Weekly Challenge 02: Average Case

CS/CE 412/471 Algorithms: Design and Analysis

Spring 2025

1. Expectations

In the insertion sort algorithm given on Page 19 of our textbook (CLRS), the different *cases* of the algorithm are distinguished by the number of times, L, that the body of the `while` loop on lines 5 to 7 executes. We are going to measure this quantity, L, to estimate the best, worst, and average running times of the algorithm.

Task

Write a program that

- contains a modified insertion sort algorithm which returns L,
- iterates a variable, n, over the values $1, 2, 3, \ldots, 11$,
- for each n,
  - generates all permutations, p, of the sequence $\langle 1, 2, 3, \ldots, n \rangle$,
  - for each p, computes and stores the value of L,
  - records the smallest, largest, and average value of L, corresponding to the best, worst, and average running times.
- plots the the best, worst, and average running times of the algorithm for each of the values of n.

Your program will take an increasing amount of time as n increases, and may take a long time to produce the result for $n = 11$. Think why that is the case. You are welcome to compute and plot values for $n > 11$.

Submission

You will submit your `py` file and an image file containing the required plot.

Hint

- Use `range` to iterate over values of n.
- Use `itertools.permutations` to generate permutations.
- You may use the `matplotlib` module for plotting.

```
import itertools
import matplotlib.pyplot as plt

def new_insertion_sort(A):
    L = 0
    n = len(A)
    for i in range(1, n):
        key = A[i]
        # Insert A[i] into the sorted subarray A[1 : i - 1]
        j = i - 1
        while j >= 0 and A[j] > key:
            A[j + 1] = A[j]
            j = j - 1
            L = L + 1
        A[j + 1] = key
    return L

def generate_permutations(n):
    lst = list(itertools.permutations(range(1, n + 1)))
    print("The number of permutations are: ", len(lst))
    return lst

def plot_results(n_values, best_case, worst_case, average_case):
    plt.figure(figsize=(10, 6))
    plt.plot(n_values, best_case, label='Best Case', color='mediumseagreen')
    plt.plot(n_values, worst_case, label='Worst Case', color='maroon')
    plt.plot(n_values, average_case, label='Average Case', color='steelblue')
    plt.title('Running Times of Modified Insertion Sort')
    plt.xlabel('n (Size of Input)')
    plt.ylabel('L (Number of While-Loop Iterations)')
    plt.legend()
    plt.grid(True)
    plt.savefig('insertion_sort_analysis.png')
    plt.show()

def main():
    n_values = range(1, 12)  # iterating a variable n from 1 - 11
    # n_values = range(1, 3) # for small testing
    best_case = []
    worst_case = []
    average_case = []

    for n in n_values:
        permutations = generate_permutations(n) # generates all permutations, p
        L_list = []

        # for each p, computes and stores the value of L
        for p in permutations:
            L = new_insertion_sort(list(p))
            L_list.append(L)

        best_case.append(min(L_list))
        worst_case.append(max(L_list))
        average_case.append(sum(L_list) / len(L_list))

    plot_results(n_values, best_case, worst_case, average_case)

if __name__ == "__main__":
    main()
```

"""
References
Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms (4th ed.). The MIT Press. Page 19

"""