Spring 2007 Math 101-02 Discrete Mathematics Project Topics

Below you will find four topics you can choose from for your project and what you are expected to do for each of these topics. Send me an email ranking the project topics in order of preference by April 4. I will announce your assigned topic on April 5. You will have to turn in your project report on May 10 at the beginning of the lecture.

The number of projects topics are limited as follows: 6 for DA, 10 for KP, 4 for FF, 6 for HT. Topics will be assigned on a first-come, first-served basis.

DA. Shortest Path: Dijkstra’s Algorithm

1. List some real-life examples of the problem.

2. Formulate the problem in graph theoretical terms. Before you do this you might have to introduce some new terminology that is not covered in class. Note that Section 13.1 of Grimaldi [2] has the directed graph version whereas Section 5.5 of Liu [5] has the undirected graph version of the algorithm.

3. State the algorithm.

4. Give two examples showing how the algorithm works. One of these should involve a graph with $|V| \geq 6$ and $|E| \geq 12$ and the other one a graph with $|V| \geq 10$ and $|E| \geq 15$. You might want to use an undirected graph in one of these examples. In each of these examples illustrate every stage of the algorithm with a different picture. (This means for every change of $i$ in Grimaldi’s notation and for every change of $P$ in Liu’s.) Explain what is happening between the pictures at one point in the first example and at two points in the second.

5. You do not have to prove anything or explain why the algorithm works. You do not have to estimate the complexity of the algorithm.

References:

KP. Minimal Spanning Tree: Kruskal’s and Prim’s Algorithms

1. List some real-life examples of the problem.

2. Formulate the problem in graph theoretical terms. Before you do this you might have to introduce some new terminology that is not covered in class.

3. State the algorithms.
4. Give two examples in which you will apply both of the algorithms. One of these should involve a graph with \(|V| \geq 6\) and \(|E| \geq 12\) and the other one a graph with \(|V| \geq 10\) and \(|E| \geq 15\).

In each of these examples illustrate every step of the algorithm with a different picture. (This means for every change of \(i\) in Kruskal’s algorithm and for every change of \(T\) in Prim’s algorithm in the notation of Section 13.2 of Grimaldi [3].) Explain what is happening between the pictures at one point in each of the first pair of examples and at two points in each of the second pair.

5. You do not have to prove anything or explain why the algorithms work. You do not have to estimate the complexity of the algorithms.

References:

**FF. Maximum Flow: Ford-Fulkerson Algorithm**

1. List some real-life examples of the problem.

2. Formulate the problem in graph theoretical terms. Before you do this you might have to introduce some new terminology that is not covered in class.

3. State the algorithm. You do not have to use the Edmonds-Karp algorithm to determine an \(f\)-augmenting path. You might prefer to use the algorithm as given in Section 6.8 of Liu [4]. Note that you will not need all the material presented in Section 13.3 of Grimaldi [2] in this context.

4. Give two examples showing how the algorithm works. One of these should involve a graph with \(|V| \geq 6\) and \(|E| \geq 12\) and the other one a graph with \(|V| \geq 10\) and \(|E| \geq 15\).

In each of these examples illustrate every stage of the algorithm with a different picture. (This means every time the flow changes.) Explain what is happening between the pictures at one point in each of the examples.

5. You do not have to prove anything or explain why the algorithm works. In particular, you do not have to state or prove the max-flow min-cut theorem. You do not have to estimate the complexity of the algorithm.
HT. Complete Matching: Hall’s Theorem

1. List some real-life examples of the problem.

2. Formulate the problem in graph theoretical terms. Before you do this you might have to introduce some new terminology that is not covered in class.

3. State the theorem and present two different proofs that do not use the max-flow min-cut theorem. Note that the proof in Section 13.4 of Grimaldi [1] uses the max-flow min-cut theorem. You may want to use the other references below. Also note that you will not need all the material presented in Section 13.4 of Grimaldi in this context.

4. At least one of your proofs should suggest an algorithm. Demonstrate how this algorithm works on a graph with $|V| \geq 12$ and $|E| \geq 18$.

5. Give an example of a problem that is not stated directly in graph theoretical terms and can be solved using the theorem. (Suggestion: An $8 \times 8$ chessboard has exactly three pieces on each row and each column. Show that some of these pieces can be removed in such a way that there is exactly one piece left on each row and on each column.)

References:


The following rules apply to all project reports:

- Your project must be submitted in an OpenOffice readable format via email or on CD in addition to a hard copy. Each page must be numbered and have your name on it. The hard copy must be one-sided.

- You may want to use uDraw(Graph), a useful and easy to use freeware for drawing graphs.
• In terms of notation, terminology and style, your report must follow the textbook of this course [1].

• In each topic the conditions on the order and the size of the graphs are given as rough guides, but certainly are not sufficient to make your examples nontrivial and interesting. Your examples must be nontrivial and interesting.

• Whenever you use material from books, papers or the web, this material must be indicated clearly and the source must be acknowledged explicitly. In particular, your report must contain a list of references. You may directly quote definitions, algorithms and theorems from the textbook [1] by citing it. In case you use definitions, algorithms or theorems from other sources, you have to paraphrase them to agree with the textbook’s notation and terminology; again, you must credit the source explicitly. However, the examples in your project must belong to you.

References: