This lecture talks about selection sort for the second homework assignment.

Sorting means ordering things in ascending or descending orders. This book, the art of computer programming volume three by Donald K nuth, has a lot of excellent materials about sorting. I encourage you to check out this book.

Sorting is used widely in our everyday life. Imagine that you want to find airplane tickets to Paris. Travel websites have options to short by the prices, travel time, layover time, and so on.

If you want to find a restaurant, websites may sort the restaurants by viewers’ recommendations.

When you use a map service to find routes, the routes are usually sorted by the estimated travel time or distances. Imagine that you want to go from Purdue’s research park to Purdue airport by a car, the top two routes take 10 and 11 minutes respectively. The first route travels 4.8 miles and the second route travels 5.4 miles.

The next few slides explain selection sort.

ECE 264 will talk about several sorting algorithms. Selection sort is one of them.

This is how the algorithm works. We will go through every element in the array and find the smallest value. If this value is not already the first element in the array, swap this element with the first element.

We will not touch the first element again.

In the next iteration, the first element is excluded. We find the smallest value. If this value is not the second element, swap the two elements.

The next iteration is similar.

We will not consider the first two elements since they are already the smallest two values in the array.

Among the remaining elements, we find the smallest value and swap it with the first value in the remaining part of the array.

This is called the selection sort because in each iteration, among the remaining part of the array we select the smallest value and swap it, if necessary, with the first element.

Selection sort has two levels of iterations. The outer level go through the first element of the array to the second last element. This iteration marks the order of the value we want to select. If this is the first element, we want to select the smallest value in the entire array. If this is the second element, we are selecting the second smallest value of the array.

The inner iteration selects the smallest value among the remaining part of the array. This iteration starts from the element after the outer iteration to the end of the array, finds and selects the smallest value.

If the smallest value is different from the first element in the remaining part of the array, swap them.

If an array has n elements, the selection sort will swap at most n times.

The outer iteration goes through n – 1 elements. The inner iteration on average goes through n – 1 over two elements. Thus, the selection sort has approximately n square comparison. For simplify, we concise only the highest power term and that is n square.

For reasons that are still unknown to me, many students really love bubble sort. When I ask the students, nobody is able to explain.

K nuth in his book of “the art of computer programming” says that “the bubble sort seems to have nothing to recommend it, except a catchy name and the fact that it leads to some interesting theoretical problems”.

If you agree with K nuth, you can forget bubble sort. ECE 264 will not talk about bubble sort.

The second homework teaches you important concepts about testing: how to create self-checking code and generate test cases.

Let’s start with main dot C.

.

The selection sort function is called on line 67. This function has two arguments. The first argument is an array of integers. The second is the size of array. i.e., the number of elements of the array.

After the selection sort is called, the array should have been correctly sorted. The main function calls another function, check order, to check whether the array is in the ascending order. If the array is not in the ascending order, an error message is printed.

This slide shows the check order function. Let’s see what this function does. It checks whether the elements are sorted in the ascending order.

This function has only one loop. It starts from the very first element to the second last element. It checks whether any element is greater than the element after it. If this occurs, the array is not sorted in the ascending order.

Please notice that this condition uses greater than, not greater than or equal to. By using greater than, this function allows two adjacent elements to be the same.

If the iteration goes to the end, the array is in the ascending order. The function returns true.

How do you generate test cases? You can type the test cases by hand, of course. However, there are simple ways to generate test cases. For example, you can use the shuf command to shuffle numbers. In this example, we shuffle numbers one to ten.

The output of the shuf command can be saved to a file by using greater than sign. This is called redirection.

Another command in Linux is the sort function. This command can sort numbers. Please remember to add minus n to treat the content as numbers. With minus n, one zero is treated as ten and it is greater than nine. Without minus n, one zero is treated as two letters and one is smaller than nine.

It is also possible to save the output of sort to a file using redirection.

This homework also provide a function for you to print the array’s elements.

Please notice that you need to pass two arguments: the array and the size of the array.

Let’s move our attention to the main function.

At the very beginning, the function checks whether A R G C is at least two. The reason is that we need to use A R G Vee one as the input file name.

As explained earlier, A R G C is at least one because A R G Vee zero is the name of the program.

This program uses A R G Vee one as the input file name.

The program opens the file for reading. The function is F. open. The second argument R means reading this file.

If F open succeeds, this function returns a function pointer. This program uses the variable F P T R to store the file pointer.

