HW2

Chris Marcon

1 (a) $E = \frac{1}{n}\sum_{i=1}^{n} |\pi_i - i|$, $Pr(\pi_i = k)$ where k is a number in the set π is $\frac{1}{n}$, hence:

$E = \frac{1}{n}\sum_{i=1}^{n} |B_i|$ where B is a random variable and $0 \le B < n$. such that:

$E = \frac{1}{n}\sum_{i=1}^{n} i = \frac{n}{2}\left(\frac{n+1}{n}\right) \Rightarrow$ $\boxed{E = \frac{n+1}{2}}$

1) (b) $f(i) = \sum_{i=1}^{n} |\pi_i - i|$ $\quad Pr(f(i) = 0) = Pr(f(i) = max)$

$$Pr(f(i) = 0) = (Pr(\pi_i = i))^n = \left(\frac{1}{n}\right)^n = \frac{1}{n^n}$$

$Pr(f(i) = k)$ where $0 < k < max$

Probability that π is NOT sorted in decreasing or increasing order (every "randomizable" case) $= 1 - \frac{2}{n^n}$

hence if we take every possible case of $f(i)$, and multiply by $\frac{1}{n}$, we get:

$\frac{1}{n}\sum_{i=1}^{n} |\pi_i - i|$ which gives the average distance each element must move to sort an array.

1) (c) For an algorithm that only does adjacent swaps, that would mean each element takes k swaps where $0 \le k \le$ length of array.

~~This gives a best case time com~~

This gives the following time complexities:

Best case: $\Omega(n)$

Avg. case: $\Theta(n^2)$

WORST CASE: O(N^2)

6.4.3 (#2)

Based on the video by Reingold, max-heapify takes $O(\log n)$ time, but to sort in either incr. or decr. order, Heapsort takes $O(n \lg(n))$ time since the nth element must be added and recursively sorted.

③ (a) Base case: use list A of 1 element.
$p = r$, so function does not loop.
~~for a 2-element list, A, sorts each element and returns.~~ breaks

Induction:
for a list of n-elements,
line 3 partitions left subarray into a size smaller than the array and recursively sorts.
line 5 will update $p = q+1$ and will recursively quicksort at position $q+1$, hence, sorting the array slice $A[q+1 : n]$.

(b) for array of n-elements, the stack depth is $\Theta(n)$ if the list is always sorted because there are $n-1$ recursions and the loop is broken at call n.

(c)
```
def Quicksort(A, p, r):
    while (p < r):
        q = PARTITION(A, p, r)
        if (q < ((r-p)/2)):
            Quicksort(A, p, q-1)
            p = q+1
        else:
            Quicksort(A, q+1, r)
            r = q-1
    return
```
← Python 3.6

(d) For unmodified Tail-Recursive Quicksort, the average stack depth is $\Theta(\lg n)$ since a conditional recursive call is made where $p < r$ and the array is partitioned in a presorted subarray

Source: BigOcheatsheet.com

in Recitation (26-1-18)

4) (a) Based on the TA Explanation, a bucket sort that separates the integer array by their number of digits would give a time complexity of $O(n)$.
Then, a radix sort could be used within each subset of the array to give an average time complexity of $\Theta(kn)$ within each bucket where the items are sorted from least significant to most significant digit (taking a counting sort as a subroutine).
Finally, join all sublists in order to complete the sort.
This gives an overall time complexity of $\Theta(kn+c) \cong O(n)$