AI Search Assignment FAQs

- Student: *Why do you insist on Python*?
- Iain: Every student has completed Computational Thinking and so can program in Python; this yields a 'level playing field' when it comes to implementation. Also, restricting to Python-only leads to fairer marking.
- Student: *Why do you take the maximum tariff from two 'correct' implementations as the 'sophistication' mark*?
- Iain: On occasion, a more sophisticated algorithm may give worse results than a more elementary algorithm. I want to reward both the quality of the tours that you find and also your accomplishments in coding up a more difficult algorithm. With things set up as above, I encourage you to code up a more sophisticated algorithm without harming your chances of getting good tours.
- *So I guess that's why you take the best overall tour found when you give the 'basic quality' mark*? *So that we don't harm the quality of the tours we find if we choose to implement a more sophisticated algorithm*?
- Iain: Correct!
- Student: *And I suppose you ask us to implement two algorithms and to experiment so that you can assess both our understanding of the algorithms in the course, through our capacity to code them up, along with our ingenuity and deeper understanding in obtaining good tours*?
- Iain: Correct again! If all I asked you to do was to produce basic implementations of two algorithms and produce some half-decent tours then there wouldn't be much of a challenge (nor fun) in that. So, I would like you to experiment and enhance. I understand that some of you will have more time and inclination for this than others so I ensure that if all you do is produce basic implementations and some half-decent tours then this will suffice to pass the coursework, which I think is fair. But I'm also giving you the opportunity to show me what you can do and be rewarded for it. In the past, many students have enjoyed the challenge of experimenting and enhancing and producing good tours.
- Student: *I think that we need to be careful when choosing our algorithms as if, for example, we choose a high-tariff algorithm and a low-tariff algorithm and our high-tariff algorithm turns out to be incorrect then our 'sophistication' mark will be low too.*

- Iain: You would be wise to bear this in mind when you are choosing the algorithms that you intend to implement. A simple calculation will show you that a high-tariff algorithm being incorrect can significantly reduce the mark you receive for this assignment. Please think very carefully about the algorithms you choose to implement (and their termination on my secret city files within one minute). There are more dangers with implementing a high-tariff algorithm, in terms of getting a smooth-running and correct implementation, than a low-tariff one but this needs to be considered not just against the mark rewards but also against the quality of tours overall. A wise course of action might be to implement more than two algorithms as the course progresses and eventually dispense with the additional one(s), but this is up to you.

- Student: *For the 'quality' mark for a tour, is this obtained by a relative comparison of how my tour compares to others obtained from the same algorithm or across all algorithms*?

- Iain: The 'quality' mark for a tour is obtained relative to all other tours for that city file, irrespective of the underlying algorithm.

- Student: *Why do you not just ask us to return the best tour we have found by whatever algorithm for each of the* 10 *city files rather than one tour for each algorithm*?

- Iain: I get to see the profile of tours you have produced for each algorithm. This allows me to have confidence that the tours you have supplied to me are indeed produced by the implementations stated, as I know (roughly) how different algorithms should perform; in fact, I automatically analyse *all* student tour files supplied to me and I can easily see when a claimed implementation is 'out of line', when two students' tours are identical or very similar, or when a tour has not been produced by the algorithm claimed by the student. Also, I might run your basic and enhanced implementations on the 10 city files to provide me with a sanity check that your codes and tours are what you say they are, though I'll probably only do this if I am suspicious!

- Student: *Are you often suspicious*?

- Iain: It doesn't really matter whether I am or I'm not as all student submissions are automatically compared against each other and the data I obtain (which is substantial) alerts me to problematic submissions. As I said, I analyse all tours submitted per algorithm and I also compare all student implementations against each other, on a pairwise basis, using plagiarism software that is specifically designed to

detect plagiarism in Python programs (the plagiarism software that I use is very good and has been developed by academic researchers). In fact, I also compare student codes with codes from previous years and with other codes that might be publicly available. I should add that each year, the city files are randomly permuted so that tours from previous years are useless!

- Student: *Do you ever obtain discrepancies*?

- Iain: Unfortunately, yes. Over the past 3 years, my analysis flagged some concerns and over 20 students have been found guilty of plagiarism, either by colluding with each other or by using programs that were implemented by students in previous years or available on the Web. They all received a mark of 0 for the assignment. My advice to all students is

DO THE WORK YOURSELF AS OTHERWISE YOU WILL BE FOUND OUT!

