Homework 5

Instructions: Please submit your solutions via Gradescope by **Friday, 25 February 2022, 10:00am**. Make sure your name, your class group number, and the name of your class teacher is put on **every page** of your submission. Your submission should, ideally, be a **PDF** file.

Exercise 5.1: QuickSort — 3 pts

(a) Give an example showing that the implementation of the QuickSort algorithm discussed in the lecture is not stable.

(b) Suppose you can use additional space. Explain how the Partition procedure can be modified so that QuickSort becomes stable.

(c) Implement the proposed Partition procedure in Python.

(d) One disadvantage of the (non-randomized) QuickSort algorithm discussed in the lecture is that it takes worst-case time $\Theta(n^2)$, even on inputs that are already sorted. One attempt to fix this is the "median-of-three rule." This rule looks at the first, the middle, and the last entry and takes the median of the three as a pivot. Implement this modified partition procedure in Python.

Exercise 5.2: Decision tree for QuickSort — 4 pts

Consider the variant of QuickSort that we discussed in the lecture, in which the Partition procedure picks the last element as pivot.

(a) Draw the decision tree for inputs A of length $|A| = 3$.

(b) Over all inputs A of length $|A| = 3$, what is the maximum number of comparisons QuickSort needs to carry out to determine the output?

(c) Give an example input that achieves that maximum number.

(d) How many reachable leaves are in the decision tree for QuickSort on inputs A of length $|A| = 3$?

(e) For each reachable leaf, give an example input that leads to that leaf.

Exercise 5.3: BucketSort — 3 pts

In the lecture, we discussed BucketSort, which sorts n numbers drawn from the Uniform distribution on $[0, 1)$, with expected running time $O(n)$.

(a) Explain why the worst-case running time for BucketSort is $\Theta(n^2)$. What simple change to the algorithm preserves its linear average-case running time and makes its worst-case running time $O(n \log n)$?

(b) We are given n points $p_i = (x_i, y_i)$, for $i = 1, \ldots, n$, in the unit circle such that $0 < x_i^2 + y_i^2 \leq 1$. Suppose that the points are uniformly distributed: that is, the probability of finding a point in any region of the circle is proportional to the area of that region. Describe and analyse an algorithm with an expected running time of $O(n)$ to sort the n points by their distances $d_i = \sqrt{x_i^2 + y_i^2}$ from the origin.

Hint: Design the bucket sizes in BucketSort to reflect the uniform distribution of the points in the unit circle.

Exercise 5.1: QuickSort **3 pts**

(a) Give an example showing that the implementation of the QuickSort algorithm discussed in the lecture is not stable.

(a) Sorting $3^1\ 3^2\ 1$

After partition $1\ 3^2\ 3^1$

↑ sort this gives $3^2\ 3^1$

So $1\ 3^2\ 3^1$. Hence not stable.

(b) Suppose you can use additional space. Explain how the Partition procedure can be modified so that QuickSort becomes stable.

(c) Implement the proposed Partition procedure in Python.

Wasn't sure about this.

(d) One disadvantage of the (non-randomized) QuickSort algorithm discussed in the lecture is that it takes worst-case time $\Theta(n^2)$, even on inputs that are already sorted. One attempt to fix this is the "median-of-three rule." This rule looks at the first, the middle, and the last entry and takes the median of the three as a pivot. Implement this modified partition procedure in Python.

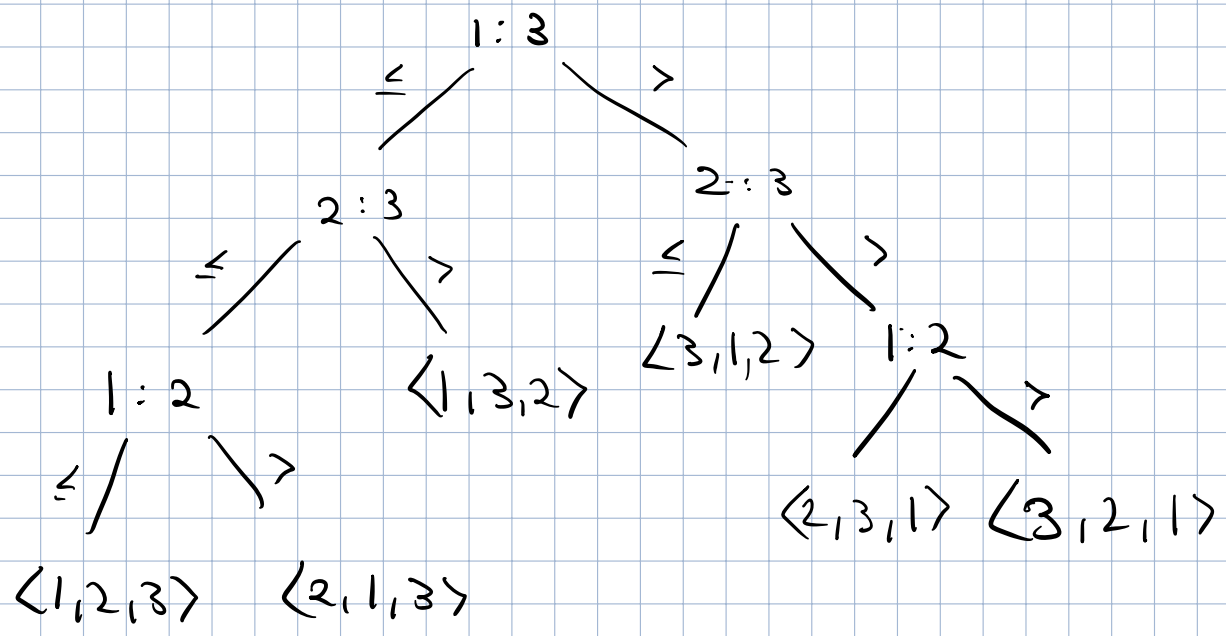
```python
def medianPartition(A, start, end):
    # find median
    mid = (start + end) // 2
    if A[start] <= A[mid] <= A[end] or A[start] >= A[mid] >= A[end]:
        A[end], A[mid] = A[mid], A[end]
    elif A[mid] <= A[start] <= A[end] or A[mid] >= A[start] >= A[end]:
        A[start], A[mid] = A[start], A[end]
    # else end is median

    i = start - 1
    for j in range(start, end):
        if A[j] <= A[end]:
            i += 1
            A[i], A[j] = A[j], A[i]
    A[end], A[i+1] = A[i+1], A[end]
    return i+1
```

