Divide and Conquer

Maximum in a Circularly Shifted Array

- Sorted array with *n* integers
- Shifted circularly by *k* positions
- Find the maximum integer
- Example:
  - [35, 42, 5, 15, 27, 29] is shifted by 2
  - [87, 89, 91, 93, 99, 63, 68, 71, 73, 81] is shifted by 5

Maximum in a Circularly Shifted Array

- You should think of binary search

Maximum in a Circularly Shifted Array

- Divide the array into left and right half
- In which half is the maximum?

12	16	18	19	24	32	35	49	1	3	4	6	8	11

19	24	32	35	49	1	3	4	6	8	11	12	16	18

Maximum in a Circularly Shifted Array

- Divide the array into left and right half
- In which half is the maximum?

12	16	18	19	24	32	35	49	1	3	4	6	8	11

19	24	32	35	49	1	3	4	6	8	11	12	16	18

- Maximum and minimum are next to each other at the rotation cut
-

Maximum in a Circularly Shifted Array

- Divide the array into left and right half
- In which half is the maximum?

12	16	18	19	24	32	35	49	1	3	4	6	8	11

19	24	32	35	49	1	3	4	6	8	11	12	16	18

- Maximum and minimum are next to each other at the rotation cut
- Maximum is in the half where the leftmost and the rightmost element are not ordered
  - But that is not quite true

Maximum in a Circularly Shifted Array

- Look at this case:

16	18	19	24	32	35	49	1	3	4	6	8	11	12

- Both left and right are well ordered.
- Maximum is at the end of the left array
- We can check this by comparing the last of the left array with the first of the right array

Maximum in a Circularly Shifted Array

- This gives us a recursion

```
left, right = lista[:len(lista)//2], lista[len(lista)//2:]
if left[0]<left[-1] and left[-1]<right[0]:
    return max_circular(right)
else:
    return max_circular(left)
```

- which will fail if either left or right is small

Maximum in a Circularly Shifted Array

- For the base case: left and right need to have each two elements
- list then has to have four elements

```
if len(lista)<4:
    return max(lista)
```

- Is this cheating?
  - Now: calculating the maximum of a list of up to four elements happens in constant time1

Maximum in a Circularly Shifted Array

- Runtime:
  - $T(n) = T(n/2)+$const
  -

Maximum in a Circularly Shifted Array

- Runtime:
  - $T(n) = T(n/2) + \text{const}$
  - Master Theorem:
    - Compare $n^{\log_2(1)}$ with const
    - Case 2: Runtime is $\Theta(\log_2(n))$

Slow Heap Builder

- What is the runtime for the following algorithm?
  - Build a heap out of an array
  - def SlowHeap(a, i,j):
    - if i==j return a[i]
    - Find k such that a[k] is minimum in a[i:j+1]
    - Exchange a[k] and a[i]
    - left, right = a[i+1: mid], a[mid+1:j+1 with mid = $\lfloor \frac{j-i-1}{2} \rfloor + i$
    - SlowHeap(left); SlowHeap(right)

S

Slow Heap Builder

- At each step:
  - Find minimum:
    - Takes time proportional to length of the array
  - Make recursive call

Slow Heap Builder

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

Slow Heap Builder

- Master Theorem:
  - Compare n with $n^{\log_2(2)} = n$
- Case 2:
  - $T(n) = \Theta(n \log_2(n))$