Homework #2

(Due: Nov 8)

> **Input:** An array $A[1:n]$ of n distinct numbers.
>
> **Output:** Numbers of $A[1:n]$ rearranged in increasing order of value.
>
> **Steps:**
>
> 1. **Pivot Selection:** Select pivot $x = A[1]$.
> 2. **Partition:** Use a stable partitioning algorithm to rearrange the numbers of $A[1:n]$ such that $A[k] = x$ for some $k \in [1, n]$, each number in $A[1:k-1]$ is smaller than x, and each in $A[k+1:n]$ is larger than x.
> 3. **Recursion:** Recursively sort $A[1:k-1]$ and $A[k+1:n]$.
> 4. **Output:** Output $A[1:n]$.

Figure 1: [Task 1] The deterministic Quicksort algorithm we analyzed in the class.

Task 1. [100 Points] The Variance of the #Comparisons Performed by Quicksort

[*Do not panic. This task is not as scary as it seems. A lot of work has already been done for you. The amount of work you will need to do for each part is often quite small and straightforward.*]

This task asks you to precisely compute the variance of the number of element comparisons performed by the Quicksort algorithm shown in Figure 1.

Let t_n be the number of comparisons performed by our Quicksort algorithm averaged over all $n!$ permutations of an input of size n, and let v_n be its variance.

Let $f_{n,k}$ be the fraction of all possible inputs of size n for which the algorithm performs exactly k comparisons. Then by definitions of mean and variance,

$$t_n = \sum_k k f_{n,k} \quad \text{and} \quad v_n = \sum_k k^2 f_{n,k} - t_n^2$$

(*a*) [**5 Points**] Consider the following generating function for $f_{n,k}$'s.

$$F_n(z) = f_{n,0} + f_{n,1}z + f_{n,2}z^2 + \ldots + f_{n,k}z^k + \ldots$$

Show that $t_n = F_n'(1)$ and $v_n = F_n''(1) + F_n'(1) - (F_n'(1))^2$.

(*b*) [**10 Points**] Argue that $F_n(z)$ can be described by the following recurrence relation:

$$F_n(z) = \begin{cases} 1, & \text{if } n \le 1, \\ \frac{z^{n-1}}{n} \sum_{k=1}^{n} F_{k-1}(z)F_{n-k}(z), & \text{otherwise.} \end{cases}$$

(*c*) [**10 Points**] Using parts (*a*) and (*b*) derive the following recurrence relation for t_n:

$$t_n = \begin{cases} 0, & \text{if } n \leq 1, \\ n - 1 + \frac{1}{n}\sum_{k=1}^{n}(t_{k-1} + t_{n-k}), & \text{otherwise.} \end{cases}$$

Recall that we already solved this recurrence in the class to show that $t_n = 2(n+1)H_n - 4n$. You do not need to solve it here.

(*d*) [**10 Points**] Let $s_n = F_n''(1)$. Show that $s_n = 0$ for $n \leq 2$, and the following recurrence holds for $n > 2$:

$$s_n = (n-1)(n-2) + \frac{1}{n}\left(\sum_{k=1}^{n}(s_{k-1} + s_{n-k}) + 2(n-1)\sum_{k=1}^{n}(t_{k-1} + t_{n-k}) + 2\sum_{k=1}^{n} t_{k-1}t_{n-k}\right)$$

(*e*) [**5 Points**] Show that the recurrence for s_n from part (*d*) can be simplified to:

$$s_n = \begin{cases} 0, & \text{if } n \leq 2, \\ \frac{2}{n}\sum_{k=0}^{n-1} s_k + \frac{2}{n}\sum_{k=1}^{n} t_{k-1}t_{n-k} + (n-1)(2t_n - n), & \text{otherwise.} \end{cases}$$

(*f*) [**25 Points**] Using $t_n = 2(n+1)H_n - 4n$ (proved in the class) and the definitions and mathematical identities involving harmonic numbers given in Table 1, show that the recurrence from part (*e*) can be written as:

$$s_n = \begin{cases} 0, & \text{if } n \leq 2, \\ \frac{2}{n}\sum_{k=0}^{n-1} s_k + \frac{4}{3}(n+1)(n+2)\left((H_n)^2 - H_n^{(2)}\right) & \\ -\frac{4}{9}(8n^2 + 21n + 7)H_n + \frac{1}{27}(95n^2 + 309n + 28), & \text{otherwise.} \end{cases}$$

