Class 25 - Sorting and Searching

Algorithms, Programs, and Problems

What is an *algorithm*?

What is the difference between an *algorithm* and a *program*?

What is the *sorting problem*?

What is a *sorting algorithm*?

Selection Sort

```
def selection_sort(alist):
    for index in range(0, len(alist)):
        maxpos = index
        maxvalue = alist[index]
        for index1 in range(index, len(alist)):
            if (alist[index1] < maxvalue):
                maxpos = index1
                maxvalue = alist[index1]
        temp = alist[index]
        alist[index] = alist[maxpos]
        alist[maxpos] = temp
```

What is the asymptotic running time of `selection_sort`?

Best-First Sort

```
def list_find_best(p):
    """
    Returns the index of the best (according to pick_better)
    element in p.
    """
    if len(p) <= 1:
        return 0
    else:
        bestpos = 1 + list_find_best(p[1:])
        if p[0] <= p[bestpos]:
            return 0
        else:
            return bestpos

def list_sort_insert(cf, p):
    if len(p) == 0:
        return p
    else:
        return list_insert_one(cf, p[0], list_sort_insert(cf, p[1:]))//second change

def list_sort_best_first(p):
    p = list(p)
    if len(p) == 0:
        return p
    else:
        bestpos = list_find_best(p)
        bestel = p.pop(bestpos) # removes p[bestpos] from p
        return [bestel] + list_sort_best_first(p)
```

What is the best case running time for `list_sort_best_first`?

What is the average case running time for `list_sort_best_first`?

What is the worst case running time for `list_sort_best_first`?

Insertion Sort

```
def list_insert_one(el, p):
    if len(p) == 0:
        return [el]
    if el < p[0]:
        return [el] + p
    else:
        return [p[0]] + list_insert_one(el, p[1:])

def list_sort_insert(p):
    if len(p) == 0:
        return p
    else:
        return list_insert_one(p[0], list_sort_insert(p[1:]))
```

What is the best case running time for `list_sort_insert`?

What is the worst case running time for `list_sort_insert`?

Generalizing Comparison

```
def list_insert_one(cf, el, p):
    if len(p) == 0:
        return [el]
    if cf(el, p[0]):
        return [el] + p
    else:
        return [p[0]] + list_insert_one(cf, el, p[1:])

def list_sort_insert(cf, p):
    if len(p) == 0:
        return p
    else:
        return list_insert_one(cf, p[0], list_sort_insert(p[1:]))
```