Audit Access Expires 5 de ago de 2020
You lose all access to this course, including your progress, on 5 de ago de 2020.

Exercise 5

Finger Exercises due Aug 5, 2020 20:30 -03

Exercise 5

3/3 points (graded)
ESTIMATED TIME TO COMPLETE: 8 minutes

Here is the code for selection sort. For simplicity, assume `L` is a list of integers:

```
def selSort(L):
    for i in range(len(L) - 1):
        minIndx = i
        minVal = L[i]
        j = i+1
        while j < len(L):
            if minVal > L[j]:
                minIndx = j
                minVal = L[j]
            j += 1
        if minIndx != i:
            temp = L[i]
            L[i] = L[minIndx]
            L[minIndx] = temp
```

And here is a suggested alternative:

```
def newSort(L):
    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

✓

Explanation:
Yes, both `selSort` and `newSort` correctly sort a list.
`selSort`:
You can sort a list by always moving the smallest element from the unsorted list to a new list. That procedure would add the elements to the new list in increasing order, and when every element from the old list has been moved over, we end up with a new sorted list. This type of sorting algorithm is often called Selection Sort. `selSort` implements this without explicitly creating a new list, by maintaining sorted (from position `0` to `i-1`) and unsorted (from position `i` to the end) parts of the list. All elements in positions before the iterating variable `i` are sorted, and unsorted for those positions at `i` or below. In each iteration, it selects the smallest element in the unsorted part of the list, and swaps it with the element at the `i` th position. That essentially adds the next smallest element from the old list and appends it to the new. It keeps doing that until the old list is empty (i.e., `i` reaches the end of the list).
`newSort`:
`newSort` is basically a slight variant of Selection Sort. In each iteration, `newSort` also tries to find the smallest element in the unsorted part of the list and appends it to the sorted part of the list. The only difference here is that instead of finding the smallest value in the unsorted part of the list with `minVal` and `minIndx`, `newSort` maintains that the element at the `i` th position is the smallest element between the `i` th and `j` th positions. So, when `j` reaches the end of the list, the `i` th position must have been the smallest element in the unsorted portion (from position `i` to the end) of the list.

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 `selSort`.]
- ○ No. `selSort` may use more - but never fewer - inserts than `newSort`.
- ○ No. Either function may use more inserts than the other.

✓

Explanation:
`selSort` only does one "swap" each iteration of `i`, but `newSort` may use up to (`n-i`) "swaps" for each iteration of `i`.

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

- [✓ Yes. `newSort` and `selSort` have the same complexity.]
- ○ No. `newSort` has higher complexity than `selSort`.
- ○ No. `selSort` has higher complexity than `newSort`.

✓

Explanation:
Yes.
In `selSort`, `i` iterates over each element of the list, and `j` checks between 1 and up to `n-i` elements. That's `n` iterations for `i`, and for each `i`, we are looking for the smallest element by checking about `n/2` elements on average. That's kind of like `n * n/2` checks, which is $O\left(n^2\right)$. `newSort` does the same thing as `selSort`, except that `selSort` just updates temporary variables `minIndx` and `minVal`, but `newSort` updates the `i` th element instead. So, it also ends up with $O\left(n^2\right)$ complexity.

Enviar

Answers are displayed within the problem

Exercise 5

Ocultar discussão

Topic: Lecture 12 / Exercise 5

Add a Post

Show all posts por atividade recente

There are no posts in this topic yet.

✖