Data Stream Mining- Lecture 2

Chandresh Kumar Maurya, Assistant Professor, Péter Kiss, Teaching Assistant

Eötvös Loránd University, Budapest, Hungary

October 14, 2020

Approximation

Most of the streaming methods are approximate. (Why?) So, need good approximation techniques such as:

- ϵ-approximation: answer is within some small fraction ϵ of the true answer.
- (ϵ, δ)-approximation: the answer is within $1 \pm \epsilon$ of the true answer with probability $1 - \delta$

Approximation

Most of the streaming methods are approximate. (Why?) So, need good approximation techniques such as:

- ϵ-approximation: answer is within some small fraction ϵ of the true answer.
- (ϵ, δ)-approximation: the answer is within $1 \pm \epsilon$ of the true answer with probability $1 - \delta$

trade-off between the size of summary, and the accuracy, typically space requirements:
$O(\frac{1}{\epsilon^2} log(1/\delta))$

Frequency of elements - Count Min-Skatch

The Problem

Count frequency of A in the stream: A, B, C, A, A, C,...? number of packets from each IP seen at a server

Frequency of elements - Count Min-Skatch

The Problem

Count frequency of A in the stream: A, B, C, A, A, C,...? number of packets from each IP seen at a server

What accuracy we want?

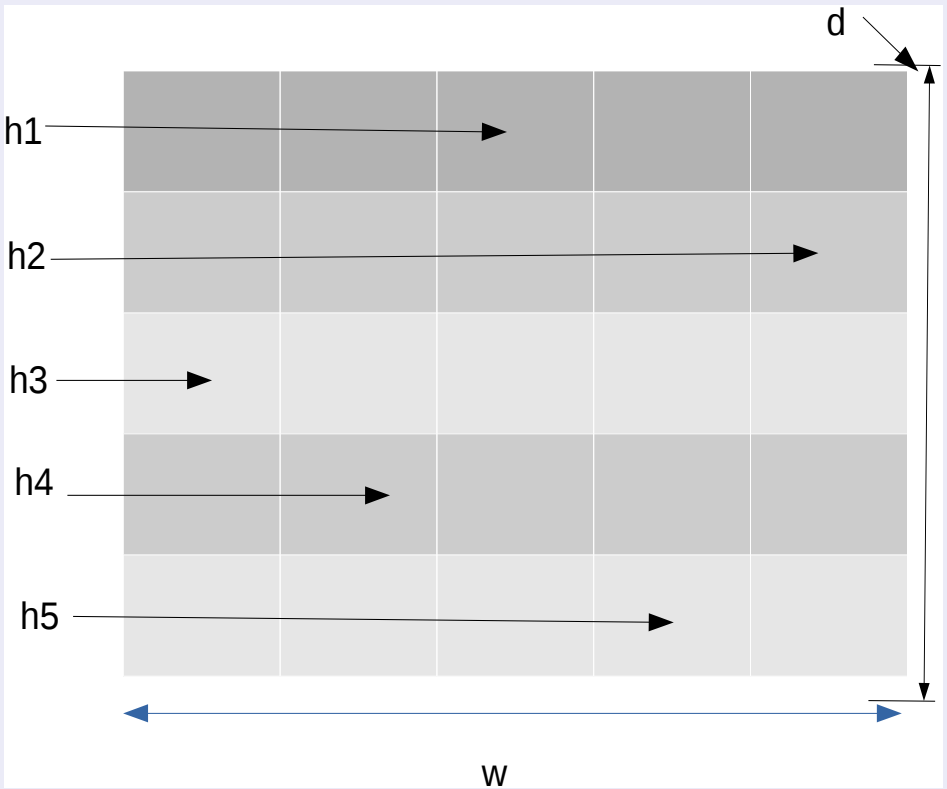
First specify: desired probability level δ, and admissible error ϵ

Sketch Table

create table CM of d rows and w columns, full of 0s
$w = 2/\epsilon$, $d = \lceil \log(1/\delta) \rceil$

