Data Structures and Algorithms



Sorting

CS51 – Fall 2025

Sorting

- Process of arranging elements in a data structure in non-decreasing order.
- At the heart of many problems, often good first step.
- Exemplary algorithms that we study in CS.

Assumptions for sorting data

- We are working with lists of elements of the same type.
- We have a way of comparing elements with each other.
- We can exchange elements.
- Many different sorting algorithms and most use these assumptions.

Ingredient 1: Compare two elements

- We want to be able to determine whether two elements are equal or which one is "smaller".
- Smaller can mean different things based on the type of data.
  - For example, numbers are sorted in non-decreasing order and strings are sorted in lexicographic order.
- We will assume we work with elements we can compare using the <, >, and == operators.

Ingredient 2: Exchange two elements

- We want to be able to exchange two elements in the list.
- Define a function exchange that takes three parameters, a list and two indices which it uses to exchange the elements in these positions.
- For example, if lst = [-12, 3, 7, -1] calling exchange(lst, 0, 3) would result in
  lst = [-1, 3, 7, -12]
- def exchange(lst, i, j):
  temp = lst[i]
  lst[i] = lst[j]
  lst[j] = temp

Bubble sort

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

- Repeatedly exchange adjacent elements that are out of order.
- At the end of first pass, the maximum element goes to the last index.
- At the end of the second pass, the second maximum element goes to the second to last index.
- After i passes, the largest i elements have been moved to the last i indices.

Bubble sort example – 1st pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
3	1	5	6	2

- i=0
- j=0
- Is lst[0]>lst[1], that is 3>1?
- Yes, then exchange them

0	1	2	3	4
1	**3**	5	6	2

Bubble sort example – 1st pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	5	6	2

- i=0
- j=1
- Is lst[1]>lst[2], that is 3>5?
- No, don't do anything

Bubble sort example – 1st pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	5	6	2

- i=0
- j=2
- Is lst[2]>lst[3], that is 5>6?
- No, don't do anything

Bubble sort example – 1st pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	5	6	2

- i=0
- j=3
- Is lst[3]>lst[4], that is 6>2?
- Yes, then exchange them

0	1	2	3	4
1	3	5	**2**	**6**

Bubble sort example – 1st pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	5	2	6

- End of first pass and the largest element, 6, is at the last index

Bubble sort example – 2nd pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	5	2	6

- i=1
- j=0
- Is lst[0]>lst[1], that is 1>3?
- No, don't do anything

Bubble sort example – 2nd pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	5	2	6

- i=1
- j=1
- Is lst[1]>lst[2], that is 3>5?
- No, don't do anything

Bubble sort example – 2nd pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	5	2	6

- i=1
- j=2
- Is lst[2]>lst[3], that is 5>2?
- Yes, then exchange them

0	1	2	3	4
1	3	**2**	**5**	6

Bubble sort example – 2nd pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	2	5	6

- End of second pass and the second largest element, 5, is at the second to last index

Bubble sort example – 3rd pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	2	5	6

- i=2
- j=0
- Is lst[0]>lst[1], that is 1>3?
- No don't do anything

Bubble sort example – 3rd pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	3	2	5	6

- i=2
- j=1
- Is lst[1]>lst[2], that is 3>2?
- Yes, then exchange them

0	1	2	3	4
1	**2**	**3**	5	6

Bubble sort example – 3rd pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	**2**	**3**	5	6

- End of third pass and the third largest element, 3, is at the third to last index

Bubble sort example – 4th pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	2	3	5	6

- i=3
- j=0
- Is lst[0]>lst[1], that is 1>2?
- No, don't do anything

Bubble sort example – 4th pass

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

0	1	2	3	4
1	2	3	5	6

- End of fourth pass and the fourth largest element, 2, is at the fourth to last index

