NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 3, Lecture 6

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Sorting

- Searching for a value
  - Unsorted array — linear scan, $O(n)$
  - Sorted array — binary search, $O(\log n)$
- Other advantages of sorting
  - Finding median value: midpoint of sorted list
  - Checking for duplicates
  - Building a frequency table of values

How to sort?

- You are a Teaching Assistant for a course
- The instructor gives you a stack of exam answer papers with marks, ordered randomly
- Your task is to arrange them in descending order

Strategy 1

- Scan the entire stack and find the paper with minimum marks
- Move this paper to a new stack
- Repeat with remaining papers
  - Each time, add next minimum mark paper on top of new stack
- Eventually, new stack is sorted in descending order

Strategy 1 …

74	32	89	55	21	64

Strategy 1 …

74 32 89 55 21 64

21

Strategy 1 …

74	~~32~~	89	55	~~21~~	64
21	32				

Strategy 1 …

74	~~32~~	89	~~55~~	~~21~~	64
21	32	55			

Strategy 1 …

74	~~32~~	89	~~55~~	~~21~~	~~64~~
21	32	55	64		

Strategy 1 …

~~74~~	~~32~~	89	~~55~~	~~21~~	~~64~~
21	32	55	64	74	

Strategy 1 …

~~74~~	~~32~~	~~89~~	~~55~~	~~21~~	~~64~~
21	32	55	64	74	89

Strategy 1 …

Selection Sort

- Select the next element in sorted order
- Move it into its correct place in the final sorted list

Selection Sort

* Avoid using a second list
    * Swap minimum element with value in first position
    * Swap second minimum element to second position
    * ...

Selection Sort

74	32	89	55	21	64

Selection Sort

74	32	89	55	21	64

Selection Sort

21	32	89	55	74	64

Selection Sort

21	32	89	55	74	64

Selection Sort

21 32 89 55 74 64

Selection Sort

21	32	89	55	74	64

Selection Sort

21	**32**	**55**	**89**	**74**	**64**

Selection Sort

21	32	55	89	74	64

Selection Sort

21 **32** **55** **64** **74** **89**

Selection Sort

21	**32**	**55**	**64**	**74**	**89**

Selection Sort

21	32	55	64	74	89

Selection Sort

21	**32**	**55**	**64**	**74**	**89**

Selection Sort

```
def SelectionSort(l):

  # Scan slices l[0:len(l)], l[1:len(l)], …
  for start in range(len(l)):

    # Find minimum value in slice . . .
    minpos = start
    for i in range(start,len(l)):
      if l[i] < l[minpos]:
        minpos = i

    # . . . and move it to start of slice
    (l[start],l[minpos]) = (l[minpos],l[start])
```

Analysis of Selection Sort

- Finding minimum in unsorted segment of length k requires one scan, k steps
- In each iteration, segment to be scanned reduces by 1
- $T(n) = n + (n-1) + (n-2) + \ldots + 1 = n(n+1)/2 = O(n^2)$