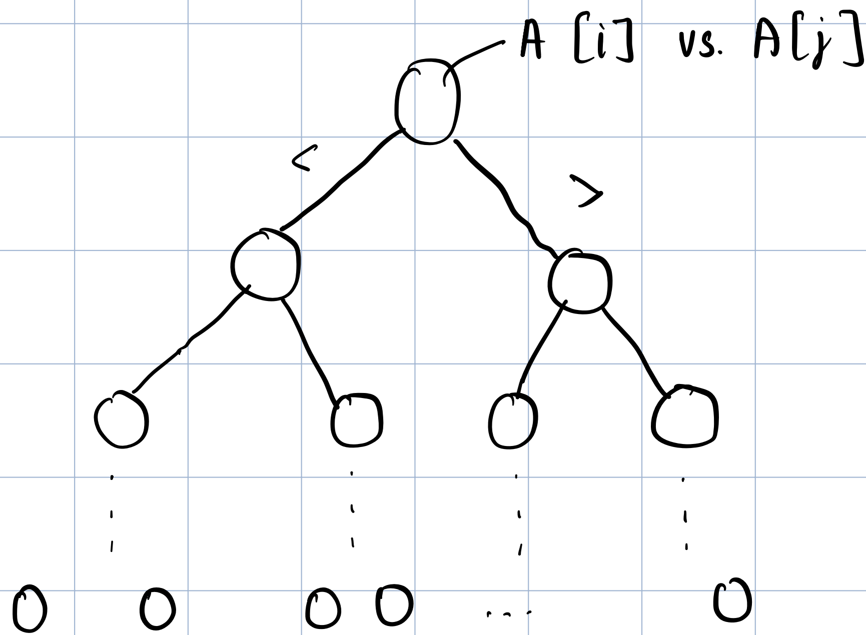
$\min\left(A[1], B[1]\right)$ is the smallest element

$A[i, \ldots, m]$

$B[j, \ldots, n]$

$\min\left(A[i], B[j]\right) =$ min of all remaining elements

Claim: Any comparision based algorithm must make $\Omega(n \log n)$ comparisons for sorting



Relation b/w decision tree height and size of A?

height = # of comparisions

leaves represent different permutations of the input end $\Bigg\}$ # leaves $\leq 2^{\text{height}}$

$\therefore$ # leaves = # of permutations of n elements $= n!$

$$\therefore \quad n! \leq 2^{\text{height}}$$

$$\log_2 n! \leq \text{height}$$

$$\Rightarrow \sum_{i=1}^{n} \log_2 i \leq \text{height}$$

$$\approx$$

$$\int_1^n \log x \, dx = n \log n - n$$

$$\therefore \quad \text{height} = \Omega(n \log n)$$

$a_1 \quad a_2 \quad a_3 \quad a_4$

- $\left.\begin{array}{ll} a_1 < a_2 \,, & a_3 < a_4 \\ a_1 < a_3 \,, & a_2 < a_4 \end{array}\right\} \rightarrow$ Not enough comparisions to sort

$a_1 \quad a_2 \quad a_3 \quad a_4$

$a_1 \quad a_3 \quad a_2 \quad a_4$

* comparisions identify a permutation.

The stock purchase problem

Given a list of a stock's prices, decide when to buy and sell

Input: A list of stock prices: $\{p_1, p_2, \ldots, p_n\}$

Output: $\max\{(p_i - p_j) : i < j\}$

```
vector <int> a(n+1);      a[0] = INT_MAX

for (int i=1  i < n+1;  i++);
    cin >> p[i];
    a[i] = p[i] - min ( a[i-1], p[i])
```
$\Big\}\ O(n)$

find max_element in a $\Big\}\ O(n)$

$p_1, \; p_2, \; \dots, \; p_n$

$p_1, \; \dots, \; p_{n/2} \;\Big|\; p_{\frac{n}{2}+1}, \; \dots, \; p_n$

recursively solve in each half

Buy in $\{ p_1, \dots, p_{\frac{n}{2}} \}$

Sell in $\{ p_{\frac{n}{2}+1}, \dots, p_{\frac{n}{2}} \}$

$$\max \Big\{ \min \{ p_1, \dots, p_{\frac{n}{2}} \} + \max \{ p_{\frac{n}{2}+1}, \dots, p_n \}, \; \text{sol}^n \, 1, \; \text{sol}^n \, 2 \Big\}$$

Multiplication of 2 integers

I/p : $A = A_1 \ A_2 \ \cdots \ A_n$

$B = B_1 \ B_2 \ \cdots \ B_n$

Size of input : n = # of bits of each number

$$\begin{array}{r} 1\ 1\ 1 \\ 1\ 0\ 1 \\ \hline 1\ 1\ 1 \\ 0\ 0\ 0\ 0 \\ 1\ 1\ 1\ +\ + \\ \hline 1\ 0\ 0\ 0\ 1\ 1 \end{array}$$

$A_1 \ A_2 \ \cdots \ A_n$

n rows of $\sim n$ (to $2n$) bits each

\# of operations $= \Theta(n^2)$

1950s : can we do better?

