CS 113 – Computer Science I

Lecture 16 – Sorting

Adam Poliak
11/03/2022
Announcements

• Assignment 08
  • Due tonight Thursday 11/10
Guessing game – in class exercise

Pair up:
- Person A chooses a number between 1 and 64
- Person B guesses the number
- Until the guess is correct:
  - Person A tells whether the guess is too high or too low
  - Person B guesses again

After 2 rounds each, choose a number between 1 and 512
Binary Search run time

As the size of our collection increases, the number of guesses/comparisons increases, but not *linearly*

The time increases by $\log n$ (we use base 2)

If our collection contains 8 data points, how many comparisons in worst case do we make:

$$\log_2 8 = 3$$

If our collection contains 512 data points, how many comparisons in worst case do we make:

$$\log_2 512 = 9$$
Sorting
Bubble Sort

Compare two adjacent items, and swap if needed

Repeat until largest item is at the back

Repeat process until done
Bubble Sort

What do we do first?
Bubble Sort

[j - 1]  [j]

0  1  2  3  4  5

10  4  3  0  11  8

Compare j-1 and j; Swap if L[j-1] > L[j]

What next?
**Bubble Sort**

```plaintext
len = 6

0 1 2 3 4 5

| 4 | 10 | 3 | 0 | 11 | 8 |
```

**What next?**

Compare j-1 and j; Swap if L[j-1] > L[j]
### Bubble Sort

```
len = 6
```

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>

**j - 1** | **j**
---|---
1 | 2

**Compare** j-1 and j; **Swap** if L[j-1] > L[j]

What next?
Bubble Sort

```
0  1  2  3  4  5
4  3  10  0  11  8
```

len = 6

```
Compare j-1 and j; Swap if L[j-1] > L[j]
```

What next?
Bubble Sort

```
len = 6
```

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>
```

```
Compare j-1 and j; Swap if L[j-1] > L[j]
```

What next?
Bubble Sort

len = 6

Compare j-1 and j; Swap if L[j-1] > L[j]

What next?
Bubble Sort

len = 6

Compare j-1 and j; Swap if L[j-1] > L[j]

What next?
Bubble Sort

len = 6

```
[ 4  3  0 10 11  8 ]
```

What next?

Compare j-1 and j; Swap if L[j-1] > L[j]
Bubble Sort

len = 6

Compare j-1 and j; Swap if L[j-1] > L[j]

What next?
Bubble Sort

len = 5

Reset and compare pairs with shorter list!

Last element has largest element!

What next?
Bubble Sort

```
0   1   2   3   4   5
3   4   0   10  8   11
```

- Last element has largest element!

```
Compare j-1 and j; Swap if L[j-1] > L[j]
```

len = 5

What next?
Bubble Sort

What next?

Compare j-1 and j; Swap if L[j-1] > L[j]
Bubble Sort

len = 5

```
0 1 2 3 4 5
3 0 4 10 8 11
```

**What next?**

Compare j-1 and j; Swap if L[j-1] > L[j]
Bubble Sort

Compare $j-1$ and $j$; Swap if $L[j-1] > L[j]$

What next?
Bubble Sort

What next?

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>8</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

len = 5

Compare j-1 and j; Swap if L[j-1] > L[j]
Bubble Sort

Compare j-1 and j; Swap if L[j-1] > L[j]

What next?
Bubble Sort

0 1 2 3 4 5

j 0 1

Reset and check pairs with shorter list

len = 4

What next?
Bubble Sort

```
0 1 2 3
0 3 4 8

Compare j-1 and j; Swap if L[j-1] > L[j]
```

What next?
Bubble Sort

len = 4

0 1 2 3 4 5

j - 1 j
1 2

Compare j-1 and j; Swap if L[j-1] > L[j]

What next?
Bubble Sort

```
0  1  2  3  4  5
0  3  4  8  10 11
```

Compare j-1 and j; Swap if L[j-1] > L[j]

What next?
Bubble Sort

Reset; Compare $j-1$ and $j$; Swap if $L[j-1] > L[j]$
Bubble Sort

Reset; Compare j-1 and j; Swap if L[j-1] > L[j]

What next?
Bubble Sort

