CSCI 241

Scott Wehrwein

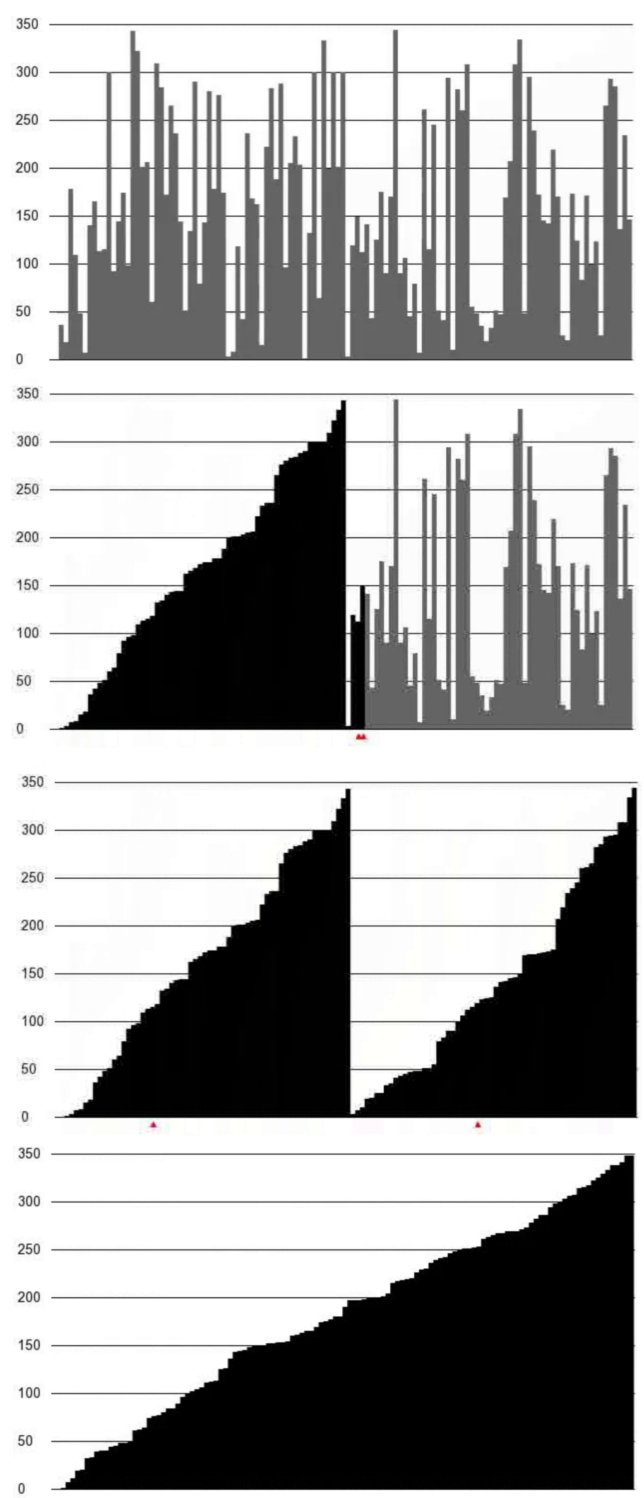
Quick Sort: Runtime

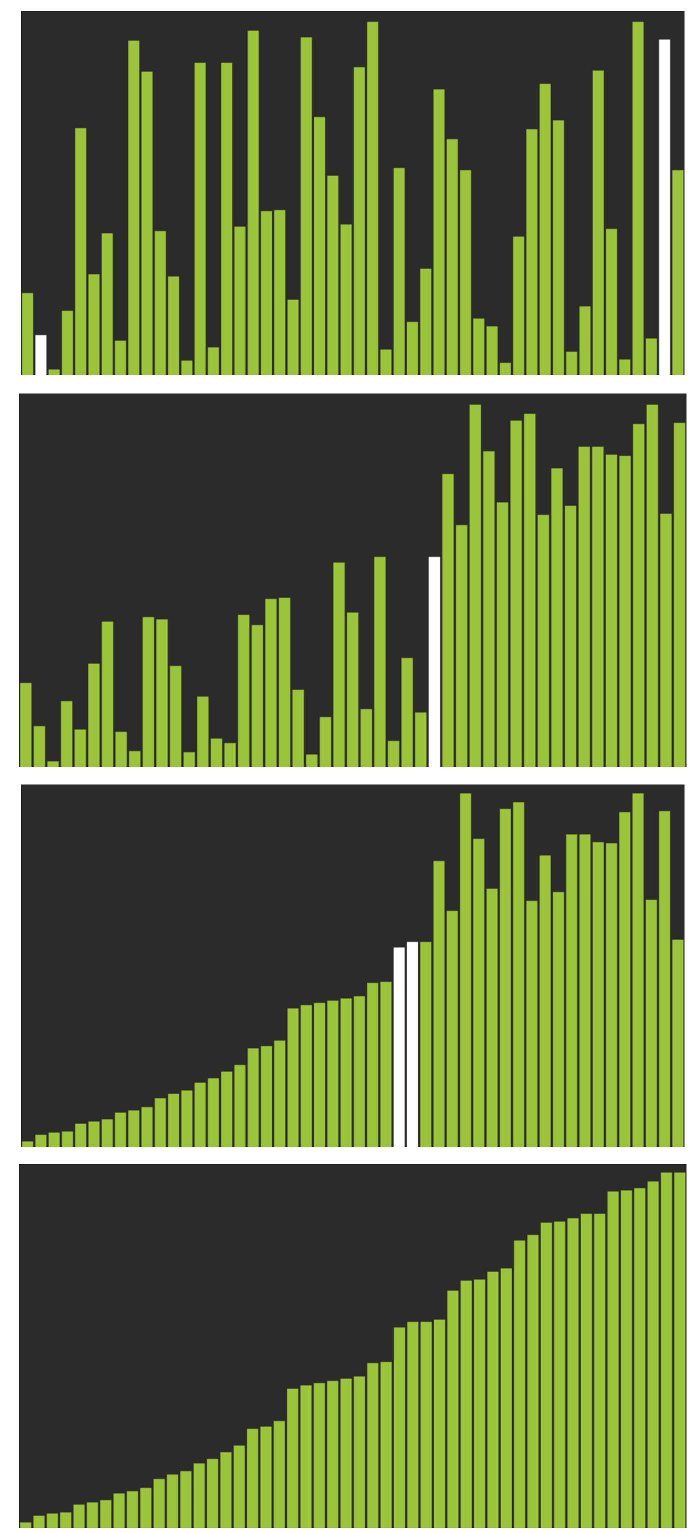
Goals

Understand the best-case and worst-case runtime analysis of quicksort.

Know the average-case runtime of quicksort.

Merge vs Quick





"real work" done here

"real work" done here

Quicksort: Runtime

```
         /** quicksort A[st..end]*/
         quickSort(A, st, end):
O(1)       if (small):
             return

O(??)      mid = partition(A,st,end)

(excluded) quickSort(A,st,mid)
           quickSort(A,mid+1,end)

           # (nothing to do!)
```

Partition: Runtime

O(1)

```
partition(A, start, end)
  initialize i, j
  choose pivot
  swap pivot to A[0]
  while [?] section != []
    # process A[i]:
    if <= p:
      move to <= p section
    else:
      move to > p section
```

