**[Homework](http://vlm1.uta.edu/~alex/courses/2320/homework/homework.html) - Homework 2**

Points: 100   
Topics: O, Θ, summations; generate random data and use it to compare two search algorithms: binary search and interpolated search.

**Answers to tasks 1 and 2 should be submitted in an electronic file called 2320\_H2.pdf.**It can be an electronic document or a hand written document that was scanned as a pdf file.

**Task 1 (10 points)**

Give the answer for the two problems below and **justify your answer using the limit theorem and computing the actual limit**. Saying: "The limit is x because this term dominates the other" is not a valid proof. To clarify, You can use the 'dominant term' argument when you talk about polynomial terms (i.e. nd).   
a) is 2n+1 = O(2n)?   
b) is 22n = O(2n)?

**Task 2 (20 points)**

a) (10 points) Let f(n) = (4/9)0 + (4/9)1 + (4/9)2 + ....+ (4/9)n. Find Θ for f(n).   
b) (10 points) Use the definition with constants to show that f(n) = nlg(n)-15n+14√ n  is Θ(nlg(n)).

**Task 3 (70 points)**

Purpose: generate random data, store it in a file, load it, use it to compare the number of iterations in binary search with those in interpolated search .   
  
We want to compare the behavior of binary search and interpolated search as follows:

* generate an array with *N* random values in the range [*start\_value, end\_value*] and store them in an array, A.
* generate *S* random values in the range [*start\_value, end\_value*]
* sort the array *A*
* for each number, *val*, from the *S* values, search it in *A* using both binary search and interpolated search.   
  Record and print :
  1. the index where *val* was found (-1 if not found in *A*)
  2. how many loop iterations each method required until it finished.
* print the average loop iterations over the *S* searches.

However when running such experiments, it is useful to be able to reproduce them at a later time and thus we want to save the random data used in the experiment in a file. So we break it in two parts:

1. generate the random data and save it in a file.
2. load the data from the file, run the experiments and print the results. For printing the results we want two modes:
   * 1-verbose (prints the sorted array and information for each value that was searched and the averages over all searches. This is useful in debugging and understanding the program and data behavior)
   * 2-not verbose (prints only the averages. This is useful when we are confident that the program is correct and we want to run it for large *N* and *S*).

Program specifications:

1. The program will repeatedly allow the user to make a menu choice: 0-exit, 1-create and save to file, 2-load from file and run binary search and interpolated search.
2. If the user selected option 1 (create and save) he will next be prompted to enter, in this order:
   1. *N* - number of elements in an array,
   2. *S* - number of values that will be searched in the array
   3. *start\_value*
   4. *end\_value*
   5. *filename* - name of file where the data will be saved. It must include the file extension (e.g. data1.txt). It can have at most 100 characters.

The program will open a file *filename* for writing and write on the first line, separated by spaces the 4 numbers: *N S start\_value end\_value*  
Next it will generate *N* random numbers in the range [*start\_value, end\_value*] and write them on a new line in the file. They will also be separated by spaces. (When loaded, these will be used as the array elements.)   
Next it will generate *S* random numbers in the range [*start\_value, end\_value*] and write them on the third line of the file. They will also be separated by spaces. (When loaded, we will search for each one of these among the array elements.)   
Here is a sample of the file format:

16 10 0 100

84 29 64 74 66 53 49 41 2 60 38 23 21 47 58 3

24 48 71 8 79 5 26 44 28 43

Here is a file where the values searched are from the array: [data2.txt](http://vlm1.uta.edu/~alex/courses/2320/homework/hw2_data/data2.txt). Load and run this file as well.

1. If the user selects option 2 (load and run) he will be asked for the file name (e.g. data1.txt) and a number (1-verbose/2-not verbose).   
   It will open the file with the given name and load the data.   
   Next it will sort the array.   
   For each of the *S* numbers on the last line, search it in the array A   
   Print the required information (in verbose style or not). See the sample run below.   
   If the file does not exist, it will print an error message.
2. The program output must match the shown behavior including: menu options (0-exit, 1-create and save, 2-load and run and 1 or 2 for verbose mode), formatting of table, not crashing for file not found, format of files,...
3. For binary search, you can use the code discussed in class. Improve the code to keep track (update) everytime the while loop runs a new iteration.
4. Implement interpolated search. This is a searching method where instead of checking the element in the middle, you check one that is proportionally away from the start as how far the search value is from the start value. For example, assume that N=100 and the numbers in the array are uniformly distributed and are in the range 0 to 1000. If you are searching for value 10, it is more likely to be among the first elements than at the middle. In your implementation, copy the binary search code and replace the calculation of the middle index with:

m = left + (value - A[left])\*(right - left) / (A[right] - A[left]);

This modification will cause interpolated search to crash for some special cases. Thoroughly test your code and fix it as needed.   
Refresh the slides on Examples of Algorithms. See slides at pages 47-50 for an application of interpolated search.