```
0  1  2  3  4  5
0  3  4  8  10 11
```

Reset; Compare $j-1$ and $j$; Swap if $L[j-1] > L[j]$

What next?
Bubble Sort

Idea: bubble highest values to the end of the list; Check a shorter sublist each time

\[
\text{bubbleSort}(L): \\
\quad \text{for } \text{len} \text{ in range(len}(L), 1, -1): \\
\quad \quad \text{for } j \text{ in range}(1, \text{len}): \# \text{ bubble} \\
\quad \quad \quad \text{if } L[j-1] > L[j]: \\
\quad \quad \quad \quad \text{swap}(j-1, j, L)
\]
Bubble sort

swap(i, j, L):

1. temp = L[i] # step 1
2. L[i] = L[j]    # step 2
3. L[j] = temp # step 3
```python
import swap
import random
import isSorted

def bubbleSort(L):
    """
    Sort the list L in place using bubble sort
    Param L (list): the list to sort
    Return: None
    """
    for end in range(len(L), 1, -1):
        for j in range(1, end):
            if L[j-1] > L[j]:
                swap.swap(j-1, j, L)
                print("swap", j-1, j)

if __name__ == '__main__':
    L = [10, 4, 3, 0, 11, 8]
    print("Before:", L)
    bubbleSort(L)
    print("After:", L, "IsSorted?", isSorted.isSorted(L))
```
Selection sort

Repeatedly find the smallest item and put it at front of list

selectionSort(L):
    for startIdx in range(len(L)):
        minIdx = findMinimum(startIdx, L)
        swap(startIdx, minIdx, L)
Selection Sort

What do we do first?
Selection Sort

Find minimum element idx between start to end

What next?
Selection Sort

Swap the elements at start and minIdx

What next?
Selection Sort

0 4 3 10 11 8

Decrease the interval.

What next?
Selection Sort

Find minimum element between start to end

What next?
Selection Sort

Swap the elements at start and minIdx

What next?
Selection Sort

0 3 4 10 11 8

Decrease the interval.

start
2

What next?
Selection Sort

Find minimum element idx between start to end

What next?
Selection Sort

Swap the elements at start and minIdx

What next?
Selection Sort

Decrease the interval.

What next?
Selection Sort

Find minimum element idx between start to end

What next?
Swap the elements at start and minIdx

What next?
Selection Sort

0  1  2  3  4  5

0  3  4  8  11  10

Decrease the interval.

What next?
Selection Sort

Find minimum element idx between start to end

What next?
Selection Sort

Swap the elements at start and minIdx

What next?
Selection Sort

Decrease the interval.

We’re done!
Selection sort

findMinimum(startIdx, L):

    minIdx = startIdx

    for i in range(startIdx, len(L)):
        if L[i] < L[minIdx]:
            minIdx = i

    return minIdx
Swap

swap(i, j, L):

temp = L[i]  # step 1
L[i] = L[j]  # step 2
L[j] = temp  # step 3
Write a function that swaps two elements

NOTE:
The if statement at the bottom only executes if we run this
file from the command line, e.g.

```python
>> python3 swap.py
Before: [0, 1]
After: [1, 0]
```

This feature allows us to use the functions in this file in other
programs using the syntax

```python
import swap
swap.swap(0,1,L)
```
Write a function that checks if a list is sorted from least to greatest

```python
>>> import random

def isSorted(L):
    """
    Returns True if the list L is sorted; False otherwise.
    Param L (list): the list to test.
    Return (bool)
    """
    for i in range(len(L)-1):
        if L[i] > L[i+1]:
            return False
    return True

if __name__ == '__main__':
    L = list(range(0, 8, 2))
    print(L, "is Sorted?", isSorted(L))
    random.shuffle(L)
    print(L, "is Sorted?", isSorted(L))
    L = [10]
    print(L, "is Sorted?", isSorted(L))
```
Sort a list in place using selection sort. Use your existing implementation for swap and isSorted!

```python
import random

def FindMinIdx(start, l):
    minVal = l[start]
    minIdx = start
    for i in range(start, len(l)):
        if l[i] < minVal:
            minVal = l[i]
            minIdx = i
    return minIdx

def selectionSort(L):
    for start in range(len(L)):
        minVal = L[start]
        minIdx = start
        for i in range(start, len(L)):
            if L[i] < minVal:
                minVal = L[i]
                minIdx = i
        swap(L, minIdx, start)

    if name == 'main':
        L = [10, 4, 3, 0, 11, 8]
        print('Before:', L)
        selectionSort(L)
        print('After:', L, 'IsSorted?', isSorted(L))
```

```python
$ python3 selectionSort.py
Before: [10, 4, 3, 0, 11, 8]
swap 0 3
swap 1 7
swap 2 2
swap 3 5
swap 4 5
swap 5 5
After: [0, 4, 3, 8, 10, 11] IsSorted? True
```
Selection sort and Bubble sort are $O(N^2)$