Homework 11

Juan Camilo Velasquez and Joaquin Rodriguez

Part 1: In the present assignment, we generated an interface to the searchutils.f90 Fortran module previously created in assignment 9, and we used f2py to generate a Python module. F2py is a NumPy's tool that provides an easy connection between Python and Fortran languages. In this case, we are using Fortran 90. After generating the python module searchUtilsTeam08.so, we created two Python files to test the linear search and binary search functions, and then we measured the CPU times of the Fortran search algorithm using NumPy's searchsorted and argwhere.

The performance_eval.py code is shown in Figure 1.

```python
# The value to search for: the second to last element in the array
idx_search = large_array[-2]

# CPU time for the Fortran Linear Search algorithm
t_start = time.time()
idx_linear = search.searchutils.linearsearch(large_array, idx_search)
t_end = time.time()
print(f"Fortran linearSearch CPU time: {t_end - t_start} seconds")

# CPU time for the Fortran Binary Search algorithm
t_start = time.time()
idx_binary = search.searchutils.binarysearch(large_array, idx_search)
t_end = time.time()
print(f"Fortran binarySearch CPU time: {t_end - t_start} seconds")

# Evaluate the CPU time for numpy's searchsorted
t_start = time.time()
idx_np_searchsorted = np.searchsorted(large_array, idx_search)
t_end = time.time()
print(f"NumPy searchsorted CPU time: {t_end - t_start} seconds")

# Evaluate the CPU time for numpy's argwhere
t_start = time.time()
idx_np_argwhere = np.argwhere(large_array == idx_search)[0][0]
t_end = time.time()
print(f"NumPy argwhere CPU time: {t_end - t_start} seconds")

# Verify the results
print(f"Index found by linearSearch: {idx_linear}")
print(f"Index found by binarySearch: {idx_binary}")
print(f"Index found by NumPy searchsorted: {idx_np_searchsorted}")
print(f"Index found by NumPy where: {idx_np_argwhere}")
```

Figure 1. LinearSeach, BinarySearch, searchsorted, and argwhere search functions.

After confirming that the linearsearch() and binarysearch() functions work correctly of the f2py's Python module, the performance_eval.py code was used to evaluate the CPU times of search algorithm calls on an array with 10 million unique elements, and used linspace to create an array of elements between 10 and -10. The index in test was (x[-2]), second to last element in the array.

For more information, the searchsorted() function is used to find indices into a sorted array, such that if elements are inserted before the indices, the order will be preserved. The argwhere() finds indices of array elements that are non-zero.

The runtimes for the two initial Fortran functions and the NumPy algorithms are shown in Table 1:

Implementation	CPU Time (sec)	Index Found
linearsearch	0.0057525634765625	9999999
binarysearch	4.76837158203125e-06	9999999
Numpy searchsorted	2.8848648071289062e-05	9999998
Numpy argwhere	0.009307861328125	9999998

Discussion and Conclusion: From the CPU performance times, the binary search implementation was the fastest among all algorithms with 4.76 µs, demonstrating efficiency for finding elements in sorted arrays. The Numpy searchsorted ranked the second fastest with 28.8 µs, then the linearsearch with 5.75 ms, and finally the argwhere function with 9.3 ms. All implementations where successful in locating the desired index.

Based on these results, we conclude that for applications requiring rapid searching in sorted arrays, the binary search offers the most efficient solution, by outperforming the other alternatives. Not to mention that the NumPy's argwhere is the least optimized option for large-scale arrays.

We have learned that choosing the appropriate search method depends on requirements of the application, such as data size and sorting conditions.