CSE548, AMS542: Analysis of Algorithms, Fall 2017 Date: Dec 1

Homework #3

Group Number: 52

Name	SBU ID	% Contribution
Rahul Bhansali	111401451	33.33%
Kiranmayi Kasarapu	111447596	33.33%
Neha Indraniya	111499447	33.33%

Collaborating Groups

Group Number	Specifics Number (e.g., specific group member? specific task/subtask?)
19	Question 3
21	Question 2 and Question 3
38	Question 1, Question 2, Question 3
42	Question 2 and Question 3
23	Question 3
17	Question 2
26	Question 2 and Question 3
24	Question 1(e), Question 2(c) and 2(f), Question 3(b)

Task 1. [60 Points] A Simple Randomized Selection Algorithm
(a) [5 Points] Explain how you will implement step 2 of RandSelect-1

When k ≤ n/logn we use Min heap to extract kth smallest element . Extracting smallest element from the min heap takes logn time . To obtain kth smallest element we need to perform extract-min at the most n/logn times.

n/logn*O(logn) = O(logn * n/logn) = O(n)

When k ≥ n − n/logn we build Max heap to extract kth smallest element. Extracting largest element from the max heap takes logn time . To obtain kth smallest element we need to perform extract-max at the most n/logn times.

n/logn * O(logn) = O(logn. n/logn) = O(n)

In both the case, we will build min heap or max heap respectively. As learnt in class we know that building a heap of n elements takes O(n) time. So the total cost of running step 2 is O(n) + O(n) = O(n). Hence Proved.

[10 Points] Prove that w.h.p. in n no element of A is chosen more than once in step 5 of RandSelect-1.

Define Indicator random variable,

Xi = 1 if that element is chosen
Xi = 0 otherwise.

Probability of choosing 1 element out of n elements is 1/n . Also we have to select $\log^2$n elements.

Pr(Xi = 1) = 1/n
E(Xi) = 1*Pr(Xi=1) +)*Pr(Xi=0) = Pr(Xi =1) = 1/n
μ = E(X) = $\sum_{k=0}^{log^2n} \quad Pr(Xi = 1) = log^2n * 1/n = \frac{log^2n}{n}$

We have to prove that no elements are chosen more than once. I.e. an element is not chosen or it is chosen only once.
So we have to prove that Pr(X<=1) hold with high probability
Therefore, we will prove that Pr(X>=1) holds with low probability.
To prove that we will use Markov's inequality.

P(X ≥ δ) ≤ E(X)/δ
Pr(X>=1) <= E(X)/δ

In our case, δ =1
Therefore Pr(X>=1) <= E(X) = $\frac{log^2n}{n}$
We know that for large values of n, or as n increases, n will be dominating log^2n,i.e n increases at a faster rate than log^2n.
We can say that log^2n= o($n^{0.2}$)
Therefore $\lim\limits_{n\to\infty} \frac{log^2n}{n^{0.2}} = 0$
So Pr(X>=1) = o($n^{0.2}$) / n = $\frac{1}{n^{0.8}}$which is very small and decreases as n increases.
Therefore the probability that elements will be chosen more than once is very low.

(c) [20 Points] Prove that in step 17 of RandSelect-1, T = O(n/logn) holds w.h.p. in n.

Define a 0-1 random variable X_i as:

X_i = 1 if an element lies in a particular bin (say T)
X_i = 0 otherwise.

So, $E[X_i] = \frac{1}{(log\ n)^2}$

Thus, $\mu = E[X] = \sum_{i=1}^{n} E[X_i] = \frac{n}{(log\ n)^2}$

To prove an upper bound on time complexity, it is sufficient to prove that:

$P(X < \frac{n}{log\ n})$is high.

=> $P(X \geq \frac{n}{log\ n})$ is low.

Thus, we will use below Chernoff Bound:

$$P(X \geq (1+\delta)\mu) \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}$$

Here,

=> $(1+\delta)\mu = \frac{n}{log\ n}$

=> $(1+\delta)\frac{n}{(log\ n)^2} = \frac{n}{log\ n}$ [Using μ value from above]

=> $(1+\delta) = log\ n$

=> $\delta = log\ n - 1$

Now, applying Chernoff Bound:

$P(X \geq \frac{n}{log\ n}) \leq \left(\frac{e^{(log\ n-1)}}{(log\ n)^{(log\ n)}}\right)^{\frac{n}{(log\ n)^2}}$

$$P(X \geq \frac{n}{log\ n}) \leq \left(\frac{e^{(log\ n-1)\frac{n}{(log\ n)^2}}}{(log\ n)^{(log\ n)\frac{n}{(log\ n)^2}}}\right)$$

$$P(X \geq \frac{n}{log\ n}) \leq \left(\frac{e^{\frac{n}{log\ n}-\frac{n}{(log\ n)^2}}}{(log\ n)^{\frac{n}{log\ n}}}\right)$$

$$P(X \geq \frac{n}{log\ n}) \leq \left(\frac{e^{\frac{n}{log\ n}\left(1-\frac{1}{log\ n}\right)}}{(log\ n)^{\frac{n}{log\ n}}}\right)$$

Thus,

$$P(X < \frac{n}{log\, n}) \le 1 - \left(\frac{e^{\frac{n}{log\, n}\left(1 - \frac{1}{log\, n}\right)}}{(log\, n)^{\frac{n}{log\, n}}} \right)$$

Here, we can see that $e^{\frac{n}{log\, n}\left(1 - \frac{1}{log\, n}\right)}$is a constant and this numerator will be dominated by the denominator i.e. $(log\, n)^{\frac{n}{log\, n}}$.