Bubble sort - optimization

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        exchanged = False
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
                exchanged = True
        if not exchanged:
                    break
```

Practice Time

- Following the same steps we just did, run Bubble sort on the following list [37, 6, 33, 25, 36, 27, 14, 7, 40, 23].

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

Selection sort

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

- Minimizes number of exchanges.
- At every pass, identifies the minimum element and moves it to the left.
- After i passes, the smallest i elements have been moved to the first i indices.

Selection sort example – 1st pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
3	1	5	6	2

- i=0
- j=1
- min_index = 0
- Is lst[1]<lst[0], that is 1<3?
- Yes, min_index = 1

0	1	2	3	4
3	**1**	5	6	2

Selection sort example – 1st pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
3	1	5	6	2

- i=0
- j=2
- min_index = 1
- Is lst[2]<lst[1], that is 5>1?
- No, don't do anything

Selection sort example – 1st pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
3	**1**	5	6	2

- i=0
- j=3
- min_index = 1
- Is lst[3]<lst[1], that is 6<1?
- No, don't do anything

Selection sort example – 1st pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
3	**1**	5	6	2

- i=0
- j=4
- min_index = 1
- Is lst[4]<lst[1], that is 2<1?
- No, don't do anything

Selection sort example – 1st pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	3	5	6	2

- Exchange element at index i=0 with element at min_index=1
- End of first pass and the smallest element, 1, is at the first index

Selection sort example – 2nd pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	**3**	5	6	2

- i=1
- j=2
- min_index = 1
- Is lst[2]<lst[1], that is 5<3?
- No, don't do anything

Selection sort example – 2nd pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	**3**	5	6	2

- i=1
- j=3
- min_index = 1
- Is lst[3]<lst[1], that is 6<3?
- No, don't do anything

Selection sort example – 2nd pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	**3**	5	6	2

- i=1
- j=4
- min_index = 1
- Is lst[4]<lst[1], that is 2<3?
- Yes, min_index = 4

0	1	2	3	4
1	3	5	6	**2**

Selection sort example – 2nd pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	2	5	6	3

- Exchange element at index i=1 with element at min_index=4
- End of second pass and the second smallest element, 2, is at the second index

Selection sort example – 3rd pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	2	**5**	6	3

- i=2
- j=3
- min_index = 2
- Is lst[3]<lst[2], that is 6<5?
- No, don't do anything

Selection sort example – 3rd pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	2	**5**	6	3

- i=2
- j=4
- min_index = 2
- Is lst[4]<lst[2], that is 3<5?
- Yes, min_index = 4

0	1	2	3	4
1	2	5	**6**	**3**

Selection sort example – 3rd pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	2	3	6	**5**

- Exchange element at index i=2 with element at min_index=4
- End of third pass and the third smallest element, 3, is at the third index

Selection sort example – 4th pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	2	3	**6**	5

- i=3
- j=4
- min_index = 3
- Is lst[4]<lst[3], that is 5<6?
- Yes, min_index = 4

0	1	2	3	4
1	2	3	6	**5**

Selection sort example – 4th pass

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

0	1	2	3	4
1	2	3	5	6

- Exchange element at index i=3 with element at min_index=4
- End of third pass and the third smallest element, 3, is at the third index

Practice Time

- Following the same steps we just did, run selection sort on the following list [37, 6, 33, 25, 36, 27, 14, 7, 40, 23].

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

Insertion sort

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```

- Intuitive algorithm that is often the first one that people think of.
- Splits the elements into two groups, a partially sorted on the left and an unsorted on the right. Then it takes an element from the unsorted group and finds its right place in the sorted group.

Insertion sort example – 1st pass

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```

0	1	2	3	4
3	1	5	6	2

- i=1
- j=1
- Is lst[1]<lst[0], that is 1<3?
- Yes, exchange them

0	1	2	3	4
1	3	5	6	2

Insertion sort example – 2nd pass

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```

0	1	2	3	4
1	3	5	6	2

- i=2
- j=2
- Is lst[2]<lst[1], that is 5<3?
- No, break inner for loop

0	1	2	3	4
1	3	5	6	2

Insertion sort example – 3rd pass

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```

0	1	2	3	4
1	3	5	6	2

- i=3
- j=3
- Is lst[3]<lst[2], that is 6<5?
- No, break inner for loop

0	1	2	3	4
1	3	5	6	2

Insertion sort example – 4th pass

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```

0	1	2	3	4
1	3	5	6	2

- i=4
- j=4
- Is lst[4]<lst[3], that is 2<6?
- Yes, exchange them

0	1	2	3	4
1	3	5	2	6

Insertion sort example – 4th pass

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```

0	1	2	3	4
1	3	5	2	6

- i=4
- j=3
- Is lst[3]<lst[2], that is 2<5?
- Yes, exchange them

0	1	2	3	4
1	3	2	5	6

Insertion sort example – 4th pass

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```

0	1	2	3	4
1	3	2	5	6

- i=4
- j=2
- Is lst[2]<lst[1], that is 2<3?
- Yes, exchange them

0	1	2	3	4
1	2	3	5	6

Insertion sort example – 4th pass

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```

0	1	2	3	4
1	2	3	5	6

- i=4
- j=1
- Is lst[1]<lst[0], that is 2<3?
- No, break inner for loop

0	1	2	3	4
1	2	3	5	6

Practice Time

- Following the same steps we just did, run insertion sort on the following list [37, 6, 33, 25, 36, 27, 14, 7, 40, 23].

```
def insertionsort (lst):
    n = len(lst)
    for i in range(1, n):
        for j in range(i, 0, -1):
            if lst[j]<lst[j-1]:
                exchange(lst, j, j-1)
            else:
                break
```