Do not be tempted to cut-and-paste from codes that are not your own as it is very difficult to avoid plagiarism (I have a personal library of codes that I have found on the Web and these codes are used in my plagiarism checks). Also, even if you cite your source then you will almost surely be deemed guilty of plagiarism as the whole point of this assignment is to assess *your* coding capability rather than someone else's. There is simply no need for you to use other people's codes or generative AI: sufficient pseudo-code for all algorithms is supplied in the lectures.

- Student: *What about if we consult the research literature and find potential enhancements to algorithms in research papers that we might code up and try*?

- Iain: It is absolutely fine to consult the research literature and try and find different ways in which to enhance your basic algorithms; indeed, I encourage this. However, you need to cite your sources (otherwise it will be deemed plagiarism) and also ensure that your code is not simply cut-and-paste: you should understand the underlying methodology and implement any enhancements for yourselves. You should provide clear descriptions in the proforma and copious comments in your code to demonstrate to me that you know what you are doing.

- Student: *Are there other 'discrepancies' that we should avoid*?

- Iain: Yes. If you claim to have made enhancements but they don't appear in your enhanced code or if you simply hand in a copy of a

basic code as your enhanced code then I interpret this as deliberately trying to mislead me and you may receive 0 as your overall mark. Also, if you mess around with the font sizes or the answer space in the proforma then the proforma will be assumed not to exist.

- Student: *What sort of things might we get fined for*?
- Iain: Last year students were fined for: not putting their submission files in a folder before zipping; not naming tour-files or codes appropriately; including tours in sub-folders within their main folder; not including a validation file; manually amending parts of the programs that I said not to touch; supplying tour-files that had been tampered with; supplying tours that were not obtained by the code claimed; not including a user-name in code and tour-files; and implementing an algorithm from outside the course without my consent. All fines could have been avoided if the students had read these instructions.
- Student: *Why do you ask us to return our validation feedback*?
- Iain: In the past, some students didn't bother validating and as a consequence, because they had tampered with something they shouldn't have or not supplied information they should have, their codes didn't run. So, I now fine students who don't return their validation feedback file as an incentive to run the validation feedback code (and so potentially save themselves from disappointment). Remember: if you don't submit your validation feedback then you will be fined, even if your codes run. By the way, even though I explained all of this last year, there were still 29 fines!
- Student: *Can I change the subject and ask about enhancement?*
- Iain: Of course. Shoot!
- Student: *First, the basic implementation. Is it the case that you are not too fussy as to the nature of a basic implementation and that so long as the implementation is of a standard version of the algorithm in question, this will be fine*?
- Iain: No, this is not the case. All basic implementations should follow the pseudocode structure as presented in the lectures. The only exception is for a brute-force search where any implementation that checks all possible tours will suffice. You should only diverge from the pseudocode in lectures in your enhanced implementations and you should explain why you have done this.
- Student: *Can we take an algorithm from the course that enhances another algorithm from the course as our enhancement*? *For example, can we take Elitist Ant System as an enhancement of Ant System*?

- Iain: No. You should regard any algorithm from the course as a 'basic' algorithm and you need to enhance any 'basic' algorithm in an original way. Your two chosen algorithms must have different codes from the algorithm codes and tariffs file.

- Student: *What about if I implement, say, a genetic algorithm but its structure is different to the one in lectures? In fact, what about if I cut-and-paste some genetic algorithm Python code I found on the Web?*

- Iain: It goes without saying that I want you to develop your own code. As I said above, you should keep to the pseudocode structure in lectures for your basic implementations. If you diverge from this structure in an enhanced implementation then this should be for experimental reasons, e.g., 'I implemented a genetic algorithm of slightly different structure to the one in lectures as this allowed me to introduce experimental variations that I couldn't do otherwise.' An illegal reason is 'I found some genetic algorithm code on the Web and the structure was different to the algorithm outlined in lectures.' ***Do not cut-and-paste code from the Web, even if you cite your source! As I explain above, I use plagiarism detection software and you will be found out!*** I will regard cut-and-paste code as plagiarism even if the source is cited. I don't want to stop students starting to implement algorithms before they have been covered in lectures but if you do this then you should bear what I have said above in mind.

- Student: *Where do you draw the line between enhancing a basic algorithm and writing an entirely different algorithm?*

- Iain: Drawing the line is easy: if your enhanced implementation is not building on your basic implementation in an obvious sense (look at the codes) or does not follow the general structure of the underlying algorithm then it is an implementation of a different algorithm (which is not allowed). If I feel that I am being deliberately misled or lied to in the proforma then I will regard this as cheating.