Hence, as n becomes large, the whole expression $\left(\frac{\square^{\frac{n}{log\, n}\left(1 - \frac{1}{log\, n}\right)}}{(log\, n)^{\frac{n}{log\, n}}} \right)$ becomes low.

Therefore, $1 - \left(\frac{e^{\frac{n}{log\, n}\left(1 - \frac{1}{log\, n}\right)}}{(log\, n)^{\frac{n}{log\, n}}} \right)$ is a high probability in n.

Hence, proved.

(d) [15 Points] Give an upper bound (the best you can come up with) on the running time of RandSelect-1 that holds w.h.p. in n

We shall analyze the algorithm line by line.

Lines 1-3	O(n) as shown in 1(a)
Lines 5-6	O(log^{2}n) , choosing log^{2}n random elements
Line 7	O(1)
Line 8	θ(log^{2}n.log(log^{2}n)) = θ(log^{2}n.log(logn)) --sorting log^{2}n elemnets
Line 9	O(log^{2}n) -- variable assignment for log^{2}n variables
Lines 10-12	in line 11 do binary search on log^{2}n elements to find in which bin the element belong to..Thus time taken is θ(log(log^{2}n)). We do it n times in step 10, Therefore θ(nlog(log^{2}n)) = θ(nlog(logn))
Line 13,	O(log^{2}n) -- as we go through at max q ci's, and q<=log^{2}n
Line 14	O(1)
Lines 15-16	O(n) -- we found the bin in step 13. Now we just go through elements in A and transfer to set T
Line 17	O(n/logn.log(n/logn)) -- In step © we proved that each bin will have O(n/logn) elements w.h.p. Therefore sorting n/logn elements takes O(n/logn.log(n/logn))
Line 18	O(1)

So the overall time complexity is: (We write only the dominating terms)
T(n) = O(n) + θ($\log^2$n.log(logn)) + O(n/logn.log(n/logn)) + θ(nlog(logn)) = θ(nlog(logn))

(e) [10 Points] Does RandSelect-1 ever fail to produce an answer? Why or why not? Is the answer it produces always guaranteed to be correct? Why or why not?

RANDSELECT-1 algorithm never fails to return a wrong answer. It always gives the correct answer.
The reason is, the only randomization we are doing is at Step 5 of the algorithm, where we are choosing $log^2 n$ pivots.
These pivots are used to build buckets for elements of set A. In each bucket we are maintaining a count of number of elements from set A, that belong to a given range. Doing this we will never miss any element from A, since we even defined s0 as -inf and sq+1 as +inf.
So all elements from set A, will definitely belong to one of the buckets Bi.
Now since the pivots are already sorted in step 8 of the algorithm. Value of elements in bucket B(i-1), will always be less than value of elements in Bi. Although we are not copying each element into the bucket, we are just maintaining a count of number of elements that belong to a particular bucket.
Now when we want to find the Kth smallest element, we will have to find out, to which bucket it will belong to, and the kth smallest element will definitely be in that range of the buckets. As the bucket ranges are sorted.
Therefore, once we find out in which bucket the kth smallest element lies, we extract those elements into set T. We sort the elements in set T(as mentioned earlier, elements in particular bucket may not be sorted. We just have the count of the number of elements that belong to that bucket). To find the kth smallest smallest element from this set T, we will have to find its position in set T, which is calculated by t in the algorithm. This will be nothing but k - (number of elements in previous bins). And now we return that value, which is exactly the kth smallest element.

Task 2. [75 Points] A Not So Simple Randomized Selection Algorithm
(a) [10 Points] Prove that after the loop in steps 11–12 terminates |A0 | = Θ (n/loglog n) holds w.h.p. in n.

