CS 170 HW 13

Due **2021-12-08, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

2 One-Sided Error and Las Vegas Algorithms

An RP algorithm is a randomized algorithm that runs in polynomial time and always gives the correct answer when the correct answer is 'NO', but only gives the correct answer with probability greater than $1/2$ when the correct answer is 'YES'.

(a) Prove that every problem in RP is in NP (i.e., show that $RP \subseteq NP$). *Hint: it may be helpful to view a randomized algorithm $R(x)$ as a deterministic algorithm $A(x, r)$ where x is the input and r is the result of the 'coin flips' which the algorithm uses for its randomness.*

(b) In lecture, we saw an example of a *Las Vegas Algorithm*: a random algorithm which always gives the right solution, but whose runtime is random. A ZPP algorithm is a Las Vegas algorithm which runs in expected polynomial time (ZPP stands for Zero-Error Probabilistic Polytime). Prove that if a problem has a ZPP algorithm, then it has an RP algorithm. *Hint: Use Markov's inequality.*

3 QuickSelect

Let A be an array of n integers, and let k be an integer. A and k are given as input. Let $\boldsymbol{X}_{i,j}$ be an indicator random variable for the event that the i-th smallest number is ever compared with the j-th smallest in $\mathsf{QuickSelect}(A, k)$.

(a) Write an exact expression for $\mathbf{E}[\boldsymbol{X}_{i,j}]$. *Hint: think about what elements will ever be chosen as pivots.*

(b) Show that the expected runtime of $\mathsf{QuickSelect}(A, k)$ is $O(n)$. *Hint: use part (a) to bound the expected number of comparisons done by the algorithm.*

1.

None, Yes

2. (a)

Let A be an algorithm that solves some of the problems in RP. A satisfies the description of RP

Notice that random algorithm will use random coin-flips to make decisions.

Let A' be a deterministic version of A. A' will use the results of coin-flips before running the same algorithm as A

Since A is an RP algorithm, there exists a polynomial-size sequence of coin flip outcomes

$c_1, \cdots, c_k \in \{0,1\}^k$ s.t A returns YES given input x and results of coin flips $c_1, \cdots, c_k$

Thus, we can view A as a NP problem if we

View the coin-flips vector as the solution of x

(b)

Let A be a zpp algorithm with $O(n^k)$, runtime

k is a constant

Define an RP algorithm B with input x of size n

(Notice that for different coin flip $A_i(x)$, $A_j(x)$ are independent. When we run multiple times, the probability of "Actual YES but get NO" will be low. We are just waiting for a different result in each coin-flip)

1) Run A $2n^k$ times

2) If A gets "YES", return YES

3) If A gets "NO", return NO

4) Otherwise, return "NO"

To prove B a RP algorithm: Notice that

if the actual answer for

x is "No" then B will always return "No"

by definition

If the actual answer is "Yes",

B will have correct answer if A stops in time,

otherwise, B will have false answer

X: a RV representing number of steps

that A will take for input x

$$Pr[X \geq a] \leq \frac{E[X]}{a} \leq \frac{n^k}{a}$$

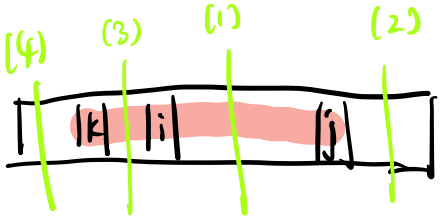
we get $Pr[X \geq 2n^k] \leq \frac{n^k}{2n^k} = \frac{1}{2}$

$\Rightarrow$ B will give the right answer with probability $\geq \frac{1}{2}$

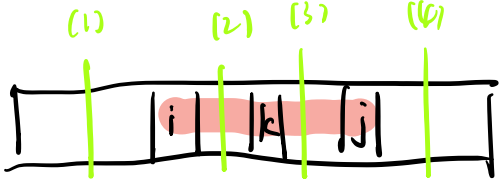
if actual answer for x is "Yes"

3. There are 3 cases in this part

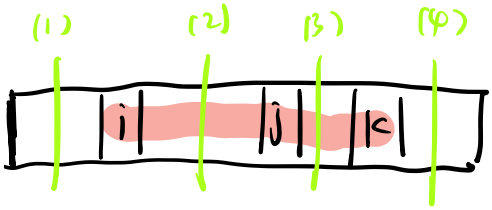
(a)



only when we choose the pivot in (4) and (2) will we have the opportunity to compare i and j



only when we pick (1) and (4) will we have the opportunity to compare i and j



only when we pick (1) and (4) will we have the opportunity to compare i and j

Since in QuickSelect, we split array according to the pivot, and we know the rank for that pivot

Let S_i be the element with rank i

① $i < j < k$. S_i is compared to S_j

iff the first pivot is in range $S_i, \dots, S_k$.

The probability is $\frac{2}{k-i+1}$

$$E[X_{ij}] = \frac{2}{k-i+1} \qquad E\left[\sum_{i<j<k} X_{ij}\right] = \sum_{i=1}^{k-2} \sum_{j=i+1}^{k-1} \frac{2}{k-i+1}$$

② $k < j$

Likewise, $\Pr[X_{i,j} = 1] = \frac{2}{j-k+1} = E[X_{i,j}]$

$$E\left[\sum_{k<i<j} X_{i,j}\right] = \sum_{j=k+1}^{n} \sum_{i=k+1}^{j-1} \frac{2}{j-k+1}$$

③ $i < k < j$

$$E[X_{i,j}] = \Pr[X_{i,j} = 1] = \frac{2}{j-i+1}$$

$$E\left[\sum_{i<k<j} X_{i,j}\right] = \sum_{i=1}^{k-1} \sum_{j=k+1}^{n} \frac{2}{j-i+1}$$

done!

(b)

Let S_j be the set of items we recursively call the algorithm on recursive level j.

$S_0 = \{1, \dots, n\}$

Define recursive level j to be "good" if $|S_{j+1}| \le \frac{3}{4}|S_j|$. $\{j_t\}$ forms a new array of good levels

Let n_i be the subinterval after the ith good level $n_i = |S_{j_{i+1}}|$ $(n_0 = n)$

Then, by definition $n_i \le \frac{3}{4} n_{i-1} \le \dots \le (\frac{3}{4})^i n$

define RV X_i to be the number of recursive calls we go through after $(i-1)$th level before reaching the ith good level

$\Rightarrow$ there are $h = \lceil \log_{\frac{4}{3}} n \rceil$ good levels

Thus, the run time is

$$|S_0| + |S_1| + \cdots \le \sum_{i=0}^{h} x_i n_i \le n \sum_{i=0}^{h} \left(\frac{3}{\varphi}\right)^i x_i$$

$$E[|S_0| + \cdots] \le n \sum_{i=0}^{h} \left(\frac{3}{\varphi}\right)^i E[x_i]$$

Since x_i should belongs to natural number

$$E[x_i] \le \sum_{k=0}^{\infty} p(x_i > k) \le \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k = 2$$

$\Rightarrow$ Expected runtime time $= 8n = O(n)$

proved!

4 Pairwise Independent Hashing

Let $\mathcal{H}$ be a class of hash functions in which each $h \in \mathcal{H}$ maps the universe $\mathcal{U}$ of keys to $\{0, 1, \ldots, m-1\}$. Recall that $\mathcal{H}$ is *universal* if for any $x \neq y \in \mathcal{U}$, $\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq 1/m$.

We say that $\mathcal{H}$ is pairwise independent if, for every fixed pair (x, y) of keys where $x \neq y$, and for any h chosen uniformly at random from $\mathcal{H}$, the pair $(h(x), h(y))$ is equally likely to be any of the m^2 pairs of elements from $\{0, 1, \ldots, m-1\}$. (The probability is taken only over the random choice of the hash function.) As an example, it turns out that $h_{a,b}(x) = ax+b \bmod m$ is a pairwise independent hash family (when the random choice is over a and b and m is prime). We will not prove this here.

(a) Show that if $\mathcal{H}$ consists of strictly fewer than m^2 functions, it cannot be pairwise independent.

(b) Show that, if $\mathcal{H}$ is pairwise independent, then it is universal.

(c) Suppose that you choose a hash function $h \in \mathcal{H}$ uniformly at random. Your friend, who knows $\mathcal{H}$ but does not know which hash function you picked, tells you a key x, and you tell her $h(x)$. Can your friend tell you $y \neq x$ such that $h(x) = h(y)$ with probability greater than $1/m$ (over your choice of h) if:

(i) $\mathcal{H}$ is universal?

