# Algorithm Research ; Program Idea(s)

This homework will be divided into two subsections. In the first section, you will provide a quick summary of each of algorithms that you will be looking at this term ; you should also select two algorithms to implement (to write from scratch) AND two algorithms to use (by finding pre-existing source code online that you can use). In the second section, you will write out a quick summary/overview of a program that you will write during this term that will implement the two algorithms you've chosen. In the third section you will clearly identify the algorithms that you’ve chosen to implement (to write from scratch) and the algorithms that you will be using (but not writing yourself)

Due date(s) for this assignment are listed in the syllabus.

## Part 1-A: Algorithm Research

**This part must be done individually.**

For this section, you will hand in a Word document containing the table that is given to you below. You must fill in the table with appropriate, accurate, and clear explanations. The objective here is to make sure that you have an idea of what each algorithm is normally used for 'in the real world'. The objective is not to force you to learn all the algorithms (that's what the rest of the term is for ☺ ), but instead to force you to learn enough that you have a reasonably good idea about which algorithms you want to learn in detail on your own. I highly recommend using Google to find pages about each algorithm, and look for a 'plain English' description of each one.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| An Example Of What To Fill In Where:   |  |  | | --- | --- | | Algorithm Name | QuickSort ***// This is filled in for you.*** | | 1 sentence summary | Sorts an array into a particular order, normally in O(N●Log2(N)) time  ***// 1 sentence, that explains what sort of problem this algorithm solves.*** | | Examples of use | Any time you want to sort an array, generally.  EXAMPLE 1:  In my game I will need to sort an array of Weapon objects so that the highest-damage weapon ended up at the front; this is an algorithm I could use to do that. EXAMPLE 2:  In my game the player will do battle with enemies. The AI for player’s allies will always attack the strongest opponent. I could use QuickSort to sort the opponents in order to determine which one the player’s allies should attack first.  ***// Short summary. Pick a unique example either from the 'real world', or else from within your particular program.*** | |

|  |  |
| --- | --- |
| Algorithm Name | AVL Trees |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | B-Trees |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | Skip Lists |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | Hash Tables |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | MergeSort |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | Heapsort |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | Dijkstra's Algorithm  (Category: Graphs: Single-source Shortest Path) |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | Floyd-Warshall Algorithm (Category: Graphs: All-pairs Shortest Path) |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | Greedy Algorithms (Category) |
| 1 sentence summary |  |
| Examples of use |  |

|  |  |
| --- | --- |
| Algorithm Name | Dynamic Programming (Category) |
| 1 sentence summary |  |
| Examples of use |  |

## Part 2: Program Idea

For this section, you will hand in a Word document containing your overall idea for a program that you will implement over the course of the term. You need to write at least one page, but no more than 2 pages. You must use a single-spaced 12-point Times New Roman (or 11 point Calibri) font, with normal page dimensions (i.e., no messing around with the margins), etc. The objective here is to force you to write in enough detail that you are forced to think the program through at a more-than-superficial level. The objective is not to force you to think through every single nuance of every aspect of the program, however ☺

You may work with a partner on your program, but keep in mind that each of you needs to implement your own, independent algorithm. In other words, you can collaborate with someone else to build up an even bigger program (since you’re each implementing two, separate algorithms each)(for a total of four algorithms from the two of you), but you cannot work together to implement the algorithms themselves.

You need to clearly specify when you want your due dates to be. The instructor reserves the right to change due dates; the instructor will attempt to consult with the student first. Note that **you are not allowed to choose a due date for the initial version of an algorithm during the last two weeks of the course**, because the instructor may not be able to send you feedback on the initial version in time to do a revision before the quarter ends.

## Part 3: Choosing Algorithms And Due Dates

Please provide a clearly labeled, concise list of the two algorithms that you’ve chosen to implement. By ‘implement’ I mean that you will be writing the code for yourself, from scratch, without copying any existing source code.

Please ALSO provide a second, clearly labeled, concise list of the two algorithms that you’ve chosen to use. By ‘use’ I mean that you will NOT be writing the code for yourself. Instead, you will find existing source code online that you can integrate into your program. **Make sure that the code has a license that allows you to use the code in your program**.

Please ALSO provide a third list of due dates that you agree to. You should list two due dates, and on each date you should list which two algorithms you will have completed. Personally, I recommend choosing to implement (write from scratch) one algorithm and to use existing code for one algorithm for each due date, but you can do whatever you’d like as long as you’re finishing two algorithms on each due date.

Please keep in mind the following:

* **You’re expected to choose a due date that’s about a week after we cover in class the algorithm that you’re implementing.** You don’t absolutely have to, but you’ll need a good reason to deviate by more than a lecture or two.
* It’ll take at least a week for your instructor to grade whatever you hand in. If you want a chance to do a revision please factor that time into your due dates.
* Everyone else will want to choose the latest possible date to hand in their work, and this will delay the instructor further.
* Remember also that choosing a very late date makes it more likely that you won’t complete the work (for example because you run into problems late in the quarter and then immediately run out of time to fix it in, or because you’ve forgotten the material that was covered two weeks ago and you’re having trouble digging up resources that help you out).
* The instructor reserves the right to change your due dates.   
  I’ll try to communicate with you before doing that, but this is an option.

Note: I’m only really splitting this out into a separate part of the assignment so that y’all don’t forget to do it. As long as it’s clear which algorithms you’re implementing and which ones you’re using then I’m fine with you combining this section and the previous section.

## How to electronically submit your homework:

Please use StudentTracker to hand in your work on or before the deadline listed on the course webpage.

Each of your files should be clearly named, so that the instructor has no problems finding the document(s). For example, naming your two Word documents, and the directory containing the game, as follows, would be great:

**Assignment\_1\_Part\_1.doc**

**Assignment\_1\_Part\_2.doc**