Define an indicator variable Xi such that

Xi = 1, if the element from A is added to A'
0, otherwise

Then the expected value of can be calculated as

E[X_i] = 1 * Pr($X_i = 1$) + 0 * Pr($X_i = 0$)
= Pr($X_i = 1$)

i.e. probability of an element being picked from A to A' = $\frac{1}{loglogn}$
expected value E[X] is calculated as follows :-

$$E[X] = \sum_{i=1}^{n} E[X_i]$$
$$= E[X] = \sum_{i=1}^{n} \frac{1}{loglogn}$$
$$= \frac{n}{loglogn}$$

Here we have to prove that, |A'| = $\theta(\frac{n}{loglogn})$ w.h.p in n which means that

|A'| ≥ $c_1 \frac{n}{loglogn}$ and |A'| ≤ $c_2 \frac{n}{loglogn}$
We have to prove both of the terms occurs with a high probability in n, Therefore,
|A'| ≤ $c_1 \frac{n}{loglogn}$ + e and |A'| ≥ $c_2 \frac{n}{loglogn}$ - e happens with low probability in n.
We will apply chernoff bounds and use $\delta = 1/2$

$$\Pr(X \le ((1-(\tfrac{1}{2}))\frac{n}{loglogn}) \le e^{-\mu\delta^2/2}$$

Solving it further,

$$\Pr(X \le (\tfrac{1}{2})\frac{n}{loglogn}) \le e^{-\frac{n}{loglogn}/8}$$

Since, as the value of n increases, the right hand term will approach closer to zero. Hence $\Pr(X \le (\frac{1}{2})\frac{n}{loglogn})$ is occurring with a low probability as n dominates.

Similarly,

$$\Pr(X \ge (1+(\tfrac{1}{2}))\frac{n}{loglogn}) \le e^{-\mu\delta^2/3}$$
$$\Pr(X \ge (\tfrac{3}{2})\frac{n}{loglogn}) \le e^{-\mu\delta^2/3}$$

Putting the values of δ *and* μ, we will obtain

$$\Pr(X \ge (\tfrac{3}{2})\frac{n}{loglogn}) \le e^{-\frac{n}{loglogn}/12}$$

Similarly, for this equation also, $\Pr(X \ge (\frac{3}{2})\frac{n}{loglogn})$ is occurring with a low probability as n dominates.

Hence, |A'| = $\theta(\frac{n}{loglogn})$ hold w.h.p in n.

2(b) [15 Points] Let Δ = kright – kleft. Show that w.h.p. in n, 2√ m log n ≤ Δ ≤ 2 √ 2m log n.

Here, we are given $\Delta = k_{right} - k_{left}$ where,

$$k_{right} = k' + 2(\sqrt{(m-k')logn})$$
$$k_{left} = k' - 2(\sqrt{k'logn})$$
$$\Delta = k_{right} - k_{left}$$
$$= k' + 2(\sqrt{(m-k')logn}) - (k' - 2\sqrt{k'logn})$$
$$= 2(\sqrt{(m-k')logn}) + 2\sqrt{k'logn})$$
$$= 2\sqrt{logn})(\sqrt{(m-k')} + \sqrt{k'})$$

We are already given that k' position will be k * m/n. Substituting that value of k' in the Δ equation, we will obtain,

$$\Delta = 2\sqrt{mlogn})(\sqrt{1-k/n} + \sqrt{k/n})$$

Now, let us calculate the expected value of E[Δ].

For this, define an indicator variable

$X_i =$ 1, if the element from lies between k_{right} *and* k_{left}

0, otherwise

$$E[X_i] = 1 * \Pr(X_i = 1) + 0 * \Pr(X_i = 0)$$
$$= \Pr(X_i = 1)$$
$$= \frac{2\sqrt{mlogn})(\sqrt{1-\frac{k}{n}} + \sqrt{k/n})}{m}$$

expected value E[X] is calculated as follows :-

$$E[X] = \sum_{i=1}^{m} E[X_i]$$
$$E[X] = \sum_{i=1}^{m} \frac{2\sqrt{mlogn})(\sqrt{1-\frac{k}{n}} + \sqrt{k/n})}{m} \text{ x m}$$
$$\mu = E[X] = 2\sqrt{mlogn})(\sqrt{1-k/n} + \sqrt{k/n})$$

Now , we prove $2\sqrt{mlogn} \le \Delta \le 2\sqrt{2mlogn}$ w.h.p in n.

Which means we have to prove that,

$\Pr(2\sqrt{mlogn} \leq \Delta)$ w.h.p in n and $\Pr(2\sqrt{2mlogn} \geq \Delta)$ w.h.p in n

we'll prove,

$\Pr(2\sqrt{mlogn} \geq \Delta)$ occurs with a low probability in n and $\Pr(2\sqrt{2mlogn} \leq \Delta)$ occurs with a low probability in n.

So let us see how to solve the following:

$$\Pr(\Delta \geq 2\sqrt{2mlogn})$$

Let us equate this, with the chernoff bounds equation,

$$(1+\delta)\mu = 2\sqrt{2mlogn}$$

$$(1+\delta)\, 2\sqrt{mlogn})(\sqrt{1-k/n} + \sqrt{k/n}\ ) = 2\sqrt{2mlogn}$$

$$(1+\delta) = \frac{\sqrt{2}}{(\sqrt{1-\frac{k}{n}} + \sqrt{k/n}\,)}$$

$$\delta = \frac{\sqrt{2}}{(\sqrt{1-\frac{k}{n}} + \sqrt{k/n}\,)} - 1$$

Here , we can see that the value of k ranges from n/logn to n-n/logn only. If we put the value of k as n-n/logn, then the value of δ will somewhere lie between (0,1). Similarly , if we take the value of k as n/logn, then also the value of δ will somewhere lie between (0,1).

