Subject: Request for Alternate Final Exam Arrangement Due to Travel Conflict Dear Professor [Professor's Last Name], I hope this message finds you well. I am writing to bring to your attention a conflict that has arisen regarding the final exam scheduled for December 16th from 6:00 PM to 8:00 PM. Unfortunately, I've encountered an unavoidable circumstance that requires my immediate attention. I have a pre-booked flight home on the same day, departing at 3:00 PM, due to an urgent family matter that demands my presence. This unexpected situation has left me in a predicament as it coincides with the final exam timing. I take my academic responsibilities seriously and would never wish to create any inconvenience. However, considering the circumstances, I wanted to reach out and humbly request your understanding and assistance in exploring potential alternatives or accommodations for taking the final exam. I am more than willing to discuss and accommodate any alternative arrangements that may be feasible to ensure I fulfill my academic obligations. I understand the importance of this final exam and assure you of my commitment to maintaining the academic integrity of the course. Your guidance and support in this matter would be immensely appreciated. Thank you very much for your understanding and consideration. I am available to discuss this further at your earliest convenience. Sincerely, [Your Name] [Your Student ID]1. \*\*Introduction (0:00 - 0:30)\*\* Fisher-Yates Shuffle

- Briefly introduce the significance of shuffling in card games.

- Emphasize the need for a reliable and efficient shuffling algorithm.

2. \*\*Overview of Fisher-Yates Shuffle (0:30 - 1:00)\*\*

- Introduce the Fisher-Yates Shuffle as a simple and effective algorithm for shuffling cards.

- Highlight its importance in providing randomness and maintaining an equal probability for each card.

3. \*\*Algorithm Explanation (1:00 - 2:30)\*\*

- Break down the Fisher-Yates Shuffle algorithm into steps.

- Emphasize the simplicity and randomness achieved by iteratively selecting and swapping cards.

- Explain how the algorithm guarantees an equal probability for each card's final position.

4. \*\*Running Time Analysis (2:30 - 4:00)\*\*

- Discuss the time complexity of the Fisher-Yates Shuffle.

- Emphasize that each card is considered once during the shuffling process, resulting in a linear time complexity of O(n), where n is the number of cards in the deck.

- Compare this linear time complexity to other, less efficient shuffling methods.

5. \*\*Conclusion and Applications (4:00 - 5:00)\*\*

- Summarize the key points: simplicity, effectiveness, and linear time complexity.

- Highlight the wide applicability of the Fisher-Yates Shuffle in various scenarios requiring randomization.

- Conclude by reinforcing the importance of understanding shuffling algorithms, especially for applications with time constraints.

Script:

**Introduction:**

So far in C.S.E 3500 we’ve only discussed how to sort a list through using methods such as Quicksort and Mergesort. However we have yet to discuss how we do the reverse, reshuffling a list into a random order.

The need for a reliable and efficient shuffling algorithm is extremely important in the creation of games, as it forms the foundation for introducing unpredictability, fairness, and excitement into gameplay.

The Fisher-Yates Shuffle Algorithm is an elegant solution that fulfills this crucial requirement in the creation of games.

**Overview: 2 mins**

The Fisher-Yates Shuffle algorithm is a reliable and efficient algorithm used for randomly shuffling elements in an array. It works by iteratively swapping each element in the array with a randomly chosen element that comes after it.

To begin we have a sorted array of five elements:



We start at the last element of the array, and generate a random index representing an element of the array.



In this case our random index is 3 meaning that C is the element that we will be swapping with E.



Once the swap has occurred we move onto the next element



C is now closed the new swapped element is D and a random element now excluding the last element



In this case the random element is A



And the next swap occurs



Now A is closed off E becomes the new element to swap

Another random index is chosen between the last elements

this time the random element is E as a result you could say E now swaps with E however nothing happens aside for E now being closed and window continues to shift to the last 2 elements

For the last swap of the shuffle B is swapped with the random element of D, and now the array is completely shuffled



**Digital Showcase of Algorithm**

[**https://www.youtube.com/watch?v=TdOUjGfv1Gs**](https://www.youtube.com/watch?v=TdOUjGfv1Gs) **2:06 with Cards**

**Start at the End of the Array: Begin the shuffle by starting at the last element of the array.**

**Randomly Select an Index: Generate a random index between 0 and the current position in the array, inclusive.**

**Swap Elements: Swap the element at the current position with the element at the randomly chosen index.**

**Move to the Previous Position: Move to the previous position in the array and repeat steps 2 and 3 until you reach the first element.**

**Python Line by Line**

[**https://www.youtube.com/watch?v=TdOUjGfv1Gs**](https://www.youtube.com/watch?v=TdOUjGfv1Gs) **2:06 with exact copy**

**Implementation:**

The following is the python implementation of the fisher-yates shuffling algorithm

The first line is the implementation of the function with the parameter, the array that you would like to shuffle

The second line is a for loop that iterates through the array backwards

The third line is the random index generator that generates a value between 0 and the current element in the array

Finally the fourth line is swapping which rotates the element at the current index with the element at the randomly generated index

These steps continue for each of the remaining elements in the array

**Time Complexity:**

The time complexity of the Fisher-Yates shuffle is O(n) linear time complexity. The reason for this linear time complexity is that each element in the array is visited once, and at each visit, a constant amount of work where an element is selected and swapped with another element in the array is performed. This linear scalability is a desirable trait, especially when dealing with large datasets, as it guarantees a predictable and manageable increase in execution time as the input size grows.

Here is a break down why this results in a linear time complexity:

1. \*\*Iterative Process\*\*: The loop runs 'n' times, ensuring that each element in the array is considered exactly once. This means that regardless of the array size, the algorithm will perform a set number of operations directly proportional to the number of elements.

2. \*\*Constant Work per Iteration\*\*: Within each iteration of the loop, the algorithm performs a constant amount of work, involving generating a random index and swapping elements. This work doesn't depend on the total number of elements but remains consistent throughout the loop.

3. \*\*No Nested Loops or Recursion\*\*: The algorithm doesn't involve nested loops or recursive calls that might exponentially increase the number of operations based on the input size. Each element is processed once within a single loop.

As a result of these factors, the time taken by the Fisher-Yates shuffle scales linearly with the number of elements in the array. For every additional element, one more iteration through the loop occurs, ensuring that the time complexity grows in a linear fashion—O(n).

**Conclusion:**

In conclusion, the Fisher-Yates shuffle stands as a simple yet highly effective algorithm for randomizing the order of elements in an array. With a time complexity of O(n), the algorithm ensures a uniform distribution of permutations, making it a reliable choice for applications such as game development, statistical simulations, and cryptographic protocols. Its straightforward implementation and ability to provide true randomness contribute to its enduring popularity in diverse fields where unpredictability and fairness are essential.