Exercise 6

Bookmark this page

Exercise 6

4 points possible (graded)

ESTIMATED TIME TO COMPLETE: 8 minutes

Here is another version of a sorting function:

```
def mySort(L):
    """ L, list with unique elements """
    clear = False
    while not clear:
        clear = True
        for j in range(1, len(L)):
            if L[j-1] > L[j]:
                clear = False
                temp = L[j]
                L[j] = L[j-1]
                L[j-1] = temp
```

Compare this to:

```
def newSort(L):
    """ L, list with unique elements """
    for i in range(len(L) - 1):
        j=i+1
        while j < len(L):
            if L[i] > L[j]:
                temp = L[i]
                L[i] = L[j]
                L[j] = temp
            j += 1
```

1. Do these two functions result in the same sorted lists?

- Yes
- No

2. Do these two functions execute the same number of assignments of values into entries of the lists?

- Yes. They execute the same number of assignments.
- No. `newSort` may use more - but never fewer - inserts than `mySort`.
- No. `mySort` may use more - but never fewer - inserts than `newSort`.
- No. Either function may use more inserts than the other.

3. Is the worst-case order of growth of these functions the same?

- Yes. `newSort` and `mySort` have the same complexity.
- No. `newSort` has higher complexity than `mySort`.
- No. `mySort` has higher complexity than `newSort`.

4. Do these two functions examine the same number of entries in the list?

- Yes. `newSort` and `mySort` examine the same number of entries.

- No. `newSort` examines more entries than `mySort`.
- No. `mySort` examines more entries than `newSort`.
- No. `mySort` and `newSort` examine different numbers of entries, but one cannot always say which function will examine the most entries.

Submit

Exercise 6

Topic: Lecture 12 / Exercise 6

Show Discussion