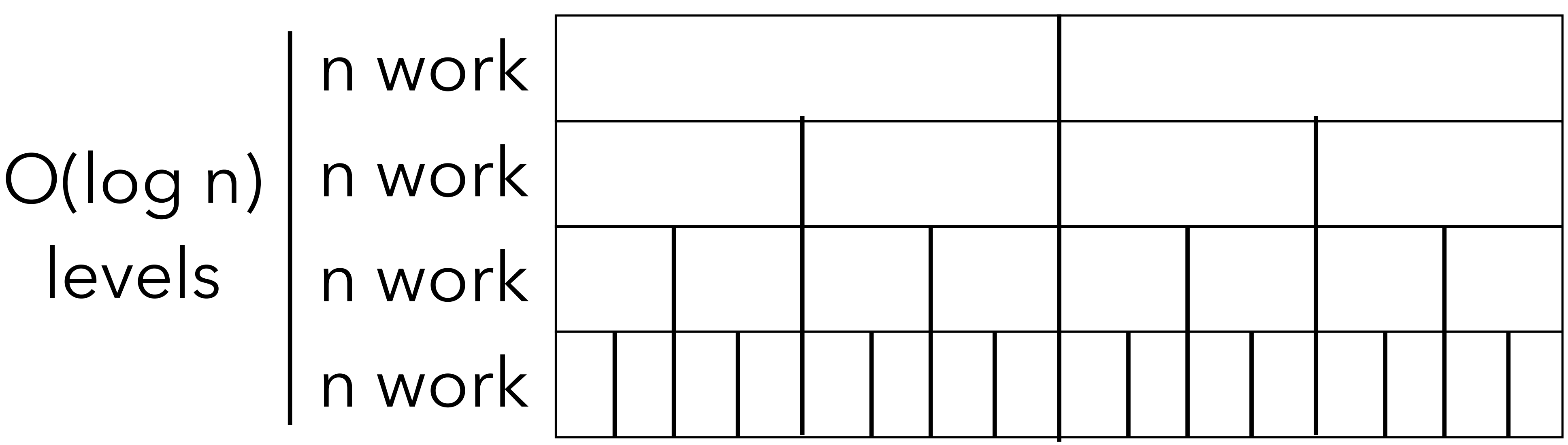
n * O(1)

Total: O(n), where n = end - start.

Runtime: Best case

Best case:

- pivot is the median of the array
- partition splits the array exactly in half
- same analysis as merge sort