Hence we can be assured that δ value always ranges from 0 to 1.

Therefore, we can apply the chernoff bound rule 2 which is

$$\Pr(\Delta \geq 2\sqrt{2mlogn}) \leq e^{-\mu\delta^2/3}$$

We have calculated both δ and μ. Let us substitute it

$$\Pr(\Delta \geq 2\sqrt{2mlogn}) \leq e^{-2\sqrt{mlogn})(\sqrt{1-\frac{k}{n}} + \sqrt{\frac{k}{n}})*(\frac{\sqrt{2}}{\left(\sqrt{1-\frac{k}{n}} + \sqrt{\frac{k}{n}}\right)} - 1)^2/3}$$

$$\Pr(\Delta \geq 2\sqrt{2mlogn}) \leq e^{-2\sqrt{mlogn}(\sqrt{2} - (\sqrt{1-\frac{k}{n}} + \sqrt{\frac{k}{n}}))^2/3}$$

As k approaches towards n-n/logn, for simplicity we'll consider k to be n , the whole term can then be replaced by a constant

$$\Pr(\Delta \geq 2\sqrt{2mlogn}) \leq e^{-c\sqrt{mlogn}}$$

Hence, as the n increases the values of $\Pr(\Delta \geq 2\sqrt{2mlogn})$ approaches 0. Hence , $\Pr(\Delta \geq 2\sqrt{2mlogn})$ occurs with a low probability in n.

Similarly, let us know prove for

$$\Pr(\Delta \leq 2\sqrt{mlogn})$$

Let us equate this, with the chernoff bounds,

$$(1-\delta)\mu) = 2\sqrt{mlogn}$$

$$(1-\delta)\, 2\sqrt{mlogn})(\sqrt{1-k/n} + \sqrt{k/n}\ ) = 2\sqrt{mlogn}$$

$$(1-\delta) = \frac{1}{(\sqrt{1-\frac{k}{n}} + \sqrt{k/n}\,)}$$

$$\delta = \frac{1}{(\sqrt{1-\frac{k}{n}} + \sqrt{k/n}\,)} +1$$

Again here , we can see that the value of k ranges from 1 to n (It actually varies from n/logn to n-n/logn, but for analysis purpose for this case we'll assume from 1 to n, as we can clearly deduce the same for this range also). If we put the value of k as n, then the value of δ will be between (0,1). Similarly , if we take the value of k as n, then also the value of δ will lie between (0,1).

Therefore, we can be assured that δ value always ranges from 0 to 1.
Therefore, we can apply the chernoff bound rule 2 which is

$$\Pr(\Delta \leq 2\sqrt{mlogn}) \leq e^{-\mu\delta^2/2}$$

We have calculated both δ and μ. Let us substitute it

$$\Pr(\Delta \leq 2\sqrt{mlogn}) \leq e^{-2\sqrt{mlogn})(\sqrt{1-\frac{k}{n}}+\sqrt{\frac{k}{n}})*(\frac{1}{\left(\sqrt{1-\frac{k}{n}}+\sqrt{\frac{k}{n}}\right)}+1)^2/2}$$

$$\Pr(\Delta \leq 2\sqrt{mlogn}) \leq e^{-2\sqrt{mlogn}(\ 1+\ (\sqrt{1-\frac{k}{n}}+\sqrt{\frac{k}{n}}))^2/2}$$

As k approaches towards n, the whole term can then be replaced by a constant

$$\Pr(\Delta \leq 2\sqrt{mlogn}) \leq e^{-c\sqrt{mlogn}}$$

Hence, as the n increases the values of $\Pr(\Delta \leq 2\sqrt{mlogn})$ approaches 0. Hence, $\Pr(\Delta \leq 2\sqrt{mlogn})$ occurs with a low probability in n. Therefore, $2\sqrt{mlogn} \leq \Delta \leq 2\sqrt{2mlogn}$ occurs w.h.p in n.

(c) [20 Points] Let xleft (resp. xright) be the kleft-th (resp. kright-th) smallest element of A0 after the termination of the loop in steps 11–12. Prove that w.h.p. in n, A does not have more than 4√ 2m log n log log n elements with value in [xleft, xright].

From RandSelect-2 algorithm, the kth position element of A will possibly end up at k' position in A' which is equal to k * (m/n).
In this problem, we are mapping A' to A, based on the same logic of mapping from A to A', the kth positional element in A' will possibly map to k * (n/m)th position in A.

