Sorting

Ethan Hawk

May 3, 2023

Questions

0.1 C code using the Arr_T struct that goes in main

```c
int main(int argc, char *argv[]) {
  int arr_size;
  if (argc > 1) {
    if (sscanf(argv[1], "%i", &arr_size) != 1) {
      fprintf(stderr, "[ERR] - Not an integer!");
      return 1;
    }
  } else {
    arr_size = 100;
  }
  Arr_T M = make_Arr(arr_size);
  populate_Arr(M);
  printf("Array before sorting: \n");
  print_Arr(M);
  combSort(M);
  printf("Array after sorting: \n");
  print_Arr(M);
  return 0;
}
```

0.1.1 Download these files to your machine if you are completing this lab in C: array.h array.c

0.2 Python code using the Arr_T class in main

```
if __name__ == "__main__":
    array = array.Arr_T()
    array.make_Arr(100)
    array.populate_Arr()
    print("Array before sorting: ")
    array.print_Arr()
    bucket_sort(array)
    print("Array after sorting: ")
    array.print_Arr()
```

0.2.1 Download this file to your machine if you are completing this lab in Python: array.py

0.3 Implement 4 of the following sorting algorithms of your choosing in either C or Python. Note the big O of each.

- bucket
- cocktail
- comb
- gnome
- insertion
- quick
- radix
- shell

0.4 Which of the sorting algorithms that you implemented was fastest? Which was the easiest to implement (and why)?

0.5 Why would someone choose to use bubblesort over quicksort? Explain, and give an example usecase.

0.6 Why do computer scientists care so much about sorting? Why does it often *need* to be fast?