6.S079 Quiz 2 Practice Problems

Q1: Amdahl's Law

Suppose you have a data science pipeline with 3 operators that take 5s, 10s, and 15s to execute, respectively. According to Amdahl's law, what is the maximum speedup you can achieve by optimizing the 2nd operator?

A. 25%
B. 33%
C. 50%
D. 66%
E. 100%

Q1: Amdahl's Law (Solution)

Suppose you have a data science pipeline with 3 operators that take 5s, 10s, and 15s to execute, respectively. According to Amdahl's law, what is the maximum speedup you can achieve by optimizing the 2nd operator?

A. 25%
B. 33%: 10/(5 + 10 + 15) = 1/3
C. 50%
D. 66%
E. 100%

Q2: Z-Order

Suppose you have 2D data items (1,1), (1,7), (2,2) , and (4,5). Put them in Z-order. Order items in ascending order, starting with the least significant bit of the first dimension.

Q2: Z-Order (Solution)

Suppose you have 2D data items (1,1), (1,7), (2,2) , and (4,5). Put them in Z-order.
Order items in ascending order, starting with the least significant bit of the first dimension.

Represent the 4 items as binary

(001, 001) -> 000011 -> 3

(001, 111) -> 101011 -> 43

(010, 010) -> 001100 -> 12

(100, 101) -> 110010 -> 50

So the Z-ordering is (1,1), (2,2), (1,7), (4,5)

Q3: Parallel Joins

Suppose you are joining two tables, S and T, on attributes S.a and T.b, which are partitioned across 3 nodes. If S is 1 million records and T is 1 billion records, both tables are randomly ordered and partitioned, and your network can transmit 1 millions records per second, which is the best strategy for computing this join:

A. Compute a shuffle join, repartitioning both tables across the 3 nodes on the join attribute
B. Have each node read all of the partitions of S to join with its partition of T
C. Have each node read all of the partitions of T to join with its partition of S
D. Have one node read all of the partitions of S and T and compute the join locally.

Q3: Parallel Joins (Solution)

Suppose you are joining two tables, S and T, on attributes S.a and T.b, which are partitioned across 3 nodes. If S is 1 million records and T is 1 billion records, both tables are randomly ordered and partitioned, and your network can transmit 1 millions records per second, which is the best strategy for computing this join:

B. Have each node read all of the partitions of S to join with its partition of T

   S is small so will be less expensive to read on all nodes than shuffle both tables

Q4: Parallel Speedup

Consider the following Dask code to merge two data frames.

```
client = Client(n_workers=X, threads_per_worker=1, memory_limit='16GB')

df = dask.dataframe.read_csv('f1.csv')
df2 = dask.dataframe.read_csv('f2.csv'

total = df2.merge(df1,on='joinattr').count()
print(total)
```

Suppose you run this on a 16 core machine, varying X over [1,2,4,8,16]. Averaging over 100 runs for each value of X, you find that the performance is as follows:

X=1	2	4	8	16
20 s	10 s	5 s	5 s	9 s

Q4: Parallel Speedup (con't):

X=1	2	4	8	16
20 s	10 s	5 s	5 s	9 s

Which of the following are possible explanations for this behavior:

A. As X becomes large, the system is running out of memory, and spilling data to disk or restarting some tasks
B. It takes several seconds to start up tasks, limiting scalability
C. It takes several seconds to perform the count() operation on a single worker, limiting scalability
D. One of the data frames contains only 4 distinct values

Q4: Parallel Speedup (Solution):

X=1	2	4	8	16
20 s	10 s	5 s	5 s	9 s

Which of the following are possible explanations for this behavior:

A. As X becomes large, the system is running out of memory, and spilling data to disk or restarting some tasks
D. One of the data frames contains only 4 distinct values

Startup or count() times don't make sense because we see perfect speedup through X=4

Q5: Bootstrap

Ben Bitdiddle claims the following code produces an accurate confidence interval on the value of the maximum of the function `f` applied the array `data`. Is he correct? Why or why not?

```
import numpy.random as r

#data is a large array of data

#r.choice returns a random sample of the data
d = r.choice(data, size=100, replace=False)

#f is some expensive function that we don't want to run on all the data
# so we just run it on a sample
d = list(map(f,d))

mxs = []
for i in range(0,100):
    s = r.choice(d, size=100, replace=True)
    mxs.append(max(s))
mxs.sort()

print(f"mean is {max(d)}, w/ +/- 5% confidence interval {mxs[4],mxs[-5]}")
```

Q5: Bootstrap (Solution)