(*g*) [**10 Points**] Using part (*f*) show that for $n \geq 0$:

$$\frac{s_{n+1}}{n+2} = \frac{s_n}{n+1} + 4\left((H_n)^2 - H_n^{(2)}\right) - \frac{8n}{n+1}H_n + 7 - \frac{10}{n+1} + \frac{6}{n+2}$$

(*h*) [**20 Points**] Solve the recurrence from part (*g*) to show that for $n \geq 0$:

$$s_n = 4(n+1)^2\left((H_n)^2 - H_n^{(2)}\right) - 4(n+1)(4n+1)H_n + n(23n+17)$$

Use of generating functions is optional for this part.

(*i*) [**5 Points**] Finally, combine your result from part (*h*) with the solution for t_n we proved in the class to show that for $n \geq 0$:

$$v_n = 7n^2 + 13n - 2(n+1)H_n - 4(n+1)^2H_n^{(2)}$$

$$H_n = \sum_{k=1}^{n} \frac{1}{k} \tag{1}$$

$$\lim_{n\to\infty} (H_n - \ln n) = \gamma \approx 0.5772156649 \tag{2}$$

$$H_n^{(2)} = \sum_{k=1}^{n} \frac{1}{k^2} \tag{3}$$

$$\lim_{n\to\infty} H_n^{(2)} = \frac{\pi^2}{6} \approx 1.644934068 \tag{4}$$

$$(H_{n+1})^2 - H_{n+1}^{(2)} = (H_n)^2 - H_n^{(2)} + \frac{2H_n}{n+1} \tag{5}$$

$$\sum_{k=1}^{n} H_{k-1} = n(H_n - 1) \tag{6}$$

$$\sum_{k=1}^{n} H_{n-k} = n(H_n - 1) \tag{7}$$

$$\sum_{k=1}^{n} H_{k-1}H_{n-k} = n\left((H_n)^2 - H_n^{(2)} - 2(H_n - 1)\right) \tag{8}$$

$$\sum_{k=1}^{n} kH_{k-1} = \frac{n(n+1)}{2}\left(H_n - \frac{1}{2} - \frac{1}{n+1}\right) \tag{9}$$

$$\sum_{k=1}^{n} kH_{n-k} = \frac{n(n+1)}{2}\left(H_n - \frac{3}{2} + \frac{1}{n+1}\right) \tag{10}$$

$$\sum_{k=1}^{n} kH_{k-1}H_{n-k} = \frac{n(n+1)}{2}\left((H_n)^2 - H_n^{(2)} - 2(H_n - 1)\right) \tag{11}$$

$$\sum_{k=1}^{n} k^2H_{k-1} = \frac{n(n+1)(2n+1)}{6}H_n - \frac{n}{36}(4n^2 + 15n + 17) \tag{12}$$

$$\sum_{k=1}^{n} k^2H_{n-k} = \frac{n(n+1)(2n+1)}{6}H_n - \frac{n}{36}(22n^2 + 15n - 1) \tag{13}$$

$$\sum_{k=1}^{n} k^2H_{k-1}H_{n-k} = \frac{n(n+1)(2n+1)}{6}\left((H_n)^2 - H_n^{(2)}\right) - \frac{n}{18}(13n^2 + 15n + 8)H_n + \frac{n}{108}(71n^2 + 111n + 34) \tag{14}$$

$$t_n = 2(n+1)H_n - 4n \tag{15}$$

Table 1: [Task 1] Definitions and mathematical identities useful for Task 1.

SELECT($A[q:r],\ k,\ \alpha,\ s_1,\ s_2,\ b$)

Input: An array of distinct elements, and an integer $k \in [1, r-q+1]$. The parameter $\alpha \in [0,1]$ is a floating point number that gives the probability of choosing s_1 as the block size to be used at this level of recursion and $1-\alpha$ is the probability of choosing s_2. Also b is an upper bound on the size of the base case.

Output: An element x of $A[q:r]$ such that $rank(x, A[q,r]) = k$.

