Big Data Algorithms Exam May 2021

Prof. John Iacono

1.
- Read these instructions.
- Exam has 7 pages. Make sure you have all of them.
- Write your name on the top of each page.
- You may bring any books or notes without limit. No electronic devices of any kind.
- Exam is from 8AM-11AM.
- You may use any fact or algorithm form the class without having to re-describe it, never copy anything from the notes: just refer to it.
- Except when asked to do so specifically, you need not code. You can just say "Run the BLAH algorithm on BLAH with parameters BLAH and then ...." I am trying to find out if you understand which algorithms to use, how they work, how you choose their parameters, and what their runtimes are.
- If something is not clear, explain what is not clear and any assumptions you may make to make it clear.
- You need not work out all math, but feel free to approximate when it is easy to do so, e.g. $\log_2 1000 \approx 10$.
- If you need more room, use the back of the page.
- All questions are weighted equally, but have very different difficulties.

Do you understand these instructions?

2. Suppose you want to store a large collection of photographs, each with a time stamp, and you wish to allow the following:

   - Insert a photograph with a given timestamp. You can assume the timestamp is provided to the millisecond: e.g. an integer with the number of milliseconds since Jan 1, 2000.
   - Given an exact time to the millisecond, return if there was a photograph taken at the time.
   - Given a time, return if there was a photograph taken within 1 minute of that time.

   (a) For this question (only), please provide code based on the following framework:

```
class PhotoCollection:
    def __init__(self):
    def insert(self,photo,time):
    def photosAtExactTime(self,time):
    def photosWithinOneMinute(self,time):
```

   Answer:

```
class PhotoCollection:
    def __init__(self):
      self._A={}
    def insert(self,photo,time):
      self._A[time]=true
    def photosAtExactTime(self,time):
      return (time in self._A)
    def photosWithinOneMinute(self,time):
      return any( (t in self._A) for t in range(time-60000,time+60000))
```

   (b) What is the runtime of each of the three operations as a function of the number of photographs currently stored, n.

   Answer: $O(1)$

   (c) Is your `photosWithinOneMinute` faster/slower/same speed as `photosAtExactTime`? Explain.

   Answer: Slower by a factor of 120,000.

3. Suppose you are a credit card company. Sometimes you accept contactless payments, and sometimes you require a PIN. You want this decision to be as fast as possible. You have a list of bad credit card numbers, and you want to certainly require a PIN for them, and you want most of the people not on the bad list to not have to type in a PIN. This data structure will be stored on a mobile device and thus must be as small as possible.

   (a) What structure from the class should be used?

   Answer: Bloom fillter

   (b) Suppose the list of bad credit card has 1 million credit card numbers. Each credit card number has 16 digits (Thus there are 10^{16} possible numbers). Suppose you are willing to have people not on the BAD list be forced to type in the pin 10% of the time. How big should the data structure be (in kilobytes)? Clearly indicate what parameters you choose.

   Answer: $-\frac{\ln \epsilon}{\ln^2 2} \approx$ 5 bits per item, approx 5 million bits, approx 600 kilobytes.

   (c) With the structure you have described, is it easy to insert and/or delete card numbers from the list of bad credit card numbers?

   Answer: Insertion works, deletion is not possible in a standard Bloom filter.

   a

4. Suppose there are 10 different vaccine centers, and you have 10000 people that you need to assign to a center, by a number of different means such as apps, telephone, and mail. Suppose rather than trying to coordinate the assignments you simply decide to assign each person randomly to a center when they make an appointment. How well will this work? For the following questions give bounds using the techniques described in class, answering "at most" or "at least" as appropriate.

   (a) What is the expected number of persons per center? Answer: 1000

   (b) What is the chance that the first center has more than 2000 people?
   Answer: Chernoff with $E[X] = 1000$ from previous question, set $\delta = 1$, thus

$$\Pr[X \geq (1+1)1000] \leq e^{-\frac{1^2 1000}{2+1}} \approx \frac{1}{10^{145}}$$

   (c) What is the chance that at least one center has more than 2000 people?
   Answer: From union bound, at most 10 times that of the previous question. Thus $\leq \frac{1}{10^{144}}$

   (d) What is the chance that all centers have less than 2000 people?
   Answer: 1—the previous answer, thus at lest $1 - \frac{1}{10^{144}}$

   (e) If someone criticizes the random method of allocation, and says that you are likely to end up with centers that are horribly unbalanced, would you agree or disagree?

Answer: Disagree. $1 - \frac{1}{10^{144}}$ is very very close to 1.

5. Suppose you have a count-min sketch data structure with width 50 and height 10, and where one million items have been added. Suppose you query the structure and ask how many times x has appeared for some x that was not one of the million items added. What do you expect the structure to return?

   Answer: I did not count this as no one got it right and it was harder than expected.

6. In class we used minhash to turn a set of strings into scores so that by looking at the scores we could determine closeness according to the Jaquard distance.

   Given two strings x and y let $s(x)$ and $s(y)$ be the score of x and y. We showed that $\Pr[s(x) = s(y)]$ is exactly the Jaquard distance.

   (a) Thus, what should the ???'s be in the following code:

   ```
   def jaquard(x,y,s):
       if s(x)==s(y):
           return ???
       else:
           return ???
   ```

   Answer: 1, 0

   (b) Why would using the above code to estimate the Jaquard distance be a bad idea?

   Answer: It only gives 1 and 0!

   (c) We used multiple scores/minhashes in order to estimate Jaquard distance between documents. In deciding how many scores/minhashes to use, what does this decision depend upon ? (please circle all those that apply):

   - The length of the documents
   - The number of documents
   - The desired approximation Answer: This
   - The desired failure probability Answer: This
   - Anything else?

7. Suppose you have a data stream of names, and you want to know for each name when it comes in if you have seen this exact element before. John has the following idea: use the distinct elements sketch. Pass every item from the stream to the distinct elements sketch, and if the number of distinct elements reported increases, then say you have a duplicate, and otherwise say you don't.

   (a) Does this work? Why/why not?

   Answer: It does not work as the sketch only returns estimates that are powers of two and thus adding a new item is not likely to change the estimate.

   (b) If it works: how big does the sketch need to be?

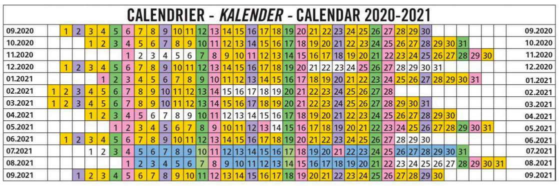
   (c) If not, how would you solve this problem using as little space as possible if

      i. you are willing to tolerate a 1% error rate and

      ii. you are not willing to have any errors. For both of these, say precisely the amount of space needed.

Answer: For (i), a Bloom filter. $-\frac{\ln \frac{1}{100}}{\ln^2 2} \approx 10$ bits, per name approximately 1.2 bytes per name. Answer: For (ii) a hash table aka a python list. You need the space to store all the names, roughly 20 bytes per name.

8. If you look at the schedules of STIB, each day has one of seven colors:



CALENDRIER - *KALENDER* - CALENDAR 2020-2021

09.2020		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30							09.2020
10.2020				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				10.2020
11.2020							1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30		11.2020
12.2020		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31						12.2020
01.2021					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			01.2021
02.2021	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28										02.2021
03.2021	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31							03.2021
04.2021				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30					04.2021
05.2021						1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		05.2021
06.2021		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30							06.2021
07.2021				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				07.2021
08.2021							1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	08.2021
09.2021			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30						09.2021

Someone has split the days into seven groups so that the days in each group people behave in a "similar" way and thus have similar transportation needs. Perhaps this was done by hand trying to taking into account things such as weekdays and weekends. But now we have data! Specifically, suppose you had the GPS data of all Belgians for every day of the year: you have for every minute of every day for 2020 the GPS location (latitude and longitude) of every one of 10 million Belgians. Given this data how would you go about classifying the days of 2020 into seven groups?

(a) What algorithm(s) would you use? Answer: Dimensionality reduction (JL) and k-means.

(b) For (each) algorithm what would be the input and output, the runtime, and choice of parameters? Make reasonable assumptions as needed.

Answer: Let $n = 10^7$ and $d = 266 * 24 * 60 * 2$. For JL, map to $k = c \ln 10^7$ dimensions, where c depends on the error and failure rate. This takes time $O(ndk) = O(nd \log n)$. Then run k-medians on the compressed data which takes time $O(nkx)$, where x is the number of rounds to converge, which is usually small.

(c) Suppose you had the option to get user's position every second instead of every minute. Obviously this would increase the data set by a factor of 60. How much would it change the runtime of your method?

Answer: This would cause JL to run 60 times slower, but would not affect the k-means.