**Sample run (from omega).**Download [data2.txt](http://vlm1.uta.edu/~alex/courses/2320/homework/hw2_data/data2.txt) from this page, or create your own.

[astefan@omega hw2]$ gcc -o search search.c

[astefan@omega hw2]$ ./search

0-exit

1-create and save random data for search.

2-load data from file, sort array and run searches.

Enter your choice: 1

Enter: N S start\_val end\_val filename(with extension): 16 10 0 1000 data1.txt

0-exit

1-create and save random data for search.

2-load data from file, sort array and run searches.

Enter your choice: 2

Enter: filename, mode(1-verbose, 2-not verbose): data1.txt 1

sorted array:

1 104 184 220 378 380 521 641 729 765 802 887 897 909 969 992

| | | found at index| repetitions |

| i| value| interp| binary| interp| binary|

| 0| 630| -1| -1| 2| 4|

| 1| 810| -1| -1| 2| 4|

| 2| 208| -1| -1| 1| 4|

| 3| 28| -1| -1| 1| 4|

| 4| 341| -1| -1| 2| 4|

| 5| 260| -1| -1| 1| 4|

| 6| 339| -1| -1| 2| 4|

| 7| 411| -1| -1| 1| 4|

| 8| 194| -1| -1| 1| 4|

| 9| 926| -1| -1| 1| 4|

| avg| | | | 1.40| 4.00|

0-exit

1-create and save random data for search.

2-load data from file, sort array and run searches.

Enter your choice: 2

Enter: filename, mode(1-verbose, 2-not verbose): data1.txt 2

| | | found at index| repetitions |

| i| value| interp| binary| interp| binary|

| avg| | | | 1.40| 4.00|

0-exit

1-create and save random data for search.

2-load data from file, sort array and run searches.

Enter your choice: 2

Enter: filename, mode(1-verbose, 2-not verbose): data2.txt 1

sorted array:

2 3 21 23 29 38 41 47 49 53 58 60 64 66 74 84

| | | found at index| repetitions |

| i| value| interp| binary| interp| binary|

| 0| 84| 15| 15| 1| 5|

| 1| 29| 4| 4| 1| 4|

| 2| 64| 12| 12| 2| 4|

| 3| 74| 14| 14| 2| 4|

| 4| 66| 13| 13| 3| 3|

| 5| 53| 9| 9| 1| 3|

| 6| 49| 8| 8| 1| 4|

| 7| 41| 6| 6| 2| 4|

| 8| 2| 0| 0| 1| 4|

| 9| 60| 11| 11| 2| 2|

| avg| | | | 1.60| 3.70|

0-exit

1-create and save random data for search.

2-load data from file, sort array and run searches.

Enter your choice: 2

Enter: filename, mode(1-verbose, 2-not verbose): dat.txt 1

File could not be opened.

0-exit

1-create and save random data for search.

2-load data from file, sort array and run searches.

Enter your choice: 0

Bye

The assignment should be submitted via [Blackboard](http://elearn.uta.edu/).   
Put search.c and 2320\_H2.pdf file in a FOLDER called hw2, zip the hw2 FOLDER (it will produce a file called hw2.zip). Submit the hw2.zip file. You will have to do these steps for every homework.

As stated on the course syllabus, programs must be in C, and must run on omega.uta.edu.   
INCLUDE THE COMPILATION AND EXECUTION INSTRUCTIONS in the 2320\_H2.pdf. If your program requires a different compilation instruction than mine and you did not include it, there will be a 10 points penalty. For example, for my implementation, those instructions would be:

compile:

gcc -o search search.c

run:

./search

**IMPORTANT:** Pay close attention to all specifications on this page, including file names and submission format. Even in cases where your answers are correct, points will be taken off liberally for non-compliance with the instructions given on this page (such as wrong file names, wrong compression format for the submitted code, and so on). The reason is that non-compliance with the instructions makes the grading process significantly (and unnecessarily) more time consuming. Contact the instructor or TA if you have any questions.

Back to [Homework](http://vlm1.uta.edu/~alex/courses/2320/homework/homework.html) page.