COP 3502 10/19/23

---

(0) Tom Post Calc Grade, post curr A, B, C lines (approx)

(1) Exam 1 Solutions posted tomorrow

(2) Leave early?

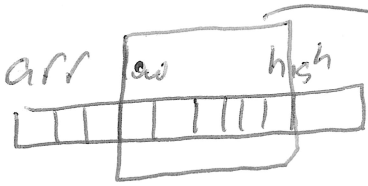
a) Merge Sort

b) Quick Sort

c) Comparison

Last time - finished Merge ~~function~~ algorithm

→ Sort this subarray

void MergeSort( , low, high) {

if (low >= high) return;

int mid = (low + high)/2;

T(n/2) → MergeSort (arr, low, mid);

T(n/2) → MergeSort (arr, mid+1, high);

O(n) → Merge (arr, low, mid, high);

}

~~MS(2,2)~~
~~ms(8,8)~~ ~~MS(1,1)~~
~~ms(7,7)~~ ~~MS(0,0)~~
~~MS(6,6)~~
~~MS(10,10)~~ ~~MS(6,7)~~ ~~MS(0,1)~~
~~MS(9,9)~~ ~~MS(6,8)~~ ~~MS(0,2)~~ ~~MS(3,5)~~
~~MS(9,10)~~ ~~MS(6,10)~~ ~~MS(0,5)~~
MS(0,10)

$$T(n) = T\left(\tfrac{n}{2}\right) + T\left(\tfrac{n}{2}\right) + O(n)$$

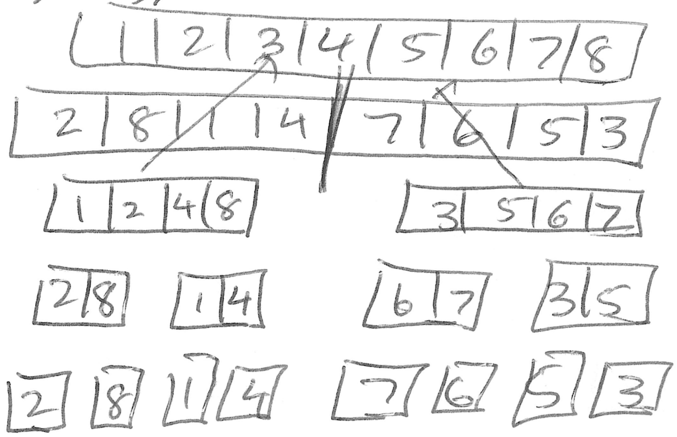
$\downarrow$

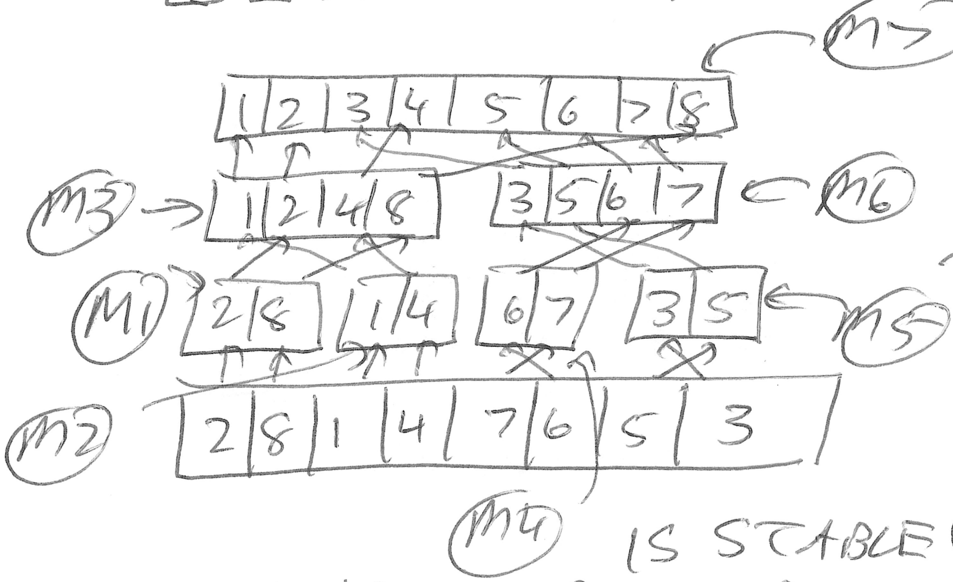
MS on n elements

$$T(n) = 2T\left(\tfrac{n}{2}\right) + O(n)$$

$$A=2,\ B=2,\ k=1 \longrightarrow O(n\lg n)$$

best, avg, worst $\longrightarrow O(n\lg n)$





layers

~ $O(\lg n)$

IS STABLE (keeps ties in original order)

~~IS IN PLACE~~ (~~can make in~~ p

benefit: worst-case $O(n\lg n)$

drawbacks: extra memory (can't do in place)

: ~~NOT IN PLACE~~ ?

Partition function

partition - split up into groups
in math split a set into multiple sets where each element in the original set belongs to exactly 1 ~~set~~ of the new sets

$\{A, B, C, D, E\} \rightarrow \{A, D\}, \{B, C, E\}$

partition element

8 1 6 19 13 15

$\boxed{10}$ 3, ~~18~~, ~~13~~, ~~19~~, ~~6~~, ~~1~~, ~~8~~, 22, 11

i j

while (i < j) {

move i (left ptr) to the right until we find value > a[part]

move j (right ptr) to the left until we find value ≤ a[part]

if (i < j) swap a[i] and a[j]

}

swap a[part] and a[j]

$\underbrace{6, 3, 8, 1,}_{\text{sort}}\ \underbrace{\boxed{10},}_{\text{sorted position}}\ \underbrace{19, 13, 15, 22, 11}_{\text{sort}}$

```
quicksort (int* arr, int low, int high) {
    if (low >= high) return;
    int part = partition (arr, low, high);
    quicksort (arr, low, part-1);
    quicksort (arr, part+1, high);
}
```

where partition elem ended up

Why on earth is quicksort faster if it often splits unevenly???

A: works in place. No extra mallocing or copy back of items

Run-Time Analysis

two even recursive cases

best case $T(n) = 2T(\frac{n}{2}) + O(n) \rightarrow O(n\lg n)$

all items on one side

worst case $T(n) = T(n-1) + O(n) \rightarrow O(n^2)$

Intuitive proof avg case $O(n\lg n)$

1/2 time we split atleast $\frac{1}{4}n$ $\frac{3}{4}n$

$T(n) = T(\frac{3n}{4}) + T(\frac{n}{4}) + O(n)$ ugly

$\leq 2T(\frac{3n}{4}) + O(n)$ → proof $O(n\lg n)$ via iteration

use master