Algorithm Analysis - Lab Exam

Dec 2020-2021

Student's name: S. Xiao Fernández Marín

Code	Plots	Questions	Total

Delivery will be made through a single .tar.gz, .zip or .rar file with the name **EXAM_1292_lastname_name [.tar.gz / .zip / .rar]** This file must contain:

- All the necessary **sources** to compile the exercise.
- A doc or PDF file containing the answers to the **questions**, and if necessary, an indication of the modifications that may have been made in the code.
- Previous document should also include the requested plots. You should ALSO INCLUDE the curve fitting results you calculated, this means the **functions and the values of the linear regression** obtained (you could take a screenshot of gnuplot).

Create a new function that generates permutations with values that follow a potential distribution (like Assignment 3 one). This function receives a N integer parameter that indicates the size of the permutation and returns the permutation (int *).

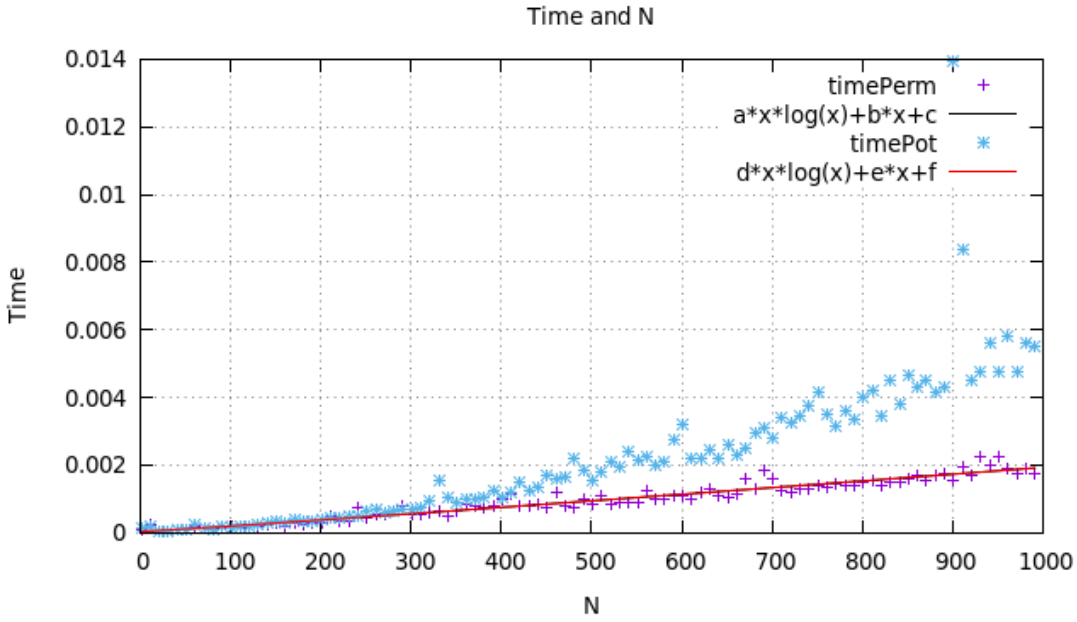
```
int* generate_perm_pot(int N){
  int i = 0;
  int *perm = NULL;
  if (N <= 0) return NULL;
  perm = (int*) calloc ( N, sizeof(perm[0]));
  if (perm == NULL) return NULL;
  for (i = 0; i < N; i++){
    perm[i] = .5+N/(1 + N*((double)rand()/(RAND_MAX)));
  } return perm;
}
```

1. **Use the Quicksort algorithm and compare the sorting times when using our original function that creates a random permutation and the new one.**

```
valgrind ./exercise5 -num_min 1 -num_max 1000 -incr 10 -numP 20 -outputFile qs_perm.txt

valgrind ./exercise5 -num_min 1000 -num_max 10000 -incr 10 -numP 20 -outputFile qs_perm_pot.txt
```

2. **Plot both times in the same graphic and figure out the complexity of both cases (use curve fitting).**



In this plot I have compared the average clock time of QuickSort. We can see that the timePerm fits perfectly the theorical average clock time. As I have put the equation of the QuickSort, it adapts it to the not potential one.

3. **Which permutation is faster to sort? Why? Explain the difference of the complexities and the reason.**

The fastest function if the timePerm one, the normal one.
This is because in the potential one makes the smallest values much more likely to appear than the bigger ones.
Because of this, when a key is generated and the algorithm tries search it, it will fail more times with the potential one, as the key that it is generated its a random number between the 1 and the 1000 (in my case and the probability of having one number is 1/1000) but the table has much smaller values because we have generate them with the potential function. Also, because we have a lot of numbers repeated, the swaps are not made as fast as they could.