COMPUTATIONAL COMPLEXITY (briefly!)

Reading: 8.1,2,3 in B&M.

Will soon be looking at algorithms, need some concepts to say whether an algorithm is efficient or not, whether problem is computationally difficult, and so on.

- *decision problem*: set of instances, question requiring yes/no answer. Examples:
  HAMILTON CYCLE: Given graph G, does it have a hamilton cycle?
  MAX CLIQUE: Given a graph G and integer k, does G have a clique of at least k vertices? Note have to introduce k to make it a decision problem.
- *polynomial time* or *good* algorithm: number of steps always bounded by some polynomial in size (number of bits) of input. As opposed to algorithms that take, for example, exponential or factorial amount of time.
- $\mathcal{P}$: decision problems solvable by polynomial time algorithm. E.g., is a graph eulerian? Just apply algorithm to find euler tour; if works say yes, otherwise no. Or check connected (polynomial time, will see later) and all degrees even. Problems in $\mathcal{P}$ are in some sense easy.
- $\mathcal{NP}$: decision problems where for each 'yes' answer there exists a *certificate* allowing the 'yes' answer to be verified in polynomial time. Includes $\mathcal{P}$. E.g., HAMILTON CYCLE: certify yes answer by giving a hamilton cycle.
- A problem Q in $\mathcal{NP}$ is $\mathcal{NP}$*-complete* if (loosely) the existence of a polynomial time algorithm for Q would imply the existence of a polynomial time algorithm for every problem in $\mathcal{NP}$. E.g. HAMILTON CYCLE. $\mathcal{NP}$-complete problems are expected not to have polynomial time algorithms and are considered difficult (Millenium Prize problem: is $\mathcal{P}$=$\mathcal{NP}$?). Fact that $\mathcal{NP}$-complete problems exist is nontrivial and deep: Cook (1971) found first one, satisfiability.
- $\mathcal{NP}$*-hard* problem: not necessarily a decision problem, at least as hard as some $\mathcal{NP}$-complete problem. E.g., *find* a hamilton cycle in a graph if one exists.