Frequency of elements - Count Min-Skatch

MC Table

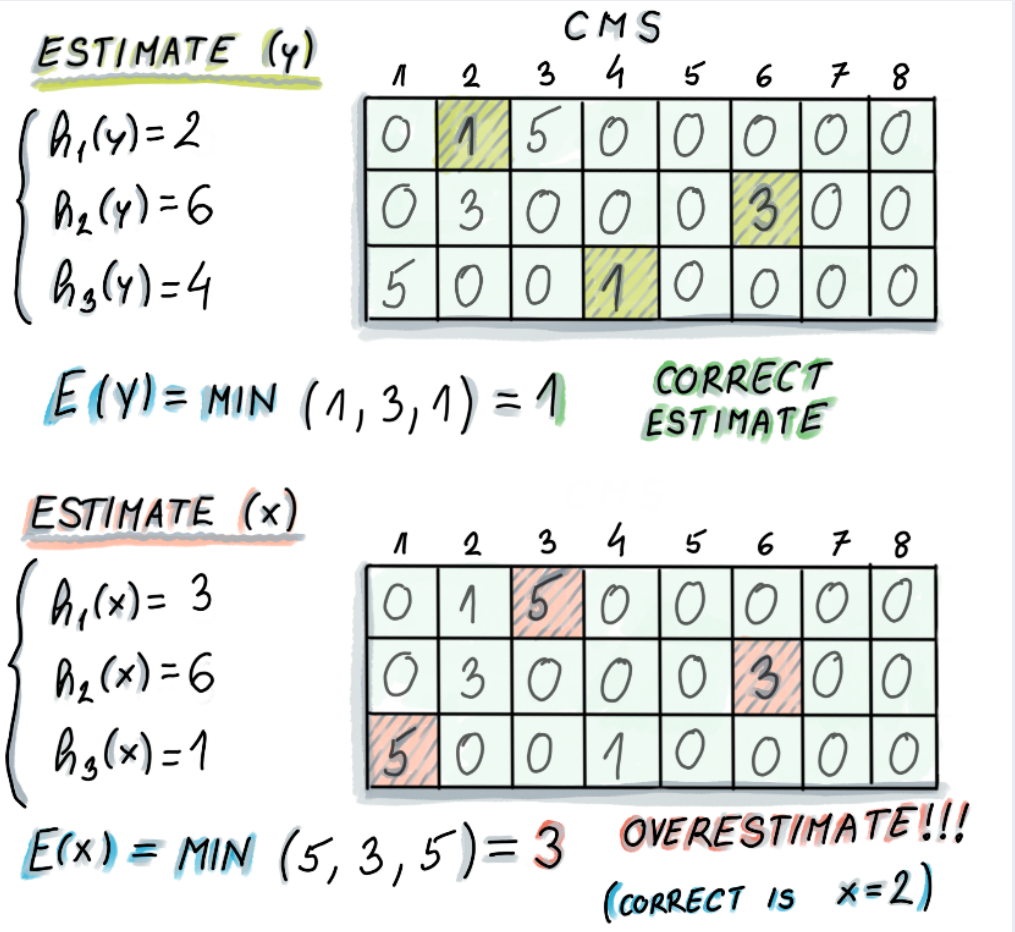


Method

define d $h(\cdot)_i, (i \in 1..d)$ hash function - one corresponding to on row of CM
each entry x of a stream is mapped to one cell per row using the corresponding $h()$, and increase the number of that that cell.

Frequency of elements

Answer



Source: https://livebook.manning.com/book/algorithms-and-data-structures-for-massive-datasets/chapter-4/v-1/29

Question: How many times have we seen the given key x (IP)?
Get the set of cells where some of the h() maps x, and take the minimum:

Unique values in a stream

How many different values have we seen in the stream? $\{0, 1, \ldots, M-1\}$ and stream looks like:

$$0, 1, 0, 0, 0, 1, 1, 2, 3, \ldots$$

Here $M = 3$. Trivial if you have space linear in M (why?). Can you do better? e.g. using space only $\log(M)$. Answer: Use Flajolet and Martin Algorithm (1985).

Flajolet - Martin Algortihm

Algorithm 1: Flajolet and Martin (FM) Algorthm

Input: stream M

Output: Cardinality of M

1 initialization: BITMAP OF L bits initialized to 0.;

2 **for** *each x in M* **do**

1. Calculate hash function $h(x)$.;
2. get binary representation of hash output, call it $bin(h(x))$.;
3. Calculate the index i such that $i = \rho(bin(h(x)))$, where $\rho()$ is such that it outputs the **position** of the least-significant set bit,i.e., position of 1.;
4. set BITMAP[i] $= 1$.;

3 **end**

4 Let R denote the largest index i such that BITMAP[i] $= 1$;

5 Cardinality of M is $2^R/\phi$ where $\phi \approx 0.77351$;

Flajolet and Martin Algortihm - Intuition

1. We have a hash function h that maps x to $h(x)$
2. then take its binary representation of $h(x)$ a bit string
3. The more different element in the stream produces more different hash values. $\rightarrow$ that will end up in the bit string end in some number of 0s *tail length*
4. let R be the maximum tail length of any element seen so far then the estimate for the number of distinct elements 2^R (actually $2^R/\phi$, for$\phi = 0.7735$)
5. Complexity : $O(n)$ for $n = |X|$ time and $O(log(m))$ space complexity

Flajolet and Martin Algortihm

Example 1

Given:

Input Stream= 1,3,2,1,2,3,4,3,1

Let's h(x)= 3X+1mod5

Solve:

Calculation	Reminder	Binary Conversion
h(1)=3(1)+1mod5=	4	100
h(3)=3(3)+1mod5 =	0	000
h(2)=3(2)+1mod5 =	2	010
h(1)=3(1)+1mod5 =	4	100
h(2)=3(2)+1mod5 =	2	010
h(3)=3(3)+1mod5 =	0	000
h(4)=3(4)+1mod5 =	3	011
h(3)=3(3)+1mod5 =	0	000
h(1)=3(1)+1mod5 =	4	100

Trailing Zeros: 2,0,1,2,1,0,0,0,2

r=2

$2^r=2^2=4$

Source:https://www.youtube.com/watch?v=TG48mumSIaw

Attention, 0 counts as 0 trailing zeros!!!

Intuition

The basic idea behind FM algorithm is that of using a hash function that maps the strings in the stream to uniformly generated random integers in the range $[0, \ldots, 2^L - 1]$. Thus we expect that:

- $1/2$ of the numbers will have their binary representation end in 0 (divisible by 2)
- $1/4$ of the numbers will have their binary representation end in 00 (divisible by 4)
- $1/8$ of the numbers will have their binary representation end in 000 (divisible by 8)
- In general, $1/2^R$ of the numbers will have their binary representation end in $[0]^R$

Then the number of unique strings will be approx. 2^R.(because using n bits, we can represent 2^n integers)

Probabilities

- The probability that a given element a has $h(a)$ ending in at least r 0s is $(\frac{1}{2})^r$.
- if there are m distinct element, the probability that NONE of them has a tail length at least r:
  $(1-2^{-r})^m$, where
  $P(\text{an elemnt does not have } r \text{ 0s in tail}) = (1-2^{-r})$
  $((1-2^{-r})^m = ((1-2^{-r})^{2^r})^{m2^{-r}} \approx e^{-m2^{-r}} = \frac{1}{e^{\frac{1}{2^r}m}}$, for large enough r, using $(1-\epsilon)^{\frac{1}{\epsilon}} \approx 1/e$ for large ϵ)
- Thus the probability of not finding r 0s for the true distinct element number m, that is much larger than 2^r: $\frac{1}{e^{\frac{1}{2^r}m}} \to 0$, and subsequently $P(\text{there is a tail at least } r) \to 1$,
- The other way, if $2^r >> m$: $P(\text{there is a tail at least } r) \to 0$
- Summerizing intuition: the more is the number of distinct element m, the bigger the probability of finding a longer tail.
- the proposed approximation is unlikely to be either too much or too low.

Combine multiple estimates

Since the above method is probabilistic how can we be more certain? Combine them! 2^r - our estimate based on the maximal observed tail length r.

The problem that there is always a chance to observe longer tail, than the right one. So if we have four element, there is a probability, that we might face $r = 3$.

Let us run the method 3 times, and assume we get $r_1 = 1, r_2 = 2, r_3 = 3$.

Average: $2^1 + 2^2 + 2^3/3 = 14/3 = 4,66$

For $m = 8$, assume $r_1 = 2, r_2 = 3, r_3 = 4$: the average $2^2 + 2^3 + 2^4 = 28/3 = 9.33$

For $m = 16$, assume $r_1 = 3, r_2 = 4, r_3 = 5$: the average $2^3 + 2^4 + 2^5 = 28/3 = 18.66$

... For these examples: median

Combine multiple estimates

Since the above method is probabilistic how can we be more certain? Combine them! 2^r - our estimate based on the maximal observed tail length r.

The problem that there is always a chance to observe longer tail, than the right one. So if we have four element, there is a probability, that we might face $r = 3$.

Let us run the method 3 times, and assume we get $r_1 = 1, r_2 = 2, r_3 = 3$.

Average: $2^1 + 2^2 + 2^3/3 = 14/3 = 4,66$

For $m = 8$, assume $r_1 = 2, r_2 = 3, r_3 = 4$: the average $2^2 + 2^3 + 2^4 = 28/3 = 9.33$

For $m = 16$, assume $r_1 = 3, r_2 = 4, r_3 = 5$: the average $2^3 + 2^4 + 2^5 = 28/3 = 18.66$

... For these examples: median this can produce only powers of 2

Combine multiple estimates

The way for better estimates: Take small groups of hash functions, and calculate the average, than take the median of those...

Moments in statistics

Moments: quantitative measures related to the shape of the function's graph.
First moment: $\mathbb{E}[X] = \sum_{i=1}^{n} x_i$
Second moment: $\mathbb{E}[X^2] = \sum_{i=1}^{n} x_i^2$
Mean:

$$\mu = \mathbb{E}[X] \tag{1}$$

Variance:

$$\begin{aligned}
\sigma^2 &= \mathbb{E}[(X - \mathbb{E}[X])^2] && (2)\\
&= \mathbb{E}[X^2 - 2X \cdot \mathbb{E}[X] + \mathbb{E}[X]^2] && (3)\\
&= \mathbb{E}[X^2] - 2 \cdot \mathbb{E}[X]\mathbb{E}[X] + \mathbb{E}[X]^2 && (4)\\
&= \mathbb{E}[X^2] - \mathbb{E}[X]^2 && (5)
\end{aligned}$$

Standard deviation:

$$\sigma = \sqrt{\mathbb{E}[X^2] - \mathbb{E}[X]^2} \tag{6}$$

Simple statistics over data streams

- **Mean/Expected value**, with $\overline{x}_t$ denoting the estimate at time t

$$\overline{x}_t = \frac{(t-1)\overline{x}_{t-1} + x_t}{t} \tag{7}$$

for this we need to maintain the actual mean $\overline{x}_t$, and the count of elemets seen i

- **Variance**, with σ_t standing for its approximation at time t:

$$\sigma_t = \sqrt{\frac{\sum_{i=1}^{t} x_i^2 - \frac{(\sum_{i=1}^{t} x_i)^2}{t}}{t-1}} = \sqrt{\frac{\mathbb{E}[X^2] \cdot t - \frac{(t \cdot \mathbb{E}[X])^2}{t}}{t-1}} \tag{8}$$

$$= \sqrt{\frac{t}{t-1}(\mathbb{E}[X^2] - \mathbb{E}[X]^2)} \tag{9}$$

for what we need to store the sum of elements, or the first moment $\sum_{i=1}^{t} x_i = \mathbb{E}[X] \cdot t$ and the sum of squared element or second moments:$\sum_{i=1}^{t} x_i^2 = \mathbb{E}[X^2] \cdot t$

Bessel's Correction

$$\sigma_t = \sqrt{\frac{t}{t-\mathbf{1}}(\mathbb{E}[X^2] - \mathbb{E}[X]^2)}$$

Bessel's Correction: *"minus one to correct for bias..."*

Sample standard deviation is a biased estimator - sample is a subset of a population it intends to represent
The fewer sample the, ones far from the population mean will have a bigger influence on the sample mean, thus the deviation might be underestimated. Taking the square root of the deviation also further decreases it.
The more sample we use the the effect of the correction will vanish as $\frac{t}{t-1} \to 1, t \to \infty$

blog post: `https://towardsdatascience.com/the-reasoning-behind-bessels-correction-n-1-eeea25ec9bc9`

Frequency Moments over streams

Frequency moments

m_i the number of occurrences of the ith element of an ordered universal base set (from which the elements are taken).
kth order moment of DS $\sum_i m_i^k$

0th moment - FM algorithm

number distinct elements $\sum_{i \in X} \underbrace{m_i^0}_{=1}$, if we define $0^0 = 0$

Frequency Moments over streams

Frequency moments

m_i the number of occurrences of the ith element of an ordered universal base set (from which the elements are taken).
kth order moment of DS $\sum_i m_i^k$

0th moment - FM algorithm

number distinct elements $\sum_{i \in X} \underbrace{m_i^0}_{=1}$, if we define $0^0 = 0$

1st moment - CMS algorithm

sum of all m_is, that is the **length of the stream**.

Second moment

2nd moment

surprise number, the higher the number is for the same length, the less even the distribution of the element is.

Example

11 value, length of 100 What is the most even distribution?

Second moment

2nd moment

surprise number, the higher the number is for the same length, the less even the distribution of the element is.

Example

11 value, length of 100 What is the most even distribution?
$9 \times 10 = 90$ $10 \times 1 = 10$ so the surprise factor
$9^2 \times 10 + 10^2 1 = 910$
What is the biggest possible?

Second moment

2nd moment

surprise number, the higher the number is for the same length, the less even the distribution of the element is.

Example

11 value, length of 100 What is the most even distribution?
$9 \times 10 = 90$ $10 \times 1 = 10$ so the surprise factor
$9^2 \times 10 + 10^2 1 = 910$
What is the biggest possible? $1 \times 10 = 10$ and $90 \times 1 = 90$ (one appears 90 times and the others 1 time each) so the surprise factor
$1^2 \times 10 + 90^2 \times 1 = 8110$

Alon-Matias-Szegedy algorithm for second moments

- First assume a fixed length n
- No enough space to store all elements to count all m_is
- Take a number of variables J, and j take a value v_j and a count c_j.
- Pick J random number $i_j \in \{1..n\}$
- When we read stream X, at reaching an i_j, set $v_j = X[i_j]$ and $c_j = 1$ from that point, any time we met v_j in the stream $c_i + +$.
- approximate the second moment from any j: $n(2c_j - 1)$

Why Alon-Matias-Szegedy algorithm does work?

- for $c[i]$ denoting the number of appearance of the element at the ith position ($v[i] = a$), in the range $i, i+1, \ldots, n$
- $\mathbb{E}_n[n(2 \cdot c - 1)] = \frac{1}{n} \sum_{i=1}^{n} n(2 \cdot c[i] - 1) = \sum_{i=1}^{n} (2 \cdot c[i] - 1)$
- grouping together the positions that represent the same element we get
  $\sum_{i=1}^{n} (2 \cdot c[i] - 1) = \sum_a \sum_{i=1}^{n} (2 \cdot c[i] - 1) I(v[i] = a)$:
- for the last position an element appears: $2 \cdot c[i] - 1 = 2 \cdot 1 - 1 = 1$
  for the next to last: $2 \cdot c[i] - 1 = 2 \cdot 2 - 1 = 3$
  and for the next : $2 \cdot c[i] - 1 = 2 \cdot 3 - 1 = 5$
  $\ldots$
  for the very first occurrence $2 \cdot c[i] - 1 = 2(\cdot m_a - 1)$
- and then : $1 + 3 + 5 + 2 \cdots + 2(\cdot m_a - 1) = m_a^2$, thus $\mathbb{E}[n(2 \cdot c - 1)] = \sum_a (m_a)^2$.

Explanation

$$1 + 3 + 5 + 2 \cdots + 2(\cdot m_{v[i]} - 1) = m^2_{v[i]}$$

difference between two squared number always odd:

$k^2 - (k-1)^2 = k^2 - (k^2 - 2k + 1) = 2k - 1$

0	1	4	9	16	25	$\ldots$	$(n-1)^2$	n^2
	1	3	5	7	9	$\ldots$	$2n-1$	

Hence a squared number can be always written as :

$k^2 = \sum_{i=1}^{k}(2i - 1)$

`https://en.wikipedia.org/wiki/Square_pyramidal_number`

$1^2=$	1							
$2^2=$	1	3						
$3^2=$	1	3	5					
$4^2=$	1	3	5	7				
$5^2=$	1	3	5	7	9			
$\vdots$	$\vdots$					$\ddots$		
$(n-1)^2=$	1	$\cdots$				$\cdots$	$2n-3$	
$n^2=$	1	$\cdots$				$\cdots$	$2n-3$	$2n-1$

AMS Example

Example "stream"

Stream $X = a, b, c, b, d, a, c, d, a, b, d, c, a, a, b$
length of the stream $n = |X| = 15$
a: 5 times, b: 4 times, c and d 3 times $\rightarrow 5^2 + 4^2 + 3^2 + 3^2 = 59$

AMS

Let $J = 3$, and the random indices be: $i_1 = 3, i_2 = 8, i_3 = 13$!

Start reading from the beginning!

AMS Example

Example "stream"

Stream $X = a, b, c, b, d, a, c, d, a, b, d, c, a, a, b$
length of the stream $n = |X| = 15$
a: 5 times, b: 4 times, c and d 3 times $\rightarrow 5^2 + 4^2 + 3^2 + 3^2 = 59$

AMS

Let $J = 3$, and the random indices be: $i_1 = 3, i_2 = 8, i_3 = 13$!

Start reading from the beginning!
at index 3: $v_1 = c, c_1 = 1$

AMS Example

Example "stream"

Stream $X = a, b, c, b, d, a, c, d, a, b, d, c, a, a, b$
length of the stream $n = |X| = 15$
a: 5 times, b: 4 times, c and d 3 times $\rightarrow 5^2 + 4^2 + 3^2 + 3^2 = 59$

AMS

Let $J = 3$, and the random indices be: $i_1 = 3, i_2 = 8, i_3 = 13$!

Start reading from the beginning!
at index 3: $v_1 = c, c_1 = 1$
at index 7: $c_1 = 2$

AMS Example

Example "stream"

Stream $X = a, b, c, b, d, a, c, d, a, b, d, c, a, a, b$
length of the stream $n = |X| = 15$
a: 5 times, b: 4 times, c and d 3 times $\rightarrow 5^2 + 4^2 + 3^2 + 3^2 = 59$

AMS

Let $J = 3$, and the random indices be: $i_1 = 3, i_2 = 8, i_3 = 13$!

Start reading from the beginning!
at index 3: $v_1 = c, c_1 = 1$
at index 7: $c_1 = 2$
at index 8: $v_2 = d, c_2 = 1$

AMS Example

Example "stream"

Stream $X = a, b, c, b, d, a, c, d, a, b, d, c, a, a, b$
length of the stream $n = |X| = 15$
a: 5 times, b: 4 times, c and d 3 times $\rightarrow 5^2 + 4^2 + 3^2 + 3^2 = 59$

AMS

Let $J = 3$, and the random indices be: $i_1 = 3, i_2 = 8, i_3 = 13$!

Start reading from the beginning!
at index 3: $v_1 = c, c_1 = 1$
at index 7: $c_1 = 2$
at index 8: $v_2 = d, c_2 = 1$
at index 8: $c_2 = 2$

AMS Example

Example "stream"

Stream $X = a, b, c, b, d, a, c, d, a, b, d, c, a, a, b$
length of the stream $n = |X| = 15$
a: 5 times, b: 4 times, c and d 3 times $\rightarrow 5^2 + 4^2 + 3^2 + 3^2 = 59$

AMS

Let $J = 3$, and the random indices be: $i_1 = 3, i_2 = 8, i_3 = 13$!

Start reading from the beginning!
at index 3: $v_1 = c, c_1 = 1$
at index 7: $c_1 = 2$
at index 8: $v_2 = d, c_2 = 1$
at index 8: $c_2 = 2$
...

at the end: Approximation:
$n(2 \cdot c_i - 1)$
$c_1 = 3 -> 15(2 \cdot 3 - 1) = 75$
$c_2 = 2 -> 15(2 \cdot 2 - 1) = 45$
$c_3 = 2 -> 15(2 \cdot 2 - 1) = 45$
The true value is 59, and if we take the average of the three estimate we get: $\frac{75+45+45}{3} = 55$

AMS for Infinite streams

- How to pick positions i for variables:
- danger of being biased towards the the ones at the "beginning"
- Idea: store as many variables as we can all time, throw some out as the stream grows - being the discarded ones replaced by new ones.
- Let us suppose there is space for s variables.
- inductively: let us assume that we have some n_0 elements, then the positions will be distributed uniformly with $p = s/n_0$. When a new element arrives, with the probability of $s/(n+1)$ pick the new position as variable position, and if we picked throw away uniformly an old one.

Bibliography I