  Also, on a related note, I would prefer that enhancements are algorithmic enhancements rather than implementation ones. For example, you might claim that your enhancement is to speed up the implementation in some way (such as using multiprocessing). This is not really what I am looking for: this is a course about algorithms, not (Python) implementations, and you should focus on algorithmic enhancements. This does not mean to say that you should not speed up your implementations; for one thing, doing so will generally mean a more extensive search and (hopefully) shorter tours. It's just that

you won't obtain enhancement marks for doing this. Also, the way I have set up my skeleton code makes it difficult to incorporate multiprocessing. Having said this, there is nothing to stop you messing around with the skeleton code and using multiprocessing in order to obtain your handed-in tours but removing skeleton code amendments and multiprocessing in the codes that you hand in. But be careful if you choose to do this and ensure that everything validates.

- Student: *Is it the case that 'enhancement' marks will be awarded using what we say in the proforma*?

- Iain: Yes. I will award 'enhancement' marks according to what you say in the proforma. You should describe the enhancements you made and any other relevant information within the context of these enhancements. Do not expect me to delve into your code to work out what you have done; on the other hand, do expect me to look at your code to verify that you have actually done what you say you have done. Clearly commented and readable code will be in your own best interests. If I cannot *quickly* verify that your enhancements are what you say they are then I'll just award a mark of 0. It is your job to be as clear as possible. The 'enhancement' mark will be determined by the depth, intricacy and novelty of your enhancements, irrespective of the quality of the tours produced, which will be rewarded through the 'enhanced quality' mark. A high number of (low-level) enhancements does not necessarily lead to a high 'enhancement' mark. As I mentioned earlier, if you use a research paper as the basis for your enhancement then please ensure that you reference the paper *in full*; that is, so that I can use your reference to find and consult the paper (an embedded DOI link will suffice). Otherwise, your enhancement might be regarded as plagiarism. Also, provide explanatory comments in your codes to convince me that you know what you are doing. As I said earlier, poorly commented code might accrue fines.

- Student: *Any tips as regards how we fill the proforma*?

- Iain: Yes. You should try and allocate space for each of your enhancements and maybe even itemize them. There is nothing to stop you using bold font, say, to name each of your enhancements. Be precise and succinct and don't waste space by telling me, for example, how a genetic algorithm works: I know this. All I'm interested in is information about your enhancements.

- Student: *Suppose that I do lots of experimentation and settle on the best variation to hand in. Will I receive credit for the experimentation that I did but didn't hand in as my enhanced algorithm*?

- Iain: It is up to you to convince me in the pdf and in your codes that you did indeed undertake the experiments you say you did. Apart from proper descriptions in the pdf, one way of doing this would be to include other experimentation code in your enhanced algorithm but commented out (even better if you tell me in the comments how to comment it back in and run it for myself).

- Student: *Is it the case that in order to get 'enhanced quality' marks, the tours produced by my enhanced implementation must be better than those produced by my basic implementation*?

- Iain: Yes, this is the case. I will define a minimum amount by which an enhanced tour has to be better than a basic tour, for each of my secret city files (these amounts will be decided when all assignments have been submitted and will be internal to the marking process but will be fair and reasonable).

- Student: *What if we have worked hard to fine-tune our implementations to perform really well on the* 10 *city files but when you run them on your secret city files, they don't do so well*?

- Iain: There is a small chance that this might happen but if your enhanced implementation improves things across many of the 10 city files, it is likely that it will improve things on my secret city files too. Also, the 'enhanced quality' mark is relatively small. Similarly, if your algorithm is randomized then it might not do so well on the solitary run of it that I undertake. Such is life! However, a decent randomized algorithm should perform well almost all of the time and for a good randomized algorithm to go wrong on one run, you'll have to be extremely unlucky.

- Student: *If I restrict the run-time of my implemented algorithms but the tour produced by the enhanced implementation is worse than that produced by the basic implementation then should I return the basic implementation as the enhanced implementation*? *That is, should the enhanced implementation always produce the tour with the shortest length*?

- Iain: Your enhanced algorithm should be the basic algorithm with additional enhancements. If your enhanced implementation produces a worse tour than the basic implementation (possibly when forced to terminate within one minute) then that creates a problem for you (as you will receive no enhanced quality mark). In particular, this doesn't say much about the quality of your enhancements!

- Student: *I don't fully understand the timeline for the assignment. The high-tariff algorithms are not covered until the end of the term*

and so what am I supposed to be doing in (and outside) AI Search practicals if I have to wait to see the high-tariff algorithms, which are the ones I want to implement?

