|  |
| --- |
| CS301  2022-2023 Spring |

Project Report

Group XXX

::Group Members::

1. **Problem Description**

*Describe your problem both as intuitively and formally. If possible, talk about the applications (where this problem) might be used in practice.*

*State the hardness of your problem in the form of a theorem. For example, give a theorem claiming that your problem is NP-complete, or NP-hard.*

*For the proof of this theorem, you can simply cite (refer to) an appropriate source in the literature which gives this proof, or you can give an explicit proof in your report. In the case you give an explicit proof, and this proof is not a novel proof suggested by your group (which is ok for your report), you must still give the citation to the original paper/book from which you got this proof.*

1. **Algorithm Description**
   1. Brute Force Algorithm

*Find a correct/exact/brute force algorithm for your problem and explain the algorithm in detail (Exponential/Factorial Time, Not Efficient). If you want, you can also design an algorithm yourself. Also give a pseudo-code of the algorithm. If the algorithm follows an algorithm design technique, e.g., divide-and-conquer, dynamic programming, etc., you need to mention this and explain why you think the algorithm is using this technique. (Guaranteed solution no matter the computational complexity.)*

* 1. Heuristic Algorithm

*Find an approximation/heuristic algorithm (Polynomial Time, Efficient) for your problem and explain the algorithm in detail. If you want, you can also design an algorithm yourself. Give a pseudo-code of the algorithm. If the algorithm follows an algorithm design technique, e.g., divide-and-conquer, dynamic programming, etc., you need to mention this and explain why you think the algorithm is using this technique.*

*In addition, if the algorithm is an approximation algorithm, you need to state and show the proof of the ratio bound. The proof need not be a novel proof that your team put forward, although this is perfectly okay. You can just include a proof that already exists in the literature, by citing the appropriate source.*

1. **Algorithm Analysis**
   1. Brute Force Algorithm

* *Claim and show that the algorithm works correctly, possibly in the form of a theorem*
* *For the complexity analysis, drive the worst-case time complexity. Try not to give upper bounds which are too loose. If possible, try to give a tight upper bound by using 𝚹.*
* *Optionally, you can also consider the complexity of the algorithm for resources other than time, e.g., the space complexity.*
  1. Heuristic Algorithm
* *Claim and show that the algorithm works correctly, possibly in the form of a theorem.*
* *For the complexity analysis, drive the worst-case time complexity. Try not to give upper bounds which too loose. If possible, try to give a tight upper bound by using 𝚹.*
* *Optionally, you can also consider the complexity of the algorithm for resources other than time, e.g., the space complexity.*

1. **Sample Generation (Random Instance Generator)**

* *Implement/find a parametric (in terms of the size of the problem) algorithm to produce random sample inputs for your problem.*
* *Put pseudo codes and explanation of the algorithm to your reports.*

1. **Algorithm Implementations**
   1. Brute Force Algorithm

*Implement the brute force algorithm and perform an initial testing of the implementation by using 15-20 samples using the sample generator tool of Section 4.*

*Note that, although you can implement this algorithm yourself (if you want to), it is also fine if you use a code that you find from the internet. However, you should be able to install and run it. Also, you need to get familiar with the source code to be able to answer any questions about the code.*

*Report the results of the initial testing by giving the number and the size of the instances tried. Report any failures and related fixes.*

* 1. Heuristic Algorithm

*Implement the heuristic algorithm and perform an initial testing of the implementation by using 15-20 samples using the sample generator tool of Section 4.*

*Note that, you do not have to implement this algorithm yourself. You can also find the source code from the internet. However, you should be able to install and run it. Also, you need to get familiar with the source code to be able to answer any questions about the code.*

1. **Experimental Analysis of The Performance** (**Performance Testing**)

*In the experimental analysis part of your project, you are expected to analyse the performance of the implementation of the algorithm (****in Section 2-b****) experimentally. The complexity results presented* ***in Section 3-b*** *are theoretical results. The worst-case time complexity of the algorithm may not be displayed in practice. You will assess the practical time complexity of the algorithm by performing performance tests. We expect you to use the statistical methods which you covered in the lectures for this part. For a certain size of input, taking only a single measurement for the running time will not necessarily give a confident measurement. You will need to measure the running time for a certain input size many times and use confidence levels of at least 90% with narrow enough confidence intervals. Here, we can consider an interval [a-b,a+b] narrow enough, if (b/a) < 0.1*

*You need to fit a line for the measurement values and visualize your results by using charts. If needed, you can consider using log-log plot. Derive the running time expression based on the equation of the fitted line.*

*Optionally, you can also perform the measurements for the space usage of your implementation.*

1. **Experimental Analysis of the Quality**

*You need to consider the solution quality of the heuristic algorithm in practice. Since the heuristic algorithms do not necessarily find the exact solutions, it is usually an issue how close they get to the exact/correct answer. To be able to understand this, you will need to design and perform experiments, to see how the quality of the solutions of the heuristic algorithm changes with respect to the problem size. Note that, to carry out these tests you will also need to use an implementation of the brute force algorithm you implemented in* ***section 2-a.***

*To this end, perform experiments for various input sizes that will allow you to derive the relation between the exact solution and the solution that the heuristic algorithm provides. You can also use a chart to visualize the comparison of the exact and the heuristic results.*

1. **Experimental Analysis of the Correctness (Functional Testing)**

*The correctness results given in Section 3-b show that the algorithm is correct.*

*However, there can be errors introduced into the implementation. In other words, there can be coding errors. For this part of work, you will need to perform testing of the implementation to assess the correctness of the implementation of the algorithm. We want you to use some testing methods for your algorithm that is covered in the lectures for this part.*

1. **Discussion**

*Discuss your results in detail. Are there any defects in your algorithm? Is there any inconsistency between your theoretical analysis and experimental analysis?*

**Submission**

On SUCourse, for each group, only ONE person will submit the report.

For Progress Reports, we expect a report as PDF file. Your filename must be in the following format:   
 CS301\_Project\_Progress\_Report\_Group\_XXX.pdf   
where XXX is your group number.

For Final Reports, we expect a report as a PDF file, and also the codes and the data produced during the experiments. Please submit a single zip file that contains all your project files. Your zip file name must be in the following format:

CS301\_Project\_Final\_Report\_Group\_XXX.zip

where XXX is your group number. In this zip file, please make sure that your final report is named as follows:

CS301\_Project\_Final\_Report\_Group\_XXX.pdf

where XXX is your group number.