Now let us define an indicator variable:

X_i = 1, if the element's value lies in between x_{right} *and* x_{left}
0, otherwise

where x_{right} is the kright-th smallest element of A' and x_{left} is the kleft-th smallest element of A' as given
we calculated in 2(b) the value of $\Delta = k_{right} - k_{left}$.
We'll define k'_{right} *and* k'_{left} which are the positions of x_{right} and x_{left} that appears in A respectively. So,

$$\Delta' = k_{right}' - k_{left}'$$

We know $k'_{right} = k_{right}$ * (n/m) and $k'_{left} = k_{left}$ * (n/m) from the argument we provided above. So,

$$\Delta' = k_{right} * \left(\frac{n}{m}\right) - k_{left} * \left(\frac{n}{m}\right)$$
$$= \Delta * (n/m)$$

Expected value of Δ' is

$$E[\Delta'] = E[\Delta * (\frac{n}{m})]$$

$$= (n/m) * E[\Delta]$$

$$= (n/m) * \mu$$

$$= \mu'$$

where $\mu = E[\Delta] == 2\sqrt{mlogn})(\sqrt{1-k/n} + \sqrt{k/n}$).

We need to prove that w.h.p. in n, A does not have more than $4\sqrt{2mlogn}$)) loglogn elements with value in [xleft, xright].

Which means we have to prove that

$$\Pr(X \leq 4(\sqrt{2mlogn}) \text{ loglogn) is high i.e w.h.p in n}$$

We will prove

$$\Pr(X \geq 4\sqrt{2mlogn})) \text{ loglogn) is low as n increases}$$

Let us apply chernoff bounds now in the above equation

$$(1+\delta)\mu' = 4\sqrt{2mlogn})) \text{ loglogn)}$$

But, μ' = (n/m) * μ, substitute this

$$(1+\delta)(n/m) * \mu = 4(\sqrt{2}mlogn) \text{ loglogn)}$$

We know the value of m, which is nothing but

$$m = \frac{n}{loglogn} \quad \text{whp n as proved in 2(a)}$$

So,

$$(1+\delta)(\text{nloglogn}/n) * \mu = 4(\sqrt{2}mlogn) \text{ loglogn)}$$

$$(1+\delta) * \mu = 4(\sqrt{2}mlogn)$$

$$(1+\delta) * 2\sqrt{mlogn})(\sqrt{1-k/n} + \sqrt{k/n}) = 4(\sqrt{2}mlogn)$$

$$(1+\delta) = 2\sqrt{2} / \sqrt{1-k/n} + \sqrt{k/n}$$

$$\delta = 2\sqrt{2} / \sqrt{1-k/n} + \sqrt{k/n} - 1$$

Now, we have derived both δ and μ, let us use this value to prove

$$\Pr(X \geq 4(\sqrt{2}mlogn) \text{ loglogn)}$$

Applying Chernoff's bound,

$$\Pr(X \geq 4(\sqrt{2}mlogn) \text{ loglogn)} \leq (\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}})^{\mu'}$$

We'll keep, for simplicity

$$x = \sqrt{1-k/n} + \sqrt{k/n}$$

$$\Pr(X \geq 4(\sqrt{2}mlogn) \text{ loglogn)} \leq (\frac{e^{\frac{2\sqrt{2}}{x}-1}}{\left(\frac{2\sqrt{2}}{x}\right)^{\left(\frac{2\sqrt{2}}{x}\right)}})^{(n/m) * \mu}$$

$$\Pr(X \geq 4(\sqrt{2}mlogn) \text{ loglogn)} \leq ((\frac{1}{e}) \frac{e^{\frac{2\sqrt{2}}{x}}}{\left(\frac{2\sqrt{2}}{x}\right)^{\left(\frac{2\sqrt{2}}{x}\right)}})^{(n/m) * \mu}$$

$$\Pr(X \geq 4(\sqrt{2}mlogn) \text{ loglogn)} \leq ((\frac{1}{e}) \frac{e^{\frac{2\sqrt{2}}{x}}}{\left(\frac{2\sqrt{2}}{x}\right)^{\left(\frac{2\sqrt{2}}{x}\right)}})^{(n/m) * 2\sqrt{mlogn})(x)}$$

$$\Pr(X \geq 4(\sqrt{2}mlogn) \text{ loglogn)} \leq ((\frac{1}{e^{x}}) \frac{e^{2\sqrt{2}}}{\left(\frac{2\sqrt{2}}{x}\right)^{(2\sqrt{2})}})^{(\text{nloglogn}/n) * 2\sqrt{logn(\frac{n}{loglogn}))}}$$

$$\Pr(X \geq 4(\sqrt{2}mlogn) \text{ loglogn)} \leq ((\frac{1}{e^{x}})(\frac{e}{\frac{2\sqrt{2}}{x}})^{2\sqrt{2}})^{(2\sqrt{nlognloglogn})}$$

Here, we can observe that, the term $(\frac{e}{\frac{2\sqrt{2}}{x}})^{2\sqrt{2}})$, will be less than 1. The reason is $2\sqrt{2}$ will dominate e.

Value of e = 2.71828,
Value of $2\sqrt{2}$ = 2.82842
Value of x can at max be 1, we proved in 2(b). So $(\frac{e}{\frac{2\sqrt{2}}{x}})^{2\sqrt{2}})$ will always be < 1.