- Iain: Of the two algorithms you implement, only the one with the highest tariff will contribute to the 'sophistication' mark. As I explain above, just because an algorithm is high-tariff does not mean to say that it will provide the best tours. It is up to you as to how you manage your time as regards the assignment but personally I would be developing basic implementions of some of the algorithms as I go, looking for algorithms that give good tours (and subsequently forgetting about implementations that don't provide good tours). Of course, doing this gives you a deeper understanding of the algorithms and so better scope to enhance them and obtain better tours. Alternatively, you might wait until you have seen all the algorithms before starting to implement. If you are managing your time in this way (though, as I have said, this would not be my choice) then clearly you would be using the practical sessions to do other things, possibly on other modules (but you would lose any potential help that the demonstrators could give you in practicals). If you wait until the final lecture before starting to implement then that still gives you plenty of time to undertake the assignment (but this would definitely not be my approach). The take-home message is: you should manage your time properly and plan in advance how you intend to work on all of the modules that you are undertaking this year.

- Student: *Can you give me some illustrations of the different mark awards depending upon what sort of a student I am*?

- Iain: OK; here are some illustrations. Let's suppose that the maximum: sophistication mark is 9; correctness mark is 4; enhancement mark is 8; basic quality mark is 8; and enhanced quality mark is 1 (though I reserve the right to vary these values slightly).

  - Student A: Maybe you are hard-pushed and will not have the time nor inclination for experimenting but correctly implement a reasonably sophisticated algorithm at tariff 8 and a less complicated algorithm at tariff 6; so, your sophistication mark will be 7.2/9 and your correctness mark will be 4/4. The lack of experimentation means: that you get an enhancement mark of 0/8; and that you only obtained moderately good tours overall. Perhaps you get a basic quality mark of 5/8 and so an overall quality mark of 5/9. You would score 16.2/30 = 54% (a mid-range lower-second mark).

- Student B: Maybe you are struggling and cannot correctly implement two algorithms but get a basic tariff-6 algorithm working, though the tours produced are not very good, resulting in a basic quality mark of 4.6/8. You would receive a sophistication mark of 5.4/9 and a correctness mark of 2/4. As you haven't enhanced then you would get an enhancement mark of 0/8 and an enhanced quality mark of 0/1. Your total mark would be $12/30 = 40\%$ (a bare pass mark).
- Student C: maybe you are like Student A but you are more inclined to experiment and enhance. You choose the same algorithms as Student A and your enhanced implementations are correct and reasonably innovative; so you obtain an enhancement mark of 4/8 and improve your tours so that you get a basic quality mark of 5.5/8. The tours of the secret city files produced by your enhanced implementations produce a reasonable improvement over those produced by your basic implementations and you obtain an enhanced quality mark of 1/1. So, you would score $21.7/30 = 72\%$ (a first-class mark).

The moral of the story? Show ambition with your implementations and experiment and enhance!

- Student: *I'm pretty ambitious and want to implement an algorithm not covered in the lectures. Can I do this*?

- Iain: Yes, ***but you must obtain explicit permission from me!*** First, take a look and see if your algorithm already has a tariff allocated in the text file `alg_codes_and_tariffs.txt`. If so then make sure that the algorithm you are thinking of really is the algorithm mentioned in the file. You need to email me for clarification so that you don't misinterpret things and to receive my consent for you to implement the algorithm. If your algorithm does not appear in `alg_codes_and_tariffs.txt`, email me with a reference as to the algorithm you wish to implement and, after consideration, I'll give you a tariff and add it to the file. Either way, I will explicitly tell you in an email whether or not you can implement the algorithm under discussion. ***You cannot implement any non-standard algorithm without my consent!*** In the past I have refused consent to students because the proposed non-standard algorithm was too complicated, because I did not feel that they had the knowledge or capability to implement the algorithm in question, because they clearly did not have a full understanding of the algorithm in question or because the algorithm was outside the spirit of this sub-module. Remember: if you receive consent to implement a non-standard algorithm then you need to provide both a description of it in the proforma (using

pseudocode and/or natural language) and also a full reference of your source. There is additional space in the proforma that can be used when a non-standard algorithm is implemented (if you don't implement a non-standard algorithm then just ignore this space)

- Student: *By the way, how do we know when we have found an optimal tour for some city set*?