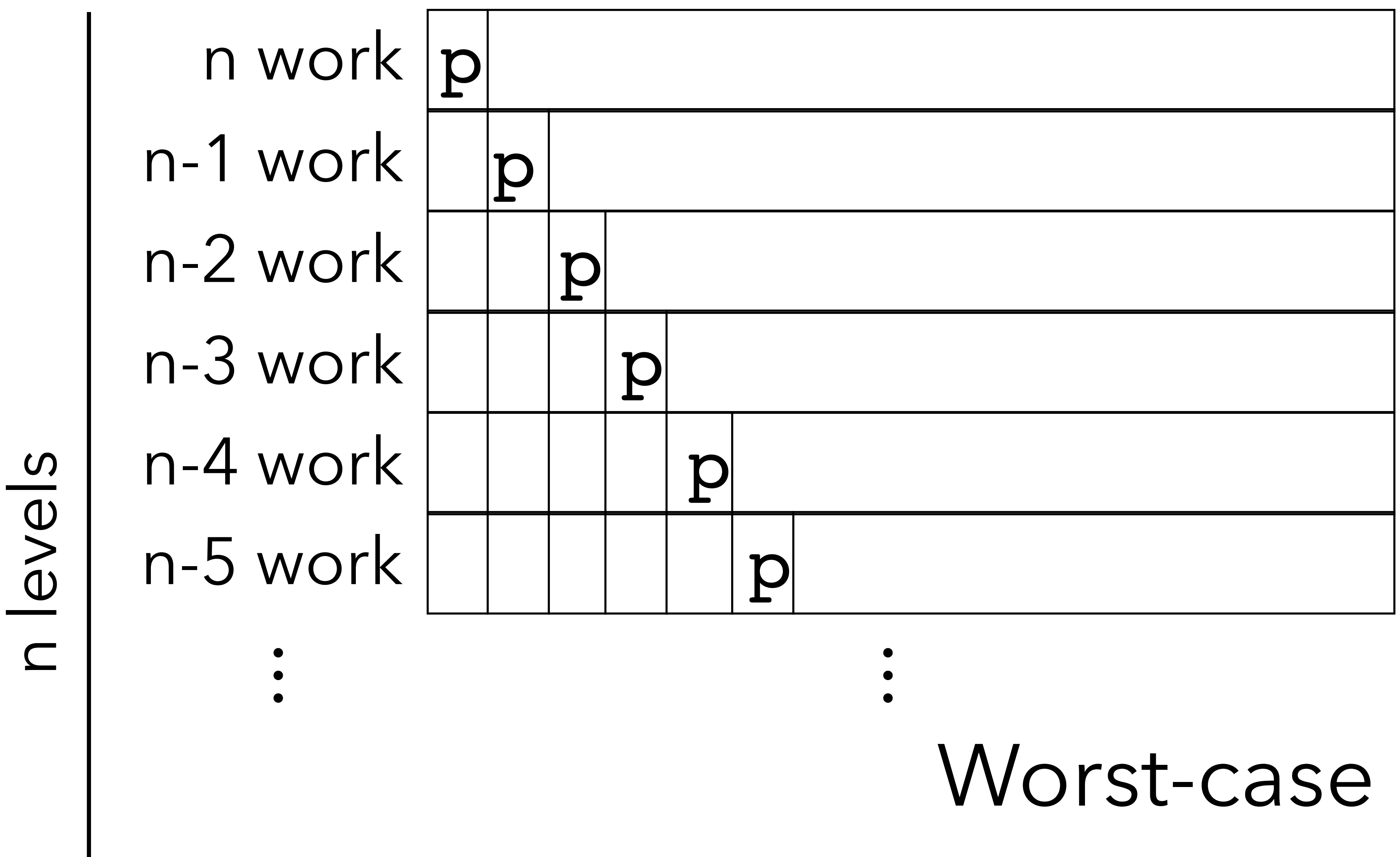


Best-case runtime: O(n log n)

Runtime: Worst case

Worst case:

- pivot is the minimum or maximum of the array
- partition splits the array into 1 and n-1.

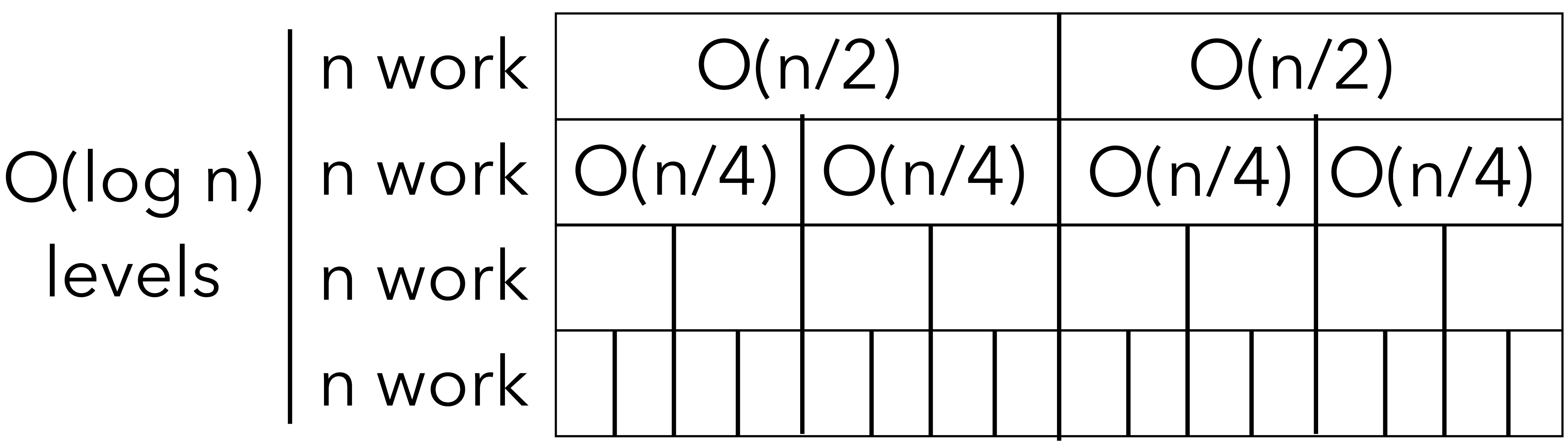


Worst-case runtime: $O(n^2)$

Runtime: Average case

Average case:

- more like best case than worst case (this is rare!)
- full analysis is out of scope, but you should know this result



Average-case runtime: O(n log n)