Ben Bitdiddle claims the following code produces an accurate confidence interval on the value of the maximum of the function `f` applied the array `data`. Is he correct? Why or why not? False, because bootstrap doesn't work for max()

```
import numpy.random as r

#data is a large array of data

#r.choice returns a random sample of the data
d = r.choice(data, size=100, replace=False)

#f is some expensive function that we don't want to run on all the data
# so we just run it on a sample
d = list(map(f,d))

mxs = []
for i in range(0,100):
    s = r.choice(d, size=100, replace=True)
    mxs.append(max(s))
mxs.sort()

print(f"mean is {max(d)}, w/ +/- 5% confidence interval {mxs[4],mxs[-5]}")
```

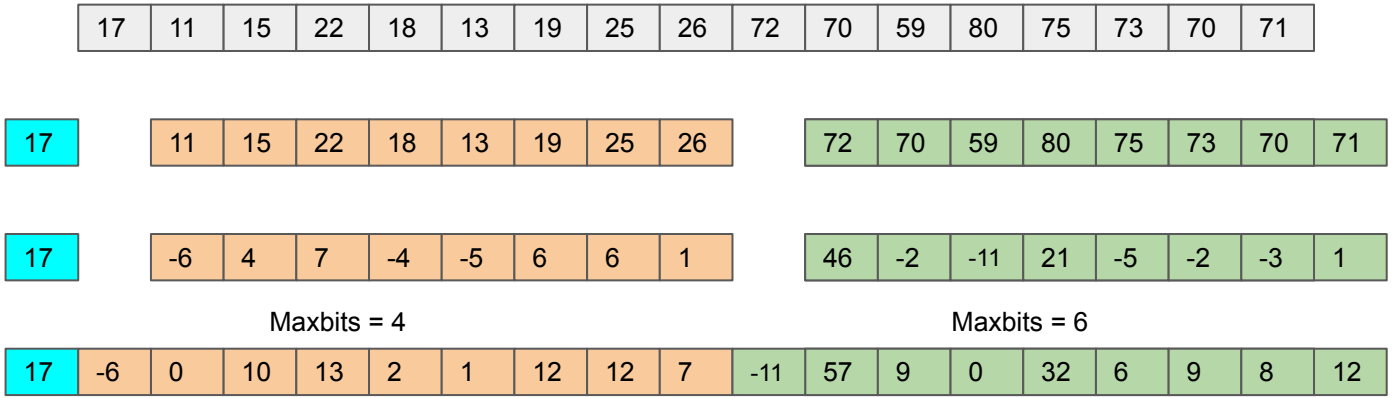
Q6: Delta Encoding & Parquet

How large would the Parquet encoding of the following array be, in bytes? Assume each integer is 8 bytes long (both in the original input and in Parquet for the reference, min delta and per-block “bits” value) and Parquet Blocks are 64 bytes each.

17	11	15	22	18	13	19	25	26	72	70	59	80	75	73	70	71

Q6: Delta Encoding & Parquet (Solution)

How large would the Parquet encoding of the following array be, in bytes? Assume each integer is 8 bytes long (both in the original input and in Parquet for the reference, min delta and per-block “bits” value) and Parquet Blocks are 64 bytes each.



8 (ref) + 8 (min) + 8 (bits) + 4 (data) + 8 (min) + 8 (bits) + 6 (data) = 50 bytes
Original input was 17 * 8 = 136 bytes

Q7: Lying With Statistics

The sales team recently presented this chart, to support the claim that our company has consistently been outperforming competitors. Find 5 problems with their visualization.

Q7: Lying With Statistics (Solution)

The sales team recently presented this chart, to support the claim that our company has consistently been outperforming competitors. Find 5 problems with their visualization.

1. Missing years
2. Different scale of vertical axes
3. Left vertical axis doesn't start at 0
4. Vertical axes not even mapped to series
5. Doesn't adjust for inflation.
6. Average is useless without knowing # of orders
7. "Average" is vague

Q8: Multi-Hypothesis Testing I

You are trying to help a car manufacturer make their cars safer by analyzing some accident data. For that you received a dataset with 10,000 records and 20 binary features describing the car (e.g, engine type), and 5 binary features per record describing what type of accident the car was involved in.

230mile battery	v4	v6	v6-turbo	red	…	Driver injured	Passenger injured	Fire
1	0	0	0	1	…	1	0	1
0	1	0	0	1	…	1	1	0
0	1	0	0	0	…	0	1	0
0	0	1	0	0	…	0	0	0

Question 1) Assuming we use traditional p-tests to test if each feature significantly impacts each accident type, what alpha threshold per test do you have to use to control *the chance of at least one type I error* at 5%?
Question 2) Assume instead we opt to use an FDR of 5%. Would we find (a) more or (b) fewer correlations? Under which conditions would we find the same amount of correlations?

