Algorithms

DATE: 10/5/22

HOW TO DEAL WITH "HARD" PROBLEMS, THAