AI interview questions

1 Algorithms

1.1 Q: Merge k (k = 2) arrays and sort them

The best way to do this would be to sort two arrays and then merge them. Complexity of that algorithm is:

$$O(nlogn + mlogm + (n + m))$$

The Python code for this solution is:

```
# Function to merge array in sorted order
def sortedMerge(a, b, res, n, m):
    # Sorting a[] and b[]
    a.sort()
    b.sort()

    # Merge two sorted arrays into res[]
    i, j, k = 0, 0, 0
    while (i < n and j < m):
        if (a[i] <= b[j]):
            res[k] = a[i]
            i += 1
            k += 1
        else:
            res[k] = b[j]
            j += 1
            k += 1

    while (i < n):  # Merging remaining
                    # elements of a[] (if any)
        res[k] = a[i]
        i += 1
        k += 1

    while (j < m):  # Merging remaining
                    # elements of b[] (if any)
        res[k] = b[j]
        j += 1
        k += 1
```

1.2 Can you explain what MapReduce is and how it works?

1.3 Find the maximum of sub sequence in an integer list.

1.4 Find max sum subsequence from a sequence of values.

1.5 Create a function that checks if a word is a palindrome.

2 Linear algebra

2.1 How to measure distance between data point?

2.2 Q: How to compute an inverse matrix faster by playing around with some computational tricks?

Let's say we have a 3x3 matrix like this:

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

First we compute the determinant of a matrix. If the result is 0, the work is done $\rightarrow$ matrix has no inverse.

$$det(M) = a \cdot (e \cdot i - h \cdot f) - b \cdot (d \cdot i - g \cdot f) + c \cdot (d \cdot h - g \cdot e)$$

The next step is to transpose the matrix and then iterate through the matrix and at each step select one element. For that element eliminate matrix row and column where that element is located and compute determinant for the "minor" 2x2 matrix.

For example: we pick element a with coordinates $i = 1, j = 1$. So we eliminate 1st row and 1st column and we are left with 2x2 matrix:

$$m = \begin{bmatrix} e & f \\ h & i \end{bmatrix}$$

The determinant of that matrix is:

$$det(m) = d \cdot i - h \cdot f$$

We repeat that for every element of the matrix. The last step is to change the sign of every 2nd element of a matrix, or mathematically, we change the sign of all elements whose sum of row and column index is odd (here we assume that row and column indices start from 1, if they start from 0 then we have to do the opposite.)

3 Probability and statistics

3.1 Three friends in Seattle told you it's rainy. Each has probability of 1/3 of lying. What's the probability of Seattle is rainy?

3.2 Discuss how to randomly select a sample from a product user population.

3.3 Q: Define variance.

Variance measures how far a dataset is spread out. The technical definition is "*The average of the squared differences from the mean*", but all it really does is to give you a very general idea of the spread of your data. A value of zero means that there is no variability, in fact, all the numbers in the dataset are the same.

The variance (σ^2) is a measure of how far each value in the data set is from the mean. Here is how it is defined: Subtract the mean from each value in the data. This gives you a measure of the distance of each value from the mean. Square each of these distances (so that they are all positive values), and add all of the squares together. Divide the sum of the squares by the number of values in the data set. The standard deviation (σ) is simply the (positive) square root of the variance.

$$\sigma^2 = \frac{\sum_{i=1}^{N} (X - \mu)^2}{N}$$

where X is the **mean value** and μ is **current example value**

3.4 How do you find percentile? Write the code for it.

The most common definition of a *percentile* is a number where a certain percentage of scores fall below that number.

$$Percentile = (number of people behind you / total number of people) * 100$$

```
def percentile(p, values):
    r = round((p/100)*len(values))
    range_of_percentile = values[:r]
    percentile_value = values[r]
    return(range_of_percentile, percentile_value)

if __name__ == '__main__':
    values = list(range(100))
    print(percentile(30, values))
```

```
>>> ([0, 1, 2, 3, ..., 27, 28, 29], 30)
```

4 Machine learning

4.1 Can you explain the fundamentals of Naive Bayes? How do you set the threshold?

4.2 Can you explain SVM?

4.3 How do you detect if a new observation is outlier? What is a bias-variance trade off ?

4.4 Describe the working of gradient boost.

4.5 How to deal with unbalanced binary classification?

4.6 How do you solve the L2-regularized regression problem?

4.7 Q: What are the different regularization metrics L1 and L2?

L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

The last part of equation that is multiplied by λ is the L1-regularization part.

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2 + \lambda \sum_{i=1}^{k} {w_i}^2$$

The last part of equation that is multiplied by λ is the L2-regularization part.

L1-regularization	**L2-regularization**
Computationaly inefficient on non-sparse cases	Computationaly efficient due to having analytical solutions
Sparse outputs	Non-sparse outputs
Built-in feature selection	No feature selection

Properties:

- **Built-in feature selection** is frequently mentioned as a useful property of the L1-norm, which the L2-norm does not. This is actually a result of the L1-norm, which tends to produces sparse coefficients (explained below). Suppose the model have 100 coefficients but only 10 of them have

non-zero coefficients, this is effectively saying that the other 90 predictors are useless in predicting the target values. L2-norm produces non-sparse coefficients, so does not have this property.

- **Sparsity** refers to that only very few entries in a matrix (or vector) is non-zero. L1-norm has the property of producing many coefficients with zero values or very small values with few large coefficients.
- **Computational efficiency.** L1-norm does not have an analytical solution, but L2-norm does. This allows the L2-norm solutions to be calculated computationally efficiently. However, L1-norm solutions does have the sparsity properties which allows it to be used along with sparse algorithms, which makes the calculation more computationally efficient.

5 Conceptual questions

5.1 Q: How do you implement autocomplete?

5.2 What would you do to summarize a twitter feed?

5.3 How to perform a series of calculations without a calculator. Explain the logic behind the steps.

6 Preprocessing

6.1 Explain the steps for data wrangling and cleaning before applying machine learning algorithms.

6.2 What is the difference between box plot and histogram?

6.3 Q: What is a difference between good and bad Data Visualization?

Good	**Bad**
Simple	Complicated
Easy to read, clean	Confusing colors, too much text
One property of data per graph	Multiple properties of data
Color consistency	Color inconsistency

Good and bad data visualization properties