Q8: Multi-Hypothesis Testing I (Solution)

You are trying to help a car manufacturer make their cars safer by analyzing some accident data. For that you received a dataset with 10,000 records and 20 binary features describing the car (e.g, engine type), and 5 binary features per record describing what type of accident the car was involved in.

230mile battery	v4	v6	v6-turbo	red	…	Driver injured	Passenger injured	Fire
1	0	0	0	1	…	1	0	1
0	1	0	0	1	…	1	1	0
0	1	0	0	0	…	0	1	0
0	0	1	0	0	…	0	0	0

Question 1) Assuming we use traditional p-tests to test if each feature significantly impacts each accident type, what alpha threshold per test do you have to use to control *the chance of at least one type I error* at 5%?

We have $0.05 = 1 - (1- \alpha)^{100} \Rightarrow \alpha = 0.0005$

Q8: Multi-Hypothesis Testing I (Solution)

You are trying to help a car manufacturer make their cars safer by analyzing some accident data. For that you received a dataset with 10,000 records and 20 binary features describing the car (e.g, engine type), and 5 binary features per record describing what type of accident the car was involved in.

Question 2) Assume instead we opt to use an FDR of 5%. Would we find (a) more or (b) fewer correlations? Under which conditions would we find the same amount of correlations?

For such an FDR, the thresholds in the BH procedure would be of the form k * 0.05 / 100 = 0.0005 *k. Compared to having a fixed alpha = 0.0005, this is more permissive, so we'd generally find at least as many correlations. We would find the same amount if every p value that BH admits is also smaller than 0.0005 - equivalently, if the smallest p-value to exceed its BH threshold is also the smallest p-value greater than 0.0005.

Q9: Multi-Hypothesis Testing II

Let’s assume we calculate the following (sorted) p-values:

1. (battery, fire): p=0.0003
2. (red, fender bender): p=0.0004
3. (v6, passenger injured): p=0.001
4. (v6, driver injured): p=0.0015
5. (v6, fire): p=0.004
6. ….

Question 1) With the Bonferroni correction for which tests would you reject the null hypothesis (total alpha = 0.05)

Question 2) Under FDR control (BH procedure) with alpha 0.05 for which tests would you reject the null hypothesis

Questions 3) Describe the difference between the Family-Wise Error Rate and FDR.

Q9: Multi-Hypothesis Testing II (Solution)

Let’s assume we calculate the following (sorted) p-values:

1. (battery, fire): p=0.0003
2. (red, fender bender): p=0.0004
3. (v6, passenger injured): p=0.001
4. (v6, driver injured): p=0.0015
5. (v6, fire): p=0.004
6. ….

Question 1) With the Bonferroni correction for which tests would you reject the null hypothesis (total alpha = 0.05)

We have 20 descriptive features and 5 target variables, so a total of k=100 tests. After the Bonferroni correction, this gives an alpha of 0.05 / 100 = 0.0005. We therefore reject the null hypothesis only for 1. and 2.

Q9: Multi-Hypothesis Testing II (Solution)

Let's assume we calculate the following (sorted) p-values:

1. (battery, fire): p=0.0003
2. (red, fender bender): p=0.0004
3. (v6, passenger injured): p=0.001
4. (v6, driver injured): p=0.0015
5. (v6, fire): p=0.004
6. ….

Question 2) Under FDR control (BH procedure) with alpha 0.05 for which tests would you reject the null hypothesis

k	p-value	FDR threshold	Reject Null Hypothesis?
1	0.0003	1 * 0.05 / 100 = 0.0005	Yes
2	0.0004	2 * 0.05 / 100 = 0.001	Yes
3	0.001	3 * 0.05 / 100 = 0.0015	Yes
4	0.0015	4 * 0.05 / 100 = 0.002	Yes
5	0.004	5 * 0.05 / 100 = 0.0025	No

Q9: Multi-Hypothesis Testing II (Solution)

Let's assume we calculate the following (sorted) p-values:

1. (battery, fire): p=0.0003
2. (red, fender bender): p=0.0004
3. (v6, passenger injured): p=0.001
4. (v6, driver injured): p=0.0015
5. (v6, fire): p=0.004
6. ….

Questions 3) Describe the difference between the Family-Wise Error Rate and FDR.

The family-wise error rate is the **probability** of a type I error (i.e. rejecting the null hypothesis when it's true, or equivalently predicting an effect when there is none - a "false positive") **in at least one experiment**.