The power term $\left(2\sqrt{nlognloglogn}\right)$ keeps increasing as n increases. As all are increasing functions. Therefore $((\frac{1}{e^x})(\frac{e}{\frac{2\sqrt{2}}{x}})^{2\sqrt{2}})^{(2\sqrt{nlognloglogn})}$ decreases and becomes very small as n increases.

So Pr(X $\geq$ 4($\sqrt{2mlogn}$) loglogn) is very less as n increases. Therefore Pr(X $\leq$ 4($\sqrt{2mlogn}$) loglogn) is high as n increases. Therefore number of elements that lie between [xleft,xright] in A is bounded by O(4($\sqrt{2mlogn}$) loglogn) whp in n.

(d) [10 Points] Prove that in step 22 of RandSelect-2, |T| = O (n/ log n) holds w.h.p. in n.

For this problem,
We have to prove that, |T| = $O(\frac{n}{logn})$ w.h.p in n.
From the algorithm (Lines 20-21), we can see that the number of elements in T directly depends on the number of elements in a bin.
So, lets find w.h.p the number of elements in a bin.
Define an indicator variable X_i such that

X_i = 1, if the element from A' lies in a particular bin (say B).
0, otherwise

Then the expected value of X_i can be calculated as

E[X_i] = 1 * Pr($X_i = 1$) + 0 * Pr($X_i = 0$)
= Pr($X_i = 1$)

which is probability of an element lying in a particular bin:

$= \frac{1}{(log\ n)^2}$ (as there are $(\log n)^2$ bins)

The total expected value of E[X] will be:-

$$E[X] = \sum_{i=1}^{n/\log(\log n)} E[X_i]$$

Here the summation runs till $n/\log(\log n)$ as that is the number of elements expected in A' w.h.p in n.

$$E[X] = \sum_{i=1}^{n/\log(\log n)} \frac{1}{(\log n)^2} = \frac{n}{\log(logn)\ (\log n)^2} = \mu$$

Now, if we can show that Pr($X < \frac{n}{\log n}$) w.h.p in n then it also means |T| = $O(\frac{n}{logn})$ w.h.p in n as |T| will be some constant factor times the number of elements in a bin.

So, we need to show: Pr($X < \frac{n}{\log n}$) is high

⇨ Pr($X \geq \frac{n}{\log n}$) is low.

So, we'll use Chernoff bound:

$$\Pr(X \geq (1+\delta)\mu) \leq (\frac{e^\delta}{(1+\delta)^{(1+\delta)}})^\mu$$

Here,

$$(1+\delta)\mu = \frac{n}{\log n}$$

$$(1+\delta)\left(\frac{n}{(\log\log n)(\log n)^2}\right) = \frac{n}{\log n}$$

$$(1+\delta) = (\log n)(\log\log n)$$

$$\delta = (\log n)(\log\log n) - 1$$

Now, applying Chernoff Bound:

$$\Pr(X \geq (1+\delta)\mu) \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}$$

$$\Pr\left(X \geq \frac{n}{\log n}\right) \leq \left(\frac{e^{\log n \log\log n - 1}}{(\log n \log\log n)^{(\log n \log\log n)}}\right)^{\frac{n}{(\log n)^2 \log\log n}}$$

$$\Pr\left(X \geq \frac{n}{\log n}\right) \leq \left(\frac{e^{\frac{n}{\log n} - \frac{n}{(\log n)^2 \log\log n}}}{(\log n \log\log n)^{\frac{n}{\log n}}}\right)$$

$$\Pr\left(X \geq \frac{n}{\log n}\right) \leq \left(\frac{e^{\frac{n}{\log n}(1 - \frac{1}{\log n \log\log n})}}{(\log n \log\log n)^{\frac{n}{\log n}}}\right)$$

In the above expression, we can see that the numerator $e^{\frac{n}{\log n}(1-\frac{1}{\log n \log\log n})}$ is a constant and its value will be dominated by the denominator i.e. $(\log n \log\log n)^{\frac{n}{\log n}}$ as the value of n increases.
So, the whole expression:

$$\frac{e^{\frac{n}{\log n}(1-\frac{1}{\log n \log\log n})}}{(\log n \log\log n)^{\frac{n}{\log n}}}$$ becomes low as n becomes large.

Therefore,

$\Pr(X \geq \frac{n}{\log n})$ is low and hence,

$\Pr(X < \frac{n}{\log n})$ = 1 - $\Pr(X \geq \frac{n}{\log n})$

= 1 - $\frac{e^{\frac{n}{\log n}(1-\frac{1}{\log n \log\log n})}}{(\log n \log\log n)^{\frac{n}{\log n}}}$

= high probability in n.

So, we have shown that the number of elements in a bin will be $O(\frac{n}{logn})$ w.h.p in n.
As |T| will be some constant factor times number of elements in a bin, so we can say |T| = $O(\frac{n}{logn})$.
Hence, proved.

(e) [10 Points] Give an upper bound (the best you can come up with) on the running time of RandSelect-2 that holds w.h.p. in n.
We shall analyze the algorithm line by line.

