CSE12 - Lecture 13 - B00

Monday, October 24, 2022 9:00 AM

PA4 due tomorrow
PA1 Late/Resubmit → due tomorrow

Sorting Quickly → Quick sort → in-place

```
public class SortQuickly {

  public static void swap(String[] array, int i1, int i2) {
    String temp = array[i1];
    array[i1] = array[i2];
    array[i2] = temp;
  }

  public static int partition(String[] array, int low, int high) {
    int pivotStartIndex = high - 1;
    String pivot = array[pivotStartIndex];
    int smallerBefore = low, largerAfter = high - 2;
    while (smallerBefore <= largerAfter) {
      if (array[smallerBefore].compareTo(pivot) < 0) {
        smallerBefore += 1;
      }
      else {
        swap(array, smallerBefore, largerAfter);
        largerAfter -= 1;
      }
    }
    swap(array, smallerBefore, pivotStartIndex);
    return smallerBefore;
  }

  public static void qsort(String[] array, int low, int high) {
    if (high - low <= 1) { return; }
    int splitAt = partition(array, low, high);
    qsort(array, low, splitAt);
    qsort(array, splitAt + 1, high);
  }

  public static void sortD(String[] array) {
    qsort(array, 0, array.length);
  }

  public static void main(String[] args) {
    String[] str = {"f", "b", "a", "e", "d", "c" };
    int[] result = SortQuickly.sortD(str);
    System.out.println(Arrays.deepToString(result));
  }
}
```

Θ(1) 3 — N 0 N — Θ(N) N-1 — don't stop until sb is larger than la

Draw the picture of sortD()

What is the tight bound of sortD:

Best case: Ω(N * log2(N)) → median partition value at every level

Worst case: O(N²) → sorted array

low = 0
high = 6

pivot Index = 5
pivot = "c"
sb = 0 1 2
la = 4 3 2 1

0	1	2	3	4	5
f	b	a	e	d	c

sb → ↑ la

d b a e f c
↑ ↑

e b a d f c
↑ ↑

0 1 2 3 4 5
a b e d f c
↑ ↑
la sb

a	b	c	d	f	e

← return 2

a	b

return 1

d	f	e

a

d	e	f

d

f

a	b	c	d	e	f

N = 6
height = 3
↳ log2(N)

Ω(N * log2(N))

1
2
3

worst case → sorted array

1	2	3	4	5

1	2	3	4

1	4	3

1	2

1

N = 5
levels = 5
Θ(N²)

median value as pivot

1	2	4	5	3

1	3

3

3	4

[] [] [] []

valid partition

results in partition method

pi

<pv | pv | >pv

return partition index