On the other hand, the FDR is the **expected ratio** of the number of type I errors to the total number of positive classifications (rejections of the null hypothesis).

Q10: Columns Vs Rows

You have a table T of 100,000,000 records, each of which has 10 columns (C0 - C9) of 1 KB each. C0 is a unique ID ranging from 0 to 99,999,999, but records are ordered randomly. The table is stored on an SSD with a peak sequential read bandwidth of 3000 MB/s and a peak random read bandwidth of 300 MB/s, when reading 1 KB blocks with a single thread. Assume that you always achieve the peak sequential read performance for the current block whenever the previous 1 KB you read was stored right before the current one, and you always achieve the peak random read performance otherwise.

The only type of query you care about is the following:

SELECT C0, C1, C2 FROM T WHERE C0 BETWEEN x AND (x + 99,999)

for some uniformly random integer 0 <= x <= 99,900,000. Assuming you have a zero-latency index to identify which blocks to read, how long does the I/O for the above query take if…

a) T is laid out as a row store (i.e. all the columns of the first record, then all the columns of the second record etc.) ?
b) T is laid out as a column store? What explains this performance difference?
c) T is laid out in the optimal way? What is the optimal layout?

Q10: Columns Vs Rows (Solution)

You have a table T of 100,000,000 records, each of which has 10 columns (C0 - C9) of 1 KB each. C0 is a unique ID ranging from 0 to 99,999,999, but records are ordered randomly. The table is stored on an SSD with a peak sequential read bandwidth of 3000 MB/s and a peak random read bandwidth of 300 MB/s, when reading 1 KB blocks with a single thread. Assume that you always achieve the peak sequential read performance for the current block whenever the previous 1 KB you read was stored right before the current one, and you always achieve the peak random read performance otherwise.

The only type of query you care about is the following:

SELECT C0, C1, C2 FROM T WHERE C0 BETWEEN x AND (x + 99,999)

for some uniformly random integer 0 <= x <= 99,900,000. Assuming you have a zero-latency index to identify which blocks to read, how long does the I/O for the above query take if…

a) T is laid out as a row store (i.e all the columns of the first record, then all the columns of the second record etc.) ?

The order of records is random, but we are reading 3 contiguous columns per record. For each record, we read the first 1 KB at 300 MB/s and the following 2 KB at 3000 MB/s. Therefore, overall we read 1 KB * 100,000 = 100 MB at 300 MB/s, taking 0.33 s, and 2 * 1 KB * 100,000 = 200 MB at 3000 MB/s, taking 0.067 s. Overall, we need 0.4 s.

Q10: Columns Vs Rows (Solution)

You have a table T of 100,000,000 records, each of which has 10 columns (C0 - C9) of 1 KB each. C0 is a unique ID ranging from 0 to 99,999,999, but records are ordered randomly. The table is stored on an SSD with a peak sequential read bandwidth of 3000 MB/s and a peak random read bandwidth of 300 MB/s, when reading 1 KB blocks with a single thread. Assume that you always achieve the peak sequential read performance for the current block whenever the previous 1 KB you read was stored right before the current one, and you always achieve the peak random read performance otherwise.

The only type of query you care about is the following:

SELECT C0, C1, C2 FROM T WHERE C0 BETWEEN x AND (x + 99,999)

for some uniformly random integer 0 <= x <= 99,900,000. Assuming you have a zero-latency index to identify which blocks to read, how long does the I/O for the above query take if…

b) T is laid out as a column store? What explains this performance difference?

The order of records is still random, but now the 3 columns we need per record are also not contiguous. Thus, every column of every record incurs a random read. Overall we read 3 * 1 KB * 100,000 = 300 MB at 300 MB/s, taking 1 s.

Q10: Columns Vs Rows (Solution)

You have a table T of 100,000,000 records, each of which has 10 columns (C0 - C9) of 1 KB each. C0 is a unique ID ranging from 0 to 99,999,999, but records are ordered randomly. The table is stored on an SSD with a peak sequential read bandwidth of 3000 MB/s and a peak random read bandwidth of 300 MB/s, when reading 1 KB blocks with a single thread. Assume that you always achieve the peak sequential read performance for the current block whenever the previous 1 KB you read was stored right before the current one, and you always achieve the peak random read performance otherwise.

The only type of query you care about is the following:

SELECT C0, C1, C2 FROM T WHERE C0 BETWEEN x AND (x + 99,999)

for some uniformly random integer 0 <= x <= 99,900,000. Assuming you have a zero-latency index to identify which blocks to read, how long does the I/O for the above query take if…

c) T is laid out in the optimal way? What is the optimal layout?