Line 1 – 3	O(n) -- Same as 1(a)
Line 5 – 6	O($log^2 n$) -- we are choosing only $log^2 n$ elements out of the total n elements
Line 7	O(1) – Declaration of variables
Line 8	$\theta(\log^2 n.\log(\log^2 n)) = \theta(\log^2 n.\log(\log n))$ --sorting $\log^2$n elemnets

Line 9-10	O($\log^2$n) -- variable assignment for $\log^2$n variables
Line 11-12	O(n) – scanning set A of n elements and choosing each element with 1/loglogn
Line 13 – 15	In line 14, we'll do a binary search to find in which bin the element lies for all log^2n bins we do this for each element in A'. The number of elements in A' = θ(n/loglogn) whp, therefore running time for these line is O($\frac{n}{loglogn} *$ log (log^2n)) = O($\frac{n}{loglogn} * 2 *$ (log($logn$)) = O(n)
Line 16	O(1) -- Initializations
Line 17 – 18	We are finding the bin in which k_left and k_right lies respectively. Total number of bins is (log^2n). So running time is O(log^2n)
Line 19	O(1) -- Initializations
Line 20-21	Scanning Set A, O(n)
Line 22	In 2(d), we prove that there are O(n/logn) elements in T whp. Therefore sorting T would take O(n/logn * (log(n/logn))) = O(n-n/logn*(loglogn)) = O(n)
Line 23-24	O(1)

Overall time complexity = O(n) + O(log^2n) + θ($\log^2$n.log(logn)) + O($\log^2$n) = O(n)

(f) [10 Points] Does RandSelect-2 ever fail to produce an answer? Why or why not? Is the answer it produces always guaranteed to be correct? Why or why not?
RADNSELECT-2 fails to produce a correct answer in some cases, therefore it does not produce correct answer always. We give an example case, where it fails to produce a correct answer.

We are choosing element into A' with probability 1/loglogn. Therefore,
Probability of an element from A not getting selected for A' = 1 - $\frac{1}{loglogn}$
Probability that no element from A gets selected for A' =

$$(1 - \frac{1}{loglogn})^n \text{ which is greater than 0}$$

Therefore A' will be NULL in this case.

If A' = NULL {which means the A' array size is 0}
From lines 13 to 15, we can observe that the c_i will be 0, as it will not enter for loop.
From lines 16, we can see that m becomes 0, and similarly k', k_{right}, k_{left} also reduces down to 0.
lines 17 & 18 will give, l = 0 and r =0,

In line 2, we have selected log^2n elements, lets say we selected the smallest element as one of the pivot. So s1 in line 8 will be the smallest element or the smallest pivot.

Now in lines 20-21, we are selecting all elements from A, that lie between s0 and s1 i.e –inf to smallest element in A, which will be 0 elements.

Now for any k>0, this particular run of the algorithm will return wrong answer.

Thus, the above algorithm fails to give correct answer and hence doesn't always return the correct kth smallest element in A.

Task 3. [45 Points] Flipping Coins in NCS 220
(a) [5 Points] Prove that the expected number of equipment activated by RAND-NCS is $61/64\, n$ (i.e., 95.3125% of n).
Solution:
It is given that 10% of the equipment had 3 control buttons each, 20% had 4 each, and each of the remaining 70% had 5. In RAND-NCS flips the button based on a coin flip. So, for 10% of equipment as 3 buttons should be on so for that the probability of being all off, i.e probability that 3 coin flips result in tails is $(1/2^3)$. Therefore the probability that at least one button is on is $(1 - 1/2^3)$. Similarly, for 20% of equipments as 4 buttons should be on so for that the probability of being off is $(1/2^4)$and for at least one to be on is $(1 - 1/2^4)$ and for 70% of equipments as 5 buttons should be on so for that the probability of being off is $(1/2^5)$, and at least one being on is $(1 - 1/2^5)$

So Expected number of equipment that will be switched on is given by:-

$E(X) = (10/100)n * (1 - 1/2^3) + (20/100)n * (1 - 1/2^4) + (70/100)n * (1 - 1/2^5)=$ 61n/64

So, expected number of equipment activated by RAND-NCS is 61n/64

(b) [15 Points] Prove that the probability that RAND-NCS at least 61n/64 equipment is at least 1/64n.
Solution:
We need to prove that:$\boldsymbol{Pr(x >= 61n/64) >= 1/64\, n}$

$$\boldsymbol{E[X]} = \sum_{k=0}^{n} k\, Pr(x = k)$$

As E[X]=$\frac{61n}{64}$, substituting the value in above equation

$$=> \frac{61n}{64} = \sum_{k=0}^{n} k\, Pr(x = k)$$

Defining limits over upper and lower bounds:-

$$= \sum_{k=0}^{\frac{61n}{64}-\frac{1}{64}} k\, PAr(x = k) + \sum_{k=\frac{61n}{64}-\frac{1}{64}}^{n} k\, Pr(x = k)$$

$=>\frac{61n}{64} <= (\frac{61n}{64} - \frac{1}{64}) \sum_{k=0}^{\frac{61n}{64}-\frac{1}{64}} Pr(x = k) + n\, Pr(x >= \frac{61n}{64})$

