1 Introduction

- Until now we have been designing algorithms for specific problems
  - We have seen running times $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^3)$...
  - We have also discussed lower bounds (for comparison based sorting).

- It is natural to ask if we can somehow classify problems according to their hardness
  - What we call complexity theory.

- First natural question to ask is if there are problems we cannot solve at all?
  - Yes, e.g. Turing halting problem (deciding if a given Turing machine halts on every input)

- We often think about problems we can solve in polynomial time $O(n^k)$ as being practically solvable
  - We have seen a lot of those, e.g Shortest Path, Minimum Spanning Tree, ...

- Similarly we think about problems we need exponential time $O(2^n)$ to solve as being practically unsolvable.

- We would like to be able to prove if a problem is practically solvable without actually having to develop an $O(n^k)$ algorithm or proving a $\Omega(2^n)$ lower bound!

- As we will discuss there is a huge class of problems for which we do not know if they are practically solvable or not
  - On the other hand, we can identify a subclass for which we have strong evidence that problems in it are practically unsolvable

2 Decision problems

- In order to keep things simple we will focus on decision problems:
  - Problems for which the output is Yes or No.
  - We say that an algorithm for a decision problem accepts an input if it answers Yes on the input.
• Examples:
  – Is input sorted?
  – Is graph acyclic?
  – Is there a shortest path of length \(< k\) between \(u\) and \(v\) in graph?

• In terms of proving lower bounds, considering decision problems do not really restrict us—
decision problems often easier than corresponding optimization problem
  – e.g., we can decide if there is a shortest path of length \(< k\) between \(u\) and \(v\) by actually
computing the shortest path and comparing its length to \(k\)
  \[\downarrow\]
  lower bound on decision problem gives lower bound on optimization problem.

3 \(P, NP\) and \(EXP\)

• We are now ready to define our complexity classes a little more formally
  – Note that if we should really do it right we would have to introduce a lot more formalism
    (out of the scope of this class - note however that CLRS goes into somewhat more detail
    than we do here)

  \[EXP = \{\text{Decision problems solvable in exponential time}\}\]
  \[P = \{\text{Decision problems solvable in polynomial time}\}\]

  – Note that for a given decision problem, \(k\) in the polynomial bound \(O(n^k)\) cannot depend
    on the problem instance.

• In order to investigate the relationship between \(EXP\) and \(P\) we define another class

  \[NP = \{\text{Decision problems for which we given a Yes solution can verify in polynomial time that it is correct}\}\]

• Examples of problems in \(NP\)
  – Is there a path of length \(< k\) between \(u\) and \(v\)? Given path we can easily verify if it
    really has length \(< k\).
  – Hamiltonian cycle problem: Is there a simple cycle containing all vertices? Again
easy to verify solution.
• Note: $N$ in $NP$ really stands for “non-deterministic”
  - If we can “guess” the solution we can solve the problem in polynomial time.

• $P \subseteq NP \subseteq EXP$
  - $P \subseteq NP$ obvious.
  - $NP \subseteq EXP$ since we can enumerate all the (exponential number of) possible solutions to the problem and check each of them in polynomial time.

• The big question is if $P = NP$?
  - Intuitively no—often much harder to solve problem than to verify a solution.
  - We really do not have any clue if $P = NP$ or not, or $NP = EXP$, or ..., but we have strong evidence that there is a core of problems in $NP$ that are not in $P$. We call this class of problems $NPC$.

4 Polynomial time reduction

• In order to define $NPC$ we need the notion of polynomial time reductions
  - Just the idea of using the solution to one problem to solve another

• $X \leq_P Y$:

  A problem $X$ is polynomial time reducible to a problem $Y$ ($X \leq_P Y$) if we can solve $X$ in a polynomial number of calls to an algorithm for $Y$ (and the instance of problem $Y$ we solve can be computed in polynomial time from the instance of problem $X$).

• Note: $X \leq_P Y$ and $Y \in P \Rightarrow X \in P$
  - Explains $\leq_P$ notation: $X \leq_P Y$, “$X$ not more than a polynomial factor harder than $Y$”

• Examples:
  - Traveling Salesman problem (TSP): Given a complete (edges between every pair of vertices) weighted undirected graph $G = (V, E)$, find the minimal weight simple cycle that visits every vertex in $V$.
  - Decision problem version of TSP: Is there a TSP path/tour of weight $< k$?
  - Hamiltonian cycle $\leq_P$ TSP:

    * Proof: Let all edges in the graph we want to solve Hamiltonian cycle for have weight 0. Make graph complete by adding edges with weight 1. Run TSP algorithm. The graph has a Hamiltonian cycle if and only if it has a TSP tour of weight 0.