$n \leftarrow r - q + 1$
if $n \leq b$ **then**
    sort $A[q:r]$
    return $A[q+k-1]$
else
    $d \leftarrow$ a floating point number between 0 and 1 (inclusive) chosen uniformly at random
    if $d \leq \alpha$ **then** $s \leftarrow s_1$
    else $s \leftarrow s_2$
    divide $A[q:r]$ into blocks B_i's each containing s consecutive elements
    (last block may contain fewer than s elements)
    for $i \leftarrow 1$ **to** $\lceil \frac{n}{s} \rceil$ **do**
        $M[i] \leftarrow$ median of B_i using sorting
    $x \leftarrow$ SELECT $\left(M\left[1 : \lceil \frac{n}{s} \rceil\right],\ \left\lfloor \frac{\lceil \frac{n}{s} \rceil + 1}{2} \right\rfloor,\ \alpha,\ s_1,\ s_2,\ b \right)$ {*median of medians*}
    $t \leftarrow$ PARTITION($A[q:r],\ x$)   {*partition around x which ends up at $A[t]$*}
    if $k = t - q + 1$ **then return** $A[t]$
    else if $k < t - q + 1$ **then return** SELECT($A[q:t-1],\ k,\ \alpha,\ s_1,\ s_2,\ b$)
        else return SELECT($A[t+1:r],\ k - t + q - 1,\ \alpha,\ s_1,\ s_2,\ b$)

Figure 2: [Task 2] Selection with probabilistic blocking.

Task 2. [50 Points] Recursive Selection with Probabilistic Blocking

Figure 2 shows a slightly generalized version of the selection algorithm we saw in the class. Instead of using a single block size (e.g., 5) at all levels of recursion, it chooses between two block sizes s_1 and s_2 with probability α and $1-\alpha$, respectively. The base case size b is also a parameter to the algorithm. Observe that when $b = 140$ and $s_1 = s_2 = 5$ (or $s_1 = 5$ with $\alpha = 1$, or $s_2 = 5$ with $\alpha = 0$), the algorithm reduces to the one we saw in the class.

(*a*) [**15 Points**] Write a recurrence relation describing the running time of SELECT on an array of size n assuming $s_1 = s_2 = 3$. Using the approach we saw in the class can you reduce the running time to $\mathcal{O}(n)$ based on your recurrence? Why or why not?

(*b*) [**20 Points**] How about calling SELECT with $s_1 = 3$, $s_2 = 5$, and $\alpha = \frac{1}{3}$? Can you get an $\mathcal{O}(n)$ upper bound based on your recurrence from part (*a*)? Explain. If so, what is the smallest value of b you can use?

(*c*) [**15 Points**] Now, how about calling SELECT with $s_1 = 3$ and $s_2 = 5$, but an arbitrary value of $\alpha < 1$? Can you still get down to $\mathcal{O}(n)$? Explain.

```
Comp-Q( n )        {Integer n ≥ 0}

if n = 0 then return 1
else
    q ← Comp-Q ( n − 1 )
    r ← Comp-R ( n − 1 )
    s ← Comp-S ( n − 1 )
    t ← Comp-S ( n )
    return q + 4r + s − t
```

```
Comp-R( n )        {Integer n ≥ 0}

if n = 0 then return 1
else
    q ← Comp-Q ( n − 1 )
    r ← Comp-R ( n − 1 )
    return 3q + r
```

```
Comp-S( n )        {Integer n ≥ 0}

if n = 0 then return 1
else
    q ← Comp-Q ( n − 1 )
    r ← Comp-R ( n − 1 )
    s ← Comp-S ( n − 1 )
    return 2q + 3r + s
```

Figure 3: [Task 3] Three mutually recursive functions.

Task 3. [50 Points] Three Mutually Recursive Functions

Figure 3 shows three mutually recursive functions Comp-Q, Comp-R and Comp-S. Each function accepts a nonnegative integer as the sole input. Now answer the following questions.

(a) [**20 Points**] Use generating functions to find the values returned by Comp-Q(n), Comp-R(n) and Comp-S(n).

(b) [**20 Points**] Use generating functions to find the running times of Comp-Q(n), Comp-R(n) and Comp-S(n).

(c) [**10 Points**] Based on part (a) can you give algorithms to compute the values returned by Comp-Q(n), Comp-R(n) and Comp-S(n) in $\mathcal{O}(n)$ time? Can you compute them in $o(n)$ time? Why or why not?