1960 : Karatsuba algorithm (divide and conquer ; master theorem)

$\rightarrow O(n^{\sim 1.585})$

1961 : Schonhage - strassen

$O(n \log n \log \log n)$

Based on fast Fourier transform (based on divide and conquer)

2007 : $O\left(n \log n \; 2^{\log^* n}\right)$

2019 : $O(n \log n)$

$A = A_1 \; A_2 \; \ldots \; A_n$

$B = B_1 \; B_2 \; \ldots \; B_n$

$A_L = A_1 \; A_2 \; \ldots \; A_{n/2}$, $A_R = A_{\frac{n}{2}+1}, \ldots, A_n$

$B_L = B_1 \; B_2 \; \ldots \; B_{n/2}$, $B_R = B_{\frac{n}{2}+1}, \ldots, B_n$

$A = 2^{n/2} \times A_L + A_R$

$B = 2^{n/2} \times B_L + B_R$

$$AB = A_L B_L \times 2^n + (A_L B_R + A_R B_L) \cdot 2^{n/2} + A_R B_R$$

$217 \quad 846$

$A_L = 217, \quad A_R = 846$

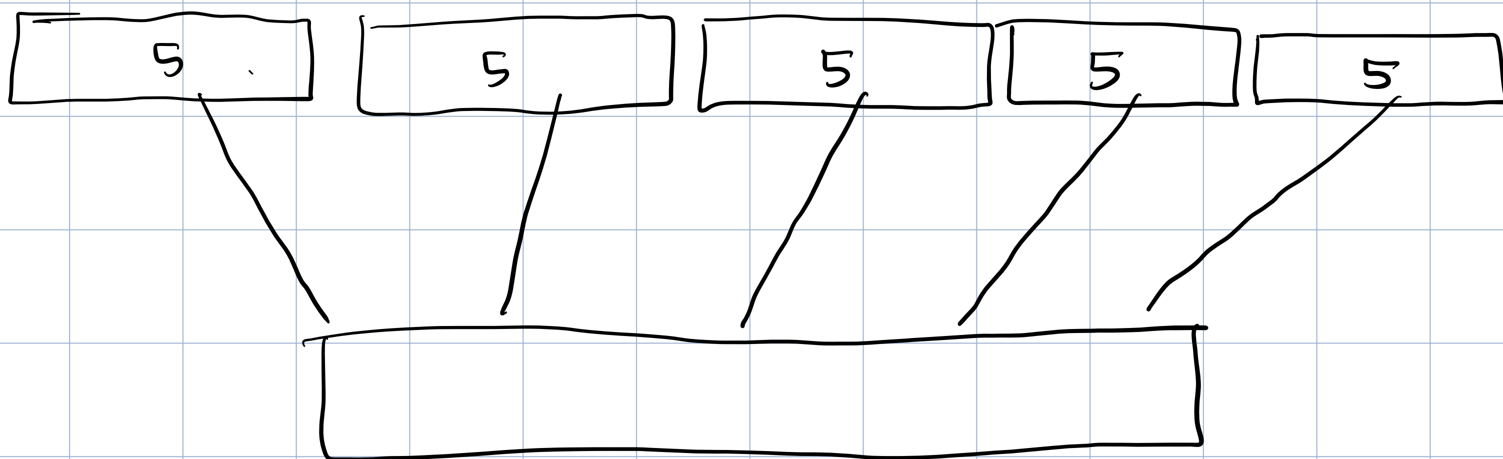
$A = A_L \times 10^3 + A_R$

Find $A_L \cdot B_L$, $A_L \cdot B_R$, $A_R \cdot B_L$, $A_R \cdot B_R$ recursively

$$T(n) = 4\,T\left(\frac{n}{2}\right) + O(n)$$

$n^{\log_2 4} \qquad n$

Problem: There are 25 horses of different speeds, you can race up to 5 of them at a time.
Find the three fastest horses with as few races as possible.



6 ??