5 $NP$-completeness

• We are now ready to define $NPC$

  A problem $Y$ is in $NP$ (it is $NP$-complete) if
  a) $Y \in NP$
  b) $X \leq_P Y$ for all $X \in NP$
Note: The problems in \textit{NPC} are the “hardest” problems in \textit{NP}

The following Theorem formalizes this and explains why \textit{NPC} is an important class:

\begin{center}
\textbf{Theorem:}
\begin{itemize}
  \item[a)] If any problem in \textit{NPC} is in \textit{P} then \textit{P} = \textit{NP}
  \item[b)] If any problem in \textit{NP} is not in \textit{P} then \textit{NPC} \cap \textit{P} = \emptyset
\end{itemize}
\end{center}

\textbf{Proof:}
\begin{itemize}
  \item[a)] \(Y \in P \cap NPC \Rightarrow\) for all \(X \in NP\) we have \(X \leq_p Y \Rightarrow X \in P\)
  \item[b)] We have \(X \in NP\) and \(X \notin P\). Assume \(Y \in NPC \cap P\). As \(X \leq_p Y\) we have \(X \in P\), which is a contradiction.
\end{itemize}

\begin{itemize}
  \item Note:
  \begin{itemize}
    \item The above theorem is the reason why we focus on the problems in \textit{NPC}.
    \item We think the world looks like this—but we really do not know:
  \end{itemize}
\end{itemize}

\begin{center}
\textbf{NP=EXP}
\textbf{P} \quad \textbf{NPC}
\end{center}

\begin{itemize}
  \item If someone found a polynomial time solution to a problem in \textit{NPC} our world would “collapse” and a lot of smart people have tried really hard to solve \textit{NPC} problems efficiently
\end{itemize}

\begin{center}
\textbf{\downarrow}
\end{center}

We regard \(Y \in NPC\) a strong evidence for \(Y\) being hard!

\begin{itemize}
  \item But how do we know that there are actually any problem in \textit{NPC}?
  \begin{itemize}
    \item If we can just find one problem in \textit{NPC} the following lemma helps us to find more:
  \end{itemize}
\end{itemize}

\begin{center}
\textbf{Lemma: If} \(Y \in NP\) and \(X \leq_p Y\) for some \(X \in NPC\) then \(Y \in NPC\)
\end{center}

\textbf{Proof:}
\begin{itemize}
  \item[a)] \(Y \in NP\)
  \item[b)] For all \(Z \in NP\) we have \(Z \leq_p X\) which means that \(Z \leq_p Y\) \(Z \leq_p X \leq_p Y\)
\end{itemize}

\begin{itemize}
  \item The lemma shows that we just need to prove \(Y \in NP\) (easy) and reduce problem in \textit{NPC} to \(Y\) to prove that \(Y\) is in \textit{NPC}
  \begin{itemize}
    \item We do not have to prove lower bound!
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item Finding the first problem in \textit{NPC} is somewhat difficult and require quite a lot of formalism
  \begin{itemize}
    \item The first problem proved to be in \textit{NPC} was \textbf{SAT}: Give a boolean formula, is there an assignment of true and false to the variables that makes the formula true?
    \begin{itemize}
      \item For example:
      \begin{center}
      \begin{align*}
        &x_{10} \land (x_4 \iff \neg x_3) \land (x_5 \iff (x_1 \lor x_2)) \land (x_6 \iff \neg x_4) \land (x_7 \iff (x_1 \land x_2 \land x_4)) \land (x_8 \iff (x_5 \lor x_6)) \land (x_9 \iff (x_6 \lor x_7) \land (x_{10} \iff (x_7 \land x_8 \land x_9))
      \end{align*}
      \end{center}
      be satisfied?
    \end{itemize}
  \end{itemize}
\end{itemize}
• By now a lot of problems have been proved $NP$-complete using the lemma:
  
  – e.g. HAMILTONIAN CYCLE, TSP, …
  – Whole books with $NPC$ problems have been written.
  – Next time we will look at some of the $NPC$ proofs.

\[ \]

We really think problems in $NPC$ do not have polynomial time solutions (they are hard!). Nevertheless, every year someone claims to have found a polynomial time solution to a problem in $NPC$ ... until now they have all been wrong.