2nd Mid-Semester Exam Solutions

Question 1 (Each part has 5 marks)

The function `check1`(a) returns `True` if a is equal to 1 and `False` otherwise.

define `find1`$(\underline{a})$
  Set n to be the length of $\underline{a}$.
  for i *in the range* $[1, n]$
    if `check1`(a_i)
      return `True`
  return `False`

The possible input tuples $\underline{a}$ consist of the numbers $1, \ldots, n$ in different permutations.

1. What is the *largest* number of times (as a function of n) that the algorithm will call `check1`. (*Worst case.*)
2. What is the *smallest* number of times (as a function of n) that the algorithm will call `check1`. (*Best case.*)
3. What is the *average* number of times (as a function of n) that the algorithm will call `check1`. (*Average case*)

Note: Remember that once **return** is executed, the algorithm stops.

Solution

Note that the algorithm returns after n calls to `check1` if $a_1, \ldots, a_{n-1}$ are different from 1. In other words, it returns after n calls to `check1` if $a_n = 1$.

Similarly, for each k in $[1, n]$, the algorithm returns after k calls to `check1` if $a_k = 1$.

Part (1). As already seen, if $a_n = 1$, we see that the algorithm returns after n call to `check1` which is the *maximum* number of calls.

Part (2). The least number of calls is 1 which is the case when $a_1 = 1$.

Part (3). Fix k in $[1, n]$. Among all possible permutations of $[1, n]$ we see that *exactly* $(n-1)!$ of them have $a_k = 1$. Thus, the fraction of possible inputs which give rise to k calls to `check1` is $1/n$. Thus, the average number of calls to `check1` is $(1/n)\sum_{k=1}^{n} k = (n+1)/2$.

Question 2 (5 marks)

The following randomized algorithm uses the function `rand`(n) which returns an random element of $[1, n]$ where each element is equality likely to be chosen. The function `cmp`(a, b) returns `True` if $a \le b$ and `False` otherwise.

```
define better(a)
    Set n to be the length of a.
    Set f to be 0.
    while f is less than n/2
        Set f to be 0.
        Set k to be the output of rand(n).
        for i in the range [1, n].
            if cmp(a_i, a_k)
                Increment f.
    return k
```

When the input $\underline{a} = (1, \ldots, n)$, what is the *expected* number of times (as a function of n) that the algorithm will call `cmp`.

Solution

We see that each time the algorithm enters the **while** loop, it makes n calls to `cmp`. Thus, if we enter this loop r times, the number of calls is rn.

Next, we see that at the end of the **while** loop, we have $f = k$. So it re-enters the loop if and only if $k < n/2$.

Thus, if $k_1, \ldots, k_r$ are the values returned by `rand`(n) in successive calls *and* $k_i < n/2$ then the algorithm enters the while loop at least r times. Moreover, it does not enter again if $k_r \geq n/2$. The probability of this is

$$P[k < n/2]^{r-1} P[k \geq n/2] = \begin{cases} \frac{(m-1)^{r-1}(m+1)}{(2m)^r} & n = 2m \\ \frac{m^{r-1}(m+1)}{(2m+1)^r} & n = 2m+1 \end{cases}$$

Thus, the expected number of calls to `cmp` are

$$n \cdot \sum_{r=1}^{\infty} r \cdot \frac{(m-1)^{r-1}(m+1)}{(2m)^r}$$

when $n = 2m$, and

$$n \cdot \sum_{r=1}^{\infty} r \cdot \frac{m^{r-1}(m+1)}{(2m+1)^r}$$

when $n = 2m + 1$.

We can calculate this to be

$$\begin{cases} \frac{(2m)^2}{m+1} & n = 2m \\ \frac{(2m+1)^2}{m+1} & n = 2m+1 \end{cases}$$

When n is large this is approximately $2n$ which is what we get if we put the probability $P[k < n/2] = 1/2$ for all n (which is approximately correct for n large).