(ii) $\mathcal{H}$ is pairwise independent?

In each case, either give a choice of $\mathcal{H}$ which allows your friend to find a collision, or prove that they cannot for any choice of $\mathcal{H}$.

5 Two-level Hashing

In lecture we saw a data structure for the static dictionary problem using $O(n^2)$ memory with $O(1)$ worst case query time (not just expected), and $O(n^2)$ expected preprocessing time. Here by preprocessing time, we mean the time it takes to create the data structure given the database (recall we picked $O(1)$ random hash functions in expectation from a universal hash family until we found one that causes no collisions, and for any hash function choice we could check whether there are any collisions and build the hash table in $O(n^2)$ time).

In this problem, we will develop a static dictionary data structure with $O(n)$ memory, $O(1)$ worst case query time, and $O(n)$ expected preprocessing time. The idea will be to use *two-level hashing*. As in lecture, we assume a database of size n with keys in the domain $[U] := \{0, \ldots, U-1\}$.

(a) Consider picking a hash function $h : [U] \to [m]$ randomly from a 2-wise independent family. Let X_i then be the number of database keys x such that $h(x) = i$. Write down an exact expression for $\mathbb{E}(\sum_{i=0}^{m-1} X_i^2)$ in terms of only m and n.

(b) Give a static dictionary data structure with $O(n)$ memory, $O(1)$ worst case query time, and $O(n)$ expected preprocessing time. **Hint:** Pick a 'good' h using part (a) with $m = n$ and consider using universal hash functions $h_1, \ldots, h_m$ with $h_i : [U] \to [X_i^2]$ with the solution from class.

4.

(a) From context, the pair $(h(x), h(y))$ is equally likely to be any of m^2 pairs of element from $\{0, \dots, m-1\}$

(x, y) given x. $h(x)$ and $|H| < m^2$

$(h(x), h(y))$

We only have $|H|$ buckets but with m^2 pairs

$\Rightarrow$ we can uniformly put them into $|H|$ buckets

$\Rightarrow$ it's not pairwise independent

(b) By definition a pairwise independent hash is also an universal hash because for $\forall\ x \neq y$

$$\Pr[h(x) = h(y)] = \sum_{m \in M} \Pr[h(x) = h(y) = m] = \frac{|M|}{|M|^2} = \frac{1}{|M|}$$

(c)

i) No, by definition $\Pr_{h \in H}[h(x) = h(y)] \leq \frac{1}{m}$. Thus, my friend can't find $y \neq x$ s.t $h(x) = h(y)$ with probability $> \frac{1}{m}$

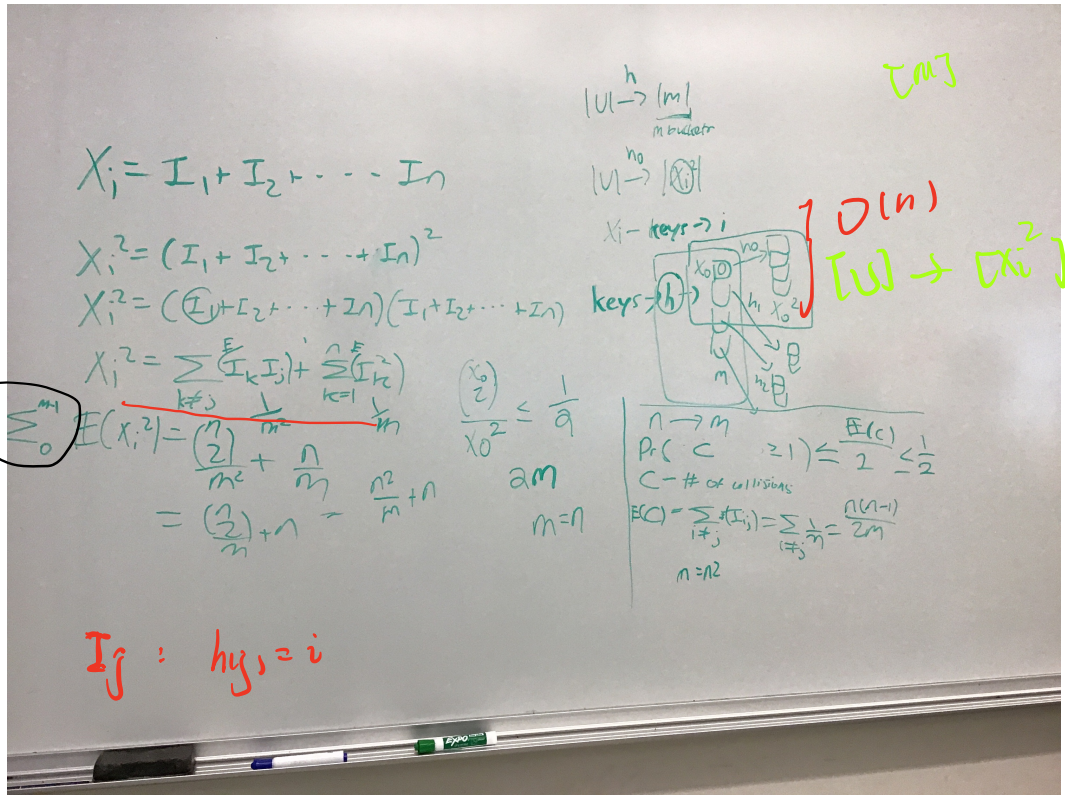
ii) No, by definition $\Pr_{h \in H}[h(x) = a \wedge h(y) = b] = \frac{1}{m^2}$ $(a, b \in \{1, \dots, m\})$

Set a, b to be the same

then for every 2 elements $x, y \in U$ $h(x) = h(y)$ will collide with probability $\frac{1}{M}$

(a)

$$E\left(\sum_{i=0}^{m-1} X_i^2\right) =$$



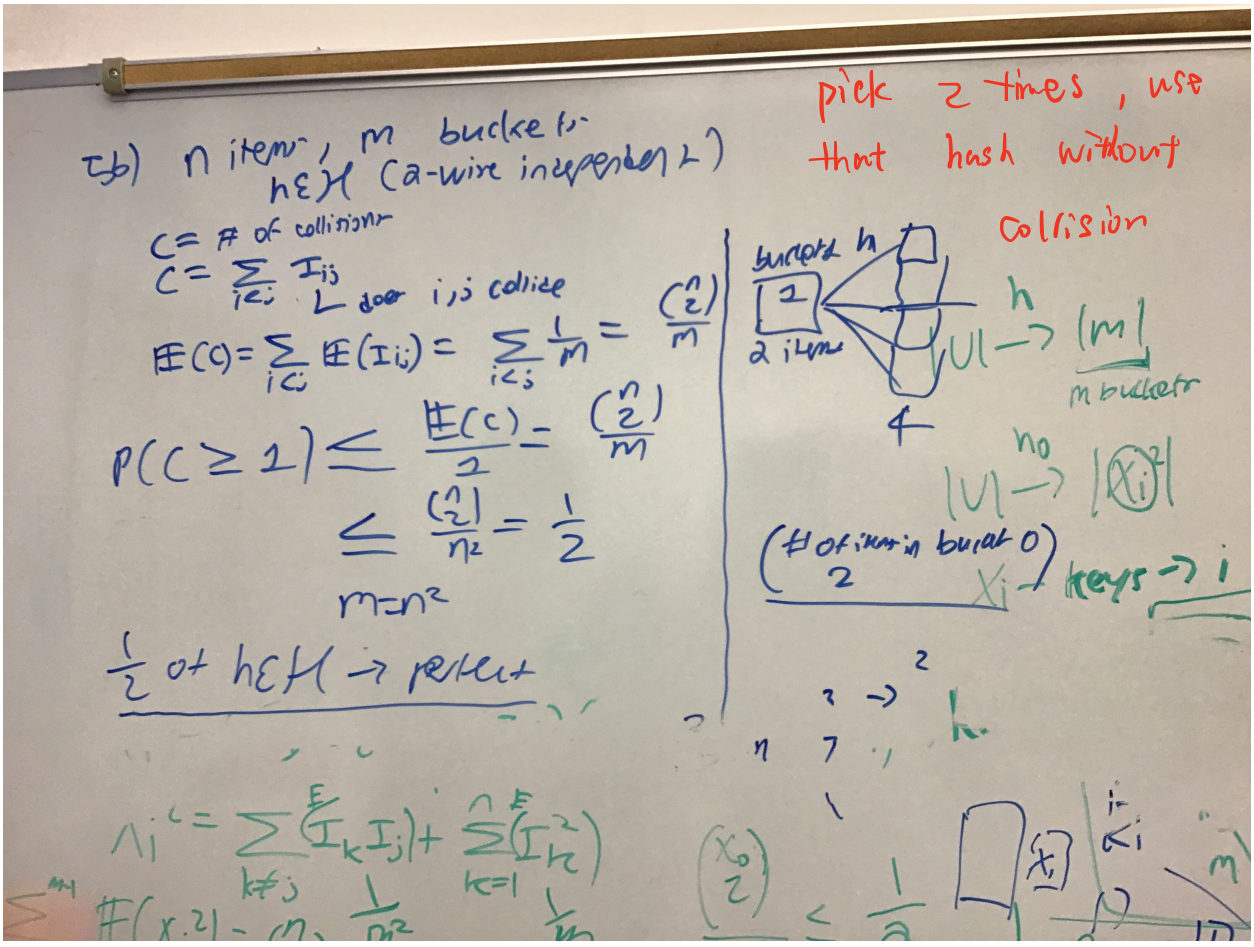
$$\frac{C_n^2}{m^2} + \frac{n}{m^2}$$

$$m\left(\frac{C_n^2}{m^2} + \frac{n}{m^2}\right) = \frac{C_n^2}{m} + \frac{n}{m}$$

$$\frac{n(n-1)}{2} + \frac{n}{m}$$

$$\frac{n}{2} - \frac{1}{2} + 1$$

if $m = n$ $\quad = \frac{n}{2} + \frac{1}{2} = O(n)$


pick 2 times, use that hash without collision
n items, m buckets
h ∈ H (2-wise independent)
C = # of collisions
C = Σ I_ij
I_ij = 1 if i,j collide
E(C) = Σ E(I_ij) = Σ 1/m = (n choose 2)/m
P(C ≥ 1) ≤ E(C)/1 = (n choose 2)/m
≤ (n choose 2)/n^2 = 1/2
m = n^2
1/2 of h ∈ H → perfect
buckets
2 items
h
|U| → |m|
m buckets
4
n_0
|U| → |X_i^2|
(# of items in bucket 0)
2
X_i
keys → i
n → 2 → 2^2
→ 3 → 3^2
→ 4
sum < n^2
better space
only 1 layer
need space n^2
2 layer

5.

(a) $X_i = I_1 + \cdots + I_n$ $\quad I_i$ is the indicator

I_j : $h(j) = i$

$$X_i^2 = (I_1 + \cdots + I_n)^2 = \sum_{i \neq j} I_i I_j + \sum I_i^2$$

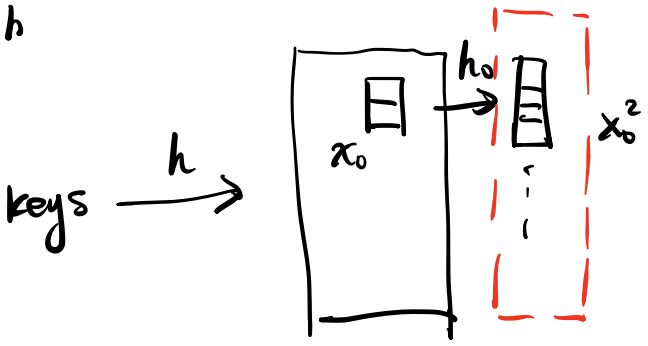
$$= \frac{\binom{n}{2}}{m^2} + \frac{n}{m^2}$$

$$\Rightarrow \quad E[\sum X_i^2] = \sum E[X_i^2] = m\left(\frac{\binom{n}{2}}{m^2} + \frac{n}{m^2}\right)$$

(b) When $m = n$

$$\sum E[X_i^2] = \frac{\binom{n}{2}}{m} + \frac{n}{m} = \frac{n}{2} + \frac{1}{2} = O(n)$$

Thus, we can create the datastructure in a 2 level hash



$\therefore \quad \sum E[k_i^2] = O(n)$

$\therefore$ the red part is $O(n)$ space

(in class, we discussed that we need $O(n^2)$ space to make half of the hash to be perfect hash)

$\therefore$ The whole space is $O(n)$ with $O(1)$ worst query time

The preprocessing is also $O(n)$ because the number of buckets we need to handle is $O(n)$