Random Sampling

Chapter 3

Randomized algorithm

- An algorithm is called randomized if its behavior is determined not only by its input but also by the values produced by a random-number generator. (Cormen et al.)

- Example: Randomized QuickSort

A (pseudo) random –number generator is a function Random (a,b) randomly generating a number in the range [a,b].

Sampling

- Sampling is a general technique for tackling massive amount of data.
- Example: To compute the median packet size of some IP packets, we could just sample some and use the median of the sample as an estimate for the true median. Statistical arguments relate the size of the sample to the accuracy of the estimate.

Random Sampling problem

- Problem: Given a sequence of item $S = (i_1, i_2, \ldots, i_n)$ and a positive integer $m \leq n$, the goal is to select a subset of m items uniformly at random.

Random Sampling problem

- Problem: Given a sequence of item $S = (i_1, i_2, \ldots, i_n)$ and a positive integer $m \leq n$, the goal is to select a subset of m items uniformly at random.

- Uniformity: any item in S has to be sampled with probability $1/n$.

Random Sampling problem

- Problem: Given a sequence of item $S = (i_1, i_2, \ldots, i_n)$ and a positive integer $m \leq n$, the goal is to select a subset of m items uniformly at random.

- Uniformity: any item in S has to be sampled with probability $1/n$.

In the 2-level model, we assume $n>M$ (the size of memory), hence the input size n is known a priori and occupies n/B pages.

2-level model and known sequence length

S[1,n] is a file on disks. Size n is known.
S cannot be modified

Algorithm 1

```
1.  Initialize S'[1,n] = S[1,n];
2. For s=0,1, …, m-1 do {
3.      p = Random(1, n-s);
4.      select the item (pointed by) S'[p];
5       swap S'[p] and S'[n-s] ;
}
```

Because of the swap (instruction 5) S'[1, n -s] contains at
Each iteration the items not jet selected.
(S' can be an array of pointers to S's items). Result in S[n-m+1, n]

In any case too much space $\Theta(n \log n)$
and time $\Theta(m)$ and $\Theta(m)$ I/O's operations.

2-level model and known sequence length

S[1,n] is a file on disk. Size n is known.
S cannot be modified

Algorithm 2

1. Initialize dictionary $\mathcal{D}$ = ø
2. **While** $|\mathcal{D}| <$ m **do** {
3. p = Random(1, n);
5 If p is not in $\mathcal{D}$ insert it;
}

Extra space O(m).
time O(m) average time: D implemented as hash table
I/O's operations: min(m, n/B)).

2-level model and known sequence length

Algorithm 2

Extra space $O(m)$

average time $O(m)$: D implemented as hash table the cost of searching, inserting is constant in average.

The cost of re-sampling is also constant in average:

The probability of extracting an already sampled element is $|\mathcal{D}|/n \leq m/n < \frac{1}{2}$.

We can assume $m < n/2$, otherwise we can consider the complement problem that selects the items not in $\mathcal{D}$.

I/O's operations: $\min(m, n/B)$ since $\mathcal{D}$ contains the positions and not the real items that must be collected from S.

2-level model and known sequence length

Substituting the hash table with a balanced search tree changes the time to $O(m \log m)$ worst case.

All the rest is unchanged but at the end $\mathcal{D}$ is sorted and this can help.

Last solution avoids the dictionary and uses sorting as basic block.

(for instance quick-sort built in any programming language)

2-level model and known sequence length

Algorithm 3

1. $\mathcal{D} = \emptyset$
2. **While** ($|\mathcal{D}| <$ m **do**) {
3. $\quad \mathcal{X}$ = randomly select m positions from [1,n];
4. $\quad$ Sort $\mathcal{X}$ and eliminate duplicates;
5. $\quad$ Set $\mathcal{D}$ as the resulting $\mathcal{X}$;
6. }

How many times is repeated the while?

Depends on the number of duplicates.

Compute the probability to do just 1 iteration.

2-level model and known sequence length

Birthday problem:

How many people we must have in the same room in order to have a probability larger than $\frac{1}{2}$ that at least 2 of them have the same birthday?

Here: people=items birthday=position in S

same birthday=duplicate

Compute the probability of the event:

A duplicate among m randomly chosen items does not occur (m people does not have the some birthday).