- Iain: Almost always you don't. I don't know what the shortest tour is for some of the city sets and in the real world, we almost always don't know this either. The point of using heuristic algorithms such as the ones we develop is that we don't have efficient exact algorithms but we need a solution that is as good as we can make it and so that this solution is produced relatively quickly. That's why we turn to heuristic methods. It will be up to you to decide when you stop looking for a better solution to some city set.

- Student: *Why do you not allow us to import non-standard modules into our Python codes*?

- Iain: For three reasons. First, many of you will have only been programming with Python for just over a year and so the practice of coding up basic data structures will do you good. Second, you simply don't need additional modules to cope with the algorithms that you will be implementing. Third, there may exist some wacky modules out there with ready-made implementations of the algorithms we are working with, which defeats the object of the assignment!

- Student: *How fussy are you that we follow the guidelines stated here and in the programs that you supply to us*?

- Iain: I am extremely fussy! ***I simply refuse to go through your submissions so as to correct mistakes you have made by not following the guidelines, no matter how trivial these mistakes might be. This is non-negotiable.*** If you submit material without bothering to validate it with the (painstakingly-prepared) program `validate_before_handin.py` then you only have yourself to blame.

VALIDATE BEFORE HAND-IN!

If you don't validate before hand-in then you will be fined!

- Student: *What should I do if when I validate my files before handing them in, I get a fatal error and can't understand why*?

- Iain: There are one or two common pitfalls. Make sure that when you download the city files from ULTRA and put them in a folder, you don't inadvertently also include garbage files in the folder (this can happen especially if you are using a Mac). Also, ensure that you download the tariff file `alg_codes_and_tariffs.txt` too and that all files are in their correct locations (see the comments at the start of the program `validate_before_handin.py`). As a last resort, email me.

- Student: *Will you answer questions we might have about the assignment*?

- Iain: Yes, of course I will. However, if you ask me a question whose answer is available in this document or the programs I supply to you then do not expect more than a very brief answer alerting you to this fact.

- Student: *I have some final questions related to the TSP. For the TSP, is there a distance given for every pair of cities*?

- Iain: Yes. You should regard an instance of the TSP assignment as a weighted complete digraph where the vertices are the 'cities' and where there is a directed edge from every 'city' x to every distinct 'city' y with the weight being the given 'distance'. Note that it might be the case in general that the distance/weight from city/vertex x to city/vertex y is different from the distance/weight from city/vertex y to city/vertex x. Also, the instances might not correspond to real sets of cities with real (Euclidean) distances; they might originate from an entirely different application or even just be randomly generated. Finally, we are always looking for a shortest length tour where any tour should contain each city/vertex exactly once.

  As a further point of clarification, the city files that I supply to you for the assignment are, in fact, symmetric in that the distance from city x to city y is equal to the distance from city y to city x (as it happens, the secret city files are symmetric too). You will see from `skeleton.py` that I (very helpfully!) unpack a city file on n cities into a full $n \times n$ distance matrix for you to use called $dist\_matrix$ (so you don't need to know the format of the city files although I'm sure you can work it out). You should not assume that any city file is symmetric as this restricts the general applicability of your implementations: if you need to know the distance from city x to city y then look up $dist\_matrix[x][y]$; and if you need to know the distance from city y to city x then look up $dist\_matrix[y][x]$. The take-home message is that your codes should assume nothing about the city files' distances.

Another point of clarification is that, for us, any city-to-city distance is always a non-negative integer but might be equal to 0. Of course, this means that it is possible (and perhaps preferable) to move between the two cities in question at a cost of 0 towards the total tour length.

- Student: *Some TSP algorithms, like Christofides' Algorithm, for example, are specifically designed for TSP instances that satisfy the triangle inequality. Can we assume the triangle inequality*?

- Iain: No. You should not assume anything at all about the instances of the TSP given to you in the assignment. If you want to know whether an instance of the TSP satisfies the triangle equality then you can easily write some code to check this. You might still consider implementing an algorithm like Christofides' Algorithm even if your TSP instance does not satisfy the triangle inequality: the algorithm will still provide a tour but any performance result requiring that the instance should satisfy the triangle inequality will not necessarily hold. (But, as I said above, you need to get express permission from me to implement an algorithm not covered in lectures.)

- Student: *So that I might be able to get a rough idea of how well my implementations will perform when run on your secret city files, can you supply a city file of size about* 100?

- Iain: I don't have city files readily to hand. However, you can easily randomly generate your own to undertake your tests with. I see things like this as wrapped in with all the experimentation I request of you. A word of caution: even if your implementation terminates within a minute for your randomly-generated 100-city city file, it might not do likewise on my secret city files (for the reasons I highlighted above).