Introduction to Analysis of Algorithms	Prelim 2 Review
CS4820 Spring 2013	Friday, April 5, 2013

The general comments from the Prelim 1 review, available on the handouts page http://www.cs.cornell.edu/Courses/cs4820/2013sp/handouts.htm, still apply.

Prelim 2 is cumulative, but will focus on topics covered since Prelim 1:

1. Flow problems and algorithms: K&T Ch. 7 and handouts http://www.cs.cornell.edu/Courses/cs4820/2013sp/Handouts/EdmondsKarp.pdf, http://www.cs.cornell.edu/Courses/cs4820/2013sp/Handouts/DinicMPM.pdf,
2. NP-completeness: K&T Ch. 8 and handout http://www.cs.cornell.edu/Courses/cs4820/2013sp/Handouts/Reductions.pdf,
3. Basic Turing machine definitions and constructions: handout http://www.cs.cornell.edu/Courses/cs4820/2013sp/Handouts/481TM.pdf, §1-3.

However, you should also review the material covered on the first prelim, as there will be at least one question on the earlier material. For coverage, please see the Prelim 1 review handout http://www.cs.cornell.edu/Courses/cs4820/2013sp/Handouts/review1.pdf.

For flow algorithms, you will need to know

- the formal definition of flow graph and flow;
- the formal definition of a circulation graph and circulation, including circulations with lower bounds;
- what a residual graph is and how to construct one;
- what an augmenting path is and how to find one;
- what a level graph is;
- the various flow algorithms and their complexities;
- how to find a max flow in a given flow graph;
- how to find a min s,t-cut in a given flow graph;
- how to construct a flow graph or a circulation graph for a given flow or circulation problem given in English.

For reductions and NP-completeness, you will need to know

- the formal definition of polynomial-time reduction between decision problems;
- the definition of the class NP;
- the definition of NP-hardness and NP-completeness;
- the definition of various NP-complete problems studied in class, including
  - satisfiability problems: Boolean satisfiability, the definition of conjunctive normal form, CNFSAT, 3CNFSAT (aka 3SAT);
  - graph problems: Clique, Independent Set, Vertex Cover, Dominating Set;
  - covering problems: Set Cover, Exact Cover (XC), Exact Cover by 3-sets (X3C), 3-dimensional matching (3DM);

  - tour problems: directed and undirected Hamiltonian circuit (HC), Traveling Salesperson (TSP);
  - numerical problems: Subset Sum (SS), Partition, Knapsack;
  - Other applications: Bin Packing, Linear and Integer Programming (LP, IP).

- how to do a reduction.

For the last, you will be given a problem B and asked to prove that it is NP-complete. To do this, there are two things you have to do.

1. Show that the problem is in NP by giving a guess-and-verify algorithm for it.
2. Show that the problem is NP-hard by selecting a known NP-hard problem A and reducing A to B. This involves
   (a) describing a polynomial-time transformation σ that constructs an instance $\sigma(x)$ of problem B from a given instance x of problem A;
   (b) showing that $\sigma(x)$ is a "yes" instance of problem B if and only if x is a "yes" instance of problem A.

For Turing machines, you will need to know the basic definitions as given in the handout. You may be asked to construct a Turing machine to perform some simple task. You should also know what Church's Thesis is.