2-level model and known sequence length

Probability that a duplicate among m randomly chosen items does not occur:

$$\frac{m!\binom{n}{m}}{n^m} = \frac{n(n-1)(n-2)\cdots(n-m+1)}{n^m} = 1 \times \left(1 - \frac{1}{n}\right) \times \left(1 - \frac{2}{n}\right) \times \cdots \left(1 - \frac{m-1}{n}\right)$$

Since $e^x \geq 1 + x$, we can upper bound the formula as:

$$e^0 \times e^{-1/n} \times e^{-2/n} \times \cdots e^{-(m-1)/n} = e^{-(1+2+\cdots+(m-1))/n} = e^{-m(m-1)/2n}$$

If $m = \sqrt{n}$ the probability of a duplicate is around 0.4

If $m << \sqrt{n}$ e^x can be approximate with $1 + x$ *then* $e^{-m(m-1)/2n}$ is a reasonable estimate of the number of re-sampling.

2-level model and known sequence length

Algorithm 3

Extra space O(m)

An average constant number of sorting steps:

O(mlogm) average time if m< M. (otherwise an external memory sorted must be employed).

I/O's operations: min(m, n/B)

For Sorting can be used also Bucket Sort. The M elements are random integers in [1,n]. We use m slots of n/m elements. Item i_j is stored in bucket jm/n . Each bucket contains O(1) items in average since the m items are uniformly sampled.

So we have average time: O(m).

Data Stream Model

- Stream: *m* elements form universe of size n, e. g.,

$$(x_1, x_2, \ldots, x_m) = 3,5,3,7,5,4, \ldots$$

Data Stream Model

- Stream: m elements form universe of size n, e. g.,

$$(x_1, x_2, \ldots, x_m) = 3,5,3,7,5,4, \ldots$$

- Goal: Compute a function of stream, e.g. the median, number of distinct elements, longest increasing sequence.

Data Stream Model

- Stream: m elements form universe of size n, e. g.,

$$(x_1, x_2, \ldots, x_m) = 3,5,3,7,5,4, \ldots$$

- Goal: Compute a function of stream, e.g. the median, number of distinct elements, longest increasing sequence.
- Features
  1. Limited working memory, sublinear in n and m

Data Stream Model

- Stream: *m* elements form universe of size n, e. g.,

$$(x_1, x_2, \ldots, x_m) = 3,5,3,7,5,4, \ldots$$

- Goal: Compute a function of stream, e.g. the median, number of distinct elements, longest increasing sequence.
- Features
  1. Limited working memory, sublinear in *n* and *m*
  2. Access data sequentially

Data Stream Model

- Stream: m elements form universe of size n, e. g.,

$$(x_1, x_2, \ldots, x_m) = 3,5,3,7,5,4, \ldots$$

- Goal: Compute a function of stream, e.g. the median, number of distinct elements, longest increasing sequence.
- Features
  1. Limited working memory, sublinear in n and m
  2. Access data sequentially
  3. Process each data quickly

Data Stream Model

- Stream: m elements form universe of size n, e. g.,

$$(x_1, x_2, \ldots, x_m) = 3,5,3,7,5,4, \ldots$$

- Goal: Compute a function of stream, e.g. the median, number of distinct elements, longest increasing sequence.
- Features
  1. Limited working memory, sublinear in n and m
  2. Access data sequentially
  3. Process each data quickly

Origins in the 70s but become popular in the last 10 years because of growing theory and very applicable.

Why's it become popular?

- Practical Appeal:
  - Faster networks, cheaper data storage, ubiquitous data logging result in massive data to be processed.
  - Applications to network monitoring, query planning, I/O efficiency for massive data, sensor networks aggregation….
- Theoretical Appeal:
  - Easy to state problems but hard to solve.
  - Links to communication complexity, compressed sensing, embeddings, pseudo-random generators, approximation…

Streaming model and known sequence length

- No pre-processing is possible
- Every item is considered once, so the algorithm must immediately decide whether or not include the item in the set.
- Algorithms are simple but the probabilistic analysis more involved.
- Why is different from the disk model?

  The Alg. must decide immediately whether or not the item must be included or not in the solution.
- We want
  - A uniform sample from range [1,n]
  - the sample size equal to m

Streaming model and known sequence length

- Consider first the case m=1 (select one item from S)
- The alg. selects S[j] with probability $\mathcal{P}$(j) properly defined: P(1)=1/n, P(2)= 1/(n-1), P(3)= 1/(n-2), ecc. ecc.
- Item j is selected with probability P(j) =1/(n-j+1);
- If it occurs it stops. Eventually item n is selected with prob. 1

How to draw an item with probability p? Draw a random real r in [0,1] and then compare it against p. If r≤p select the item.

Streaming model and known sequence length

- The condition m=1 is guaranteed
- We must show that the probability of sampling S[j] is 1/n independently of j:
- n-j+1 is the number of the remaining elements in the sequence and every one of them must be sampled uniformely .
- By induction.
- The first j-1 items have probability 1/n each to be sampled then the probability of not sampling anyone of them is 1-(j-1)/n. It results that the probability to sample i_j is

$$1-(j-1)/n \times 1/(n-j+1)=1/n$$

Streaming model and known sequence length

Case m ≥ 1

Algorithm 4

```
1.  s = 0;
2.  for ( j=1; j≤n; j++) {
3.      p = Random [0,1];
4.      if (p ≤ m-s/(n-j+1)) {
5.          select S[j];
6.           s++
7.       }
8.  }
```

$\mathcal{P}(j)$ is now set to m-s/(n-j+1), s is number of elements already selected before S[j].

Streaming model and known sequence length

$\mathcal{P}(j)$ is now set to m-s/(n-j+1), s is number of elements already selected before S[j].

$\mathcal{P}(j)$ is formulated in this way because equals the probability P that S[j] is included in a random sample of size m-s taken from S[j,n] of n-j+1 items.

$$P = \frac{\text{number of ways of selecting m-s elements including S[j]} *}{\text{all ways of selecting m-s elements}}$$

$$\frac{\binom{n-j}{m-s-1}}{\binom{n-j+1}{m-s}} = \frac{m-s}{n-j+1}$$

$$\text{recall } \binom{b}{a} = \frac{b!}{a!\,(b-a)!}$$

- number of ways of selecting m-s elements including S[j] equals the number of waysof m-s-1 items taken from S[j+1,n] of n-j items with item S[j].

Streaming model and known sequence length

Algorithm 4 takes O(n/B) I/Os into 2-level model
O(n) time, n random samples (random generation)
O(m) space.

The space is needed to store the m sampled items, since the stream flows away.

It is possible solve the problem with only $m<n$ random samples!

Idea : generate random jumps instead of indices.
(How many elements to skip over before selecting the next item).

Simple algorithm : exponential complexity

Complex algorithm: linear time

Streaming model and unknown sequence length

N unknown ; different strategies
First solution: Heap H of size m and a number generator RANDOM (0,1); H is a min-heap containing the items with max r values

Algorithm5

```
1.  Initialize the heap H with m dummy pairs (-∞, 0);
2.  for each item S[j] {
3.      r_j = Random (0,1);
4.      m = minimum key in H;
5.      if (r_j > m) {
5.          extract from H the minimum key;
6.          Insert (rj, S[j])  in H;
7.        }
8.  }
9.  Return H;
```

Streaming model and unknown sequence length

Algorithm 5 associates a random key to each item and maintains in the heap the top key m items

Takes O(1) to detect min H. O(logm) to extract and Insert in H,

Algorithm 5 takes O(n/B) I/Os, O(nlogm) time, generates n random number and O(m) space.

The next solution is the so called reservoir sampling (Knuth 1997), better than Algorithm 5 both in time and space.

Reservoir array R[1,m] to keep the candidate sample.
At each step:

- S[j] is included in R with prob. $\mathcal{P}(j)$ =m/j
- an item at random is kicked out from R with prob. 1/m

Streaming model and unknown sequence length

Algorithm 6: Reservoir Sampling

```
1.  Initialize array R[1,m] = S[1,m];
2.  for each next item S[j] {
3.      h = Random (1,j);
4.      if (h ≤ m) {
5.          set R[h] =S[j];
6.      }
7.  }
8.  Return  R;
```

Since h is chosen between 1 and j the prob. that h is ≤ m is m/j what we wished to set for P(j).

Streaming model and unknown sequence length

Analysis Reservoir Sampling

Since h is chosen between 1 and j, the prob. ($h \leq m$) = m/j as we wished to set for P(j).
If S[j] is chosen, an item is kicked out from R with prob=1/m since the item is selected at random.

Theorem:The m selected items are drawn uniformly at Random from S, that is with prob. m/n.

Proof. By induction.
Base: n=m hence every item must be selected with prob m/m=1. This is true by step 1 of the algorithm.

Streaming model and unknown sequence length

Analysis Reservoir Sampling

Inductive step: n>m ; n-1 → n

1. True for S[n]: this item is inserted in R with prob. m/n (step 4 of alg.).

2. Consider the selection of the items S[1,…,n-1]. Any item has prob m/(n-1) to be selected by inductive hypothesis. It stays in R if
   - S[n] is selected and the item it is not kicked out.
   - S[n] is not selected or

   S[n] is not selected with prob 1-m/n. Hence

   $$\frac{m}{n} \times \frac{m-1}{m} + 1 - \frac{m}{n} = \frac{n-1}{n}$$

3. Overall an item belongs to R iff
   - it was in R at step n-1
   - it was not kicked out at step n

   $m/(n-1) \times n-1/n = m/n$

Streaming model and unknown sequence length

Analysis Reservoir Sampling

Algorithm 6 takes O(n/B) I/Os, O(n) time, n random number and exactly m space.

Hence it is time, space and I/Os optimal for the stream model of computation