Exercise 5.2: Decision tree for QuickSort **4 pts**

Consider the variant of QuickSort that we discussed in the lecture, in which the Partition procedure picks the last element as pivot.

(a) Draw the decision tree for inputs A of length $|A| = 3$.

(a)



(b) Over all inputs A of length $|A| = 3$, what is the maximum number of comparisons QuickSort needs to carry out to determine the output?

(b) 3

(c) Give an example input that achieves that maximum number.

(c) ⟨1,2,3⟩

(d) How many reachable leaves are in the decision tree for QuickSort on inputs A of length $|A| = 3$?

6

(e) For each reachable leaf, give an example input that leads to that leaf.

See above

Exercise 5.3: BucketSort **3 pts**

In the lecture, we discussed BucketSort, which sorts n numbers drawn from the Uniform distribution on $[0, 1)$, with expected running time $O(n)$.

(a) Explain why the worst-case running time for BucketSort is $\Theta(n^2)$. What simple change to the algorithm preserves its linear average-case running time and makes its worst-case running time $O(n \log n)$?

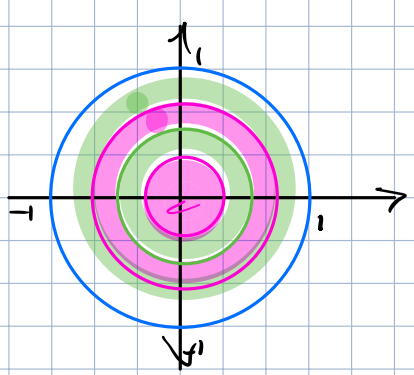
(a) In the worst case, all the inputs land in the same bucket. Thus using insertion sort to sort n items requires $\Theta(n^2)$ worst case running time.

Method to improve worst case

▷ Sort each bucket using an $O(n \lg n)$ algorithm like randomised quicksort or heapsort.

(b) We are given n points $p_i = (x_i, y_i)$, for $i = 1, \ldots, n$, in the unit circle such that $0 < x_i^2 + y_i^2 \leq 1$. Suppose that the points are uniformly distributed: that is, the probability of finding a point in any region of the circle is proportional to the area of that region. Describe and analyse an algorithm with an expected running time of $O(n)$ to sort the n points by their distances $d_i = \sqrt{x_i^2 + y_i^2}$ from the origin.

(b) Since the probability of any region is propotional to its area, we want the buckets to have equal area. One aproach is using buckets as rings.



As the total area is $\pi(1)^2 = \pi$, we want each ring to be $\frac{\pi}{n}$ in area. Then we can sort each ring by distance and then join the buckets to get sorted list.

For first ring: $\pi r^2 = \frac{\pi}{n} \Rightarrow r^2 = \frac{1}{n}$ so

$$x_i^2 + y_i^2 \leq \frac{1}{n}$$

For second ring $\pi r^2 = \frac{2\pi}{n} \Rightarrow r^2 = \frac{2}{n}$ $\frac{1}{n} < x_i^2 + y_i^2 \leq \frac{2}{n}$

⋮

For nth ring $\pi r^2 = \frac{n\pi}{n} \Rightarrow r^2 = 1$ so $\frac{n-1}{n} < x_i^2 + y_i^2 \leq \frac{n}{n} = 1$

Splitting the points into rings takes $O(n)$ time.
Then we can sort each ring using insertion sort.
As shown in the lectures this takes $O(n)$ time expected as our points are i.i.d. And then simply join the buckets together which also takes $O(n)$ time.
So overall $O(n) + O(n) + O(n) = O(n)$ time.