The maximum value of$\sum_{k=0}^{\frac{61n}{64}-\frac{1}{64}} Pr(x = \frac{61n}{64})$ will be 1. So,

$$=> \frac{61n}{64} <= (\frac{61n}{64} - \frac{1}{64}).1 + n\,Pr(x >= \frac{61n}{64})$$
$$=> \frac{61n}{64} - (\frac{61n}{64} - \frac{1}{64}) <= n\,Pr(x >= \frac{61n}{64})$$
$$=> \frac{1}{64} <= n\,Pr(x >= \frac{61n}{64})$$
$$=> \frac{1}{64n} <= Pr(x >= \frac{61n}{64})$$

(c) [10 Points] Use your results from part 3(b) to design an algorithm that activates at least 61n/64 equipment w.h.p. in n. What is the running time of your algorithm?
Solution:
From above we have $Pr(x >= k) => \frac{1}{64n}$,

So, $1 - Pr(x < \frac{61n}{64}) >= \frac{1}{64}n$

$$=> Pr(x < \frac{61n}{64}) <= 1 - \frac{1}{64}n$$

To get a constant probability bound, we will run the above algorithm 64n times. The probability that all the 64n runs will result in x< $\frac{61n}{64}$i.e. the number of equipment that are activated in all 64n runs is less than 61n/64. So the probability after 64n runs we will end up having less than 61n/64 is given by

$$Pr(x < \frac{61n}{64}) <= (1 - \frac{1}{64n})^{64n} < \frac{1}{e}$$

So the probability that at least one run of the 64n runs will result in $X >= \frac{61n}{64}$equipment being activated is given by

$$Pr\left(x >= \frac{61n}{64}\right) > 1 - \frac{1}{e}$$

Now, this is constant probability bound. We know that once we have an algorithm which returns a correct answer with constant probability, running the algorithm logn times will result in getting a correct answer with high probability. So now run this algorithm logn times to get correct answer w.h.p.

So, the running time complexity of this algorithm will be O(logn * 64n) * running time of RAND_NCS = O(nlogn) * RAND_NCS
We can design a lasVegas or Monte Carlo Algorithm, as we know how to check if >=95% of equipment is activated. And we know that in logn runs we will get correct answer whp in n.

(d) [10 Points] Given any ε ∈ (0, 61/64) , prove that RAND-NCS activates at least (61/64 – ε) n equipment with probability at least ∈.
Solution:
We need to prove that:$\boldsymbol{Pr(x >= (\frac{61}{64} - \in) n)} >= \in$

$$Let\ \boldsymbol{E[X]} = \sum_{k=0}^{n} k\,Pr(x = k)$$

Defining limits over upper and lower bounds:-

$$= \sum_{k=0}^{(\frac{61}{64}-\varepsilon)n} k\,PAr(x=k) + \sum_{k=(\frac{61}{64}-\varepsilon)n}^{n} k\,Pr(x=k)$$

As E[X]= $\frac{61n}{64}$, substituting the value in above equation

=>$\frac{61n}{64} <= (\frac{61}{64}-\ \boldsymbol{\varepsilon})n \sum_{k=0}^{(\frac{61}{64}-\varepsilon)n} \quad Pr(x=k) + n\,Pr(x >= (\frac{61}{64}-\ \boldsymbol{\varepsilon})n)$

The maximum value of $\sum_{k=0}^{(\frac{61}{64}-\varepsilon)n} \quad Pr(x=k)$ will be <= 1. So,

$$=> \frac{61n}{64} <= (\frac{61}{64}-\in)n\,.\,1 + n\,Pr(x >= (\frac{61}{64}-\ \boldsymbol{\varepsilon})n)$$
$$=> \frac{61n}{64} - \ (\frac{61}{64}-\in)n\ \ <= n\,Pr(x >= (\frac{61}{64}-\ \boldsymbol{\varepsilon})n)$$
$$=>\ n \in <= n\,Pr(x >= (\frac{61}{64}-\ \boldsymbol{\varepsilon})n)$$
$$=> \in <=\ Pr(x >= (\frac{61}{64}-\ \boldsymbol{\varepsilon})n)$$

(e) [5 Points] Use your results from part 3(d) to design an algorithm that activates at least (61/64 – ∈) n equipment w.h.p. in n. What is the running time of your algorithm?
Solution:
From above we have $\boldsymbol{Pr(x >=\ (\frac{61}{64}} - \in)\boldsymbol{n}\ )\ \ >= \in,$

Since this probability does not depend on n, we can assume that this is a constant probability > 0 as ∈ belongs to (0, 61/64). Since we know an algorithm which runs with constant probability, if we run this algorithm logn times, as we learned in class, then it should return a right answer i.e activate >95% of the equipment w.h.p. (Similar to 3(c) after getting constant probability)

So, the running time complexity of this algorithm will be O(logn) * running time of RAND_NCS = O(logn) * RAND_NCS