The optimal layout is to sort the records and then use a column store. Going from column to column is “random”, but we are reading 100,000 contiguous records per column. For each column, we read the first item at 300 MB/s and the following ones at 3000 MB/s. Therefore, overall we read 1 KB * 3 = 3 KB at 300 MB/s, taking 10 µs, and 3 * 99,999 * 1 KB ~ 300 MB at 3000 MB/s, taking 0.1 s. Overall, we need ~0.1 s.

Q11: Speed-up vs Scale-up

You want to calculate the value of $g(x) = f^{100}(x)$ for some expensive function f(x), for 8 different values of x. Unsatisfied with the performance of your 8-core computer, you are considering upgrading to a 64-core machine (of the same exact technology) instead. What is the most likely outcome? The size of the problem here is the number of values you are finding g(x) of.

a) A significant speedup, but no significant scaleup.
b) A significant scaleup, but no significant speedup.
c) A significant scaleup and a significant speedup.
d) Neither - you should buy a newer machine.

Q11: Speed-up vs Scale-up (Solution)

You want to calculate the value of $g(x) = f^{100}(x)$ for some expensive function f(x), for 8 different values of x. Unsatisfied with the performance of your 8-core computer, you are considering upgrading to a 64-core machine (of the same exact technology) instead. What is the most likely outcome? The size of the problem here is the number of values you are finding g(x) of.

b) A significant scaleup, but no significant speedup.

The nature of your task is such that parallelizing the processing of each value is impossible. Therefore, there will be no speedup from dedicating more cores to this task. Having more cores will contribute to scaleup, though, allowing more values to be processed in parallel.

Q12: Compression Techniques

Which compression technique is likely to produce a very compact representation for the following types of data? Assume that you can compress parts of each dataset differently, if needed.

a) A history of your COVID Pass test results, where each record indicates one of the strings “POSITIVE”/”NEGATIVE”/”INCONCLUSIVE”
b) A time series of the EUR/USD exchange rate.
c) A column in a database of New York city residents, indicating the tail number of their private plane, if they own one.
d) An unsorted column indicating the owner (using their username as a string) of each file on your research lab’s server.

Q12: Compression Techniques (Solution)

Which compression technique is likely to produce a very compact representation for the following types of data? Assume that you can compress parts of each dataset differently, if needed.

a) A history of your COVID Pass test results, where each record indicates one of the strings “POSITIVE”/”NEGATIVE”/”INCONCLUSIVE”

   Dictionary encoding (because the strings are long-ish) and run-length encoding (because most tests are negative) would be reasonable choices, ideally in combination.

b) A time series of the EUR/USD exchange rate.

   Delta encoding, since this rate generally exhibits small fluctuations around a somewhat fixed value.

Q12: Compression Techniques (Solution)

Which compression technique is likely to produce a very compact representation for the following types of data? Assume that you can compress parts of each dataset differently, if needed.

c) A column in a database of New York city residents, indicating the tail number of their private plane, if they own one.

This column will contain an overwhelming amount of NULL values, so just storing a list of non-NULL tuples would be most efficient.

d) An unsorted column indicating the owner (using their username as a string) of each file on your research lab's server.

Dictionary encoding would work best, since users are likely to be only a handful, compared to the potential large number of files.

Q13: Bayes Factor

You are in charge of a toll station with two booths, each with its own queue. Incoming vehicles always line up for the shortest (in terms of absolute length) queue on arrival, and remain on that queue until processed. Processing each vehicle is approximately equally difficult, but variations arise in practice because of random factors (e.g. someone looking for change, a longer vehicle taking up more space in the queue). However, allegations have been received that booth 1 is twice as slow as booth 2. You decide to collect some data.

At the end of a day, you measure that booth 1 processed 45 vehicles, while booth 2 processed 55 vehicles. You start worrying - booth 2 processed more than 20% more vehicles!

If you go ahead and calculate the Bayes factor, what is the interpretation regarding the hypothesis that booth 1 is indeed twice as slow?

Q13: Bayes Factor (Solution)

Two hypotheses:

- In H_0, the booths process vehicles at the same rate, so that each vehicle has p = ½ of being processed by booth 1.
- In H_1, booth 1 is twice as slow, so that each vehicle has p = ⅓ of being processed by booth 1 (that queue tends to be longer).

For the Bayes factor, we have:

- P(45 | H_0) = (100 choose 45) * (½)^45 * (½)^55
- P(45 | H_1) = (100 choose 45) * (⅓)^45 * (⅔)^55
- BF_01 = P(45 | H_0) / P(45 | H_1) = (3/2)^45 * (¾)^55 ≅ 11.28

This is in fact considered strong evidence in favor of H_0.