If F open fails, F P T R will be NULL. Before we read the content of the file, it is necessary checking whether F open is successful or not. If F open fails, print an error message and return exit failure.

Many reasons can make F open fail. For example, if the file does not exist, then F open fails. Even if the file exists, it is possible this programmer has no permission to read this file. In that case, F open also fails.

Let’s continue discussing the content of the main function.

Lines 39 to 42 read one integer at a time from the file and count the number of integers. The function F scan F is similar to the function scan F, except F scan F reads from a file. The first argument is the file pointer.

F scan F % d means reading an integer. The return value of F scan F is the number of items successfully read. In this case, only one integer is read. Thus, if one integer is read successfully, F scan F returns one. When the program reaches the end of the file and cannot read any more integer, F scan F returns zero.

Line 45 allocates memory to store the data from the file. The amount of memory needs depends on the number of integers in the file. Thus, we have to count the number of integers earlier in lines 39 to 42.

If memory allocation fails, A R R will be NULL. The program closes the opened file and returns exit failure.

In line 39, the program got out of while because the program reaches the end of the file. After the memory has been allocated, we need to go back to the beginning of the file, read the content of the file, and store the integers in the array.

Line 53 uses F seek to return the beginning of the file.

Lines 55 to 65 read integers from the file. Each time, the program reads one integer in line 57 and stores the newly read value into the array.

If anything goes wrong, this program closes the file in line 60, releases the allocated memory in line 61, and returns exit failure in line 62.

If the reads all integers successfully, the program reaches line 66 and closes the file.

At the moment, the program has all the integers from the file and is ready to call the selection sort function in line 67.

Line 70 checks whether the array’s elements are sorted in the ascending order. If the array is not correctly sorted, the program prints an error message.

Line 74 prints the array and line 75 releases the memory allocated earlier by malloc.

Let’s summarize what we have done so far in the main function.

It checks whether A R G C is at least two because we need to use A R G Vee one as the input file name.

The main function opens the file whose name is A R G Vee one. If F opens fails, the program returns exit failure.

The main function reads integers from the file and counts the number of integers.

Then, the main function allocates memory to store the integers.

Using F seek, the program returns to the beginning of the file.

The program reads the file again and stores the integers in the array. The main function uses F close to close the opened file.

At the moment, the program is ready to call the S sort function to sort the array.

After the array is sorted, the main function checks whether the array elements are in the ascending order.

Finally, the program uses free to release memory and returns exit success.

How do we test this program? You can find three test cases in Make file.

The sort program takes each test case and saves the output to a file. Then the diff command compares whether the program’s output is identical to the expected output.

It is possible to run all three test cases by typing

Make test all

This make file provides another way to run multiple test cases.

If you type make test for, all three test cases will run.

How does this work?

There is a counter called case and its value can be one or two or three. To get the value of case, we need to put two dollar signs in front.

The last part of this lecture explains correct and wrong ways to test programs.

The basic principle is to separate product code from development code. What is product code? It is the code to create products. Your homework submission should be considered as product code.

Product code should be polished. Make sure it is well written and contains only necessary parts for the products. Do not use assert in product code because you do not want the program to stop suddenly. There should be no debugging messages.

In contrast, the development code is for internal use. You may experiment different methods before deciding which method to use for the product. You may have additional instrumentation code for testing. You may use assert here during testing. If you wish, you can add debugging messages here. The key difference is that development code is for your own use.

Software developers will tell you the one to five rule: for each line of product code, you need to write five additional lines of development code. The development code is never shipped to customers but you need development code to help you test.

Here is an illustration how to separate product code and development code. Imagine that X is the function that will be in your product.

If you want to test function X, you write in a separate file to prepare the data needed for testing X. After X is called, your development code check whether data processed by X is correct.

For the second homework assignment, the X is the S Sort function and the development code is the main function to test S sort.

Next, I show you a wrong way to test your code. This mix the product code with the development code. The testing code is scattered around the necessary product code.

There are many problems when you mix product code with test code. First, by mixing the testing code with the product code, the product code will likely be slower. Second, your product may have unwanted debugging messages in front of customers.

Many students claim that they will clean up the test code before submission. It is almost impossible to do this correctly in a rush. There were many cases when students either did not completely clean up their debugging code in submission, or accidentally deleted code that should have been kept.

In summary, mixing product code with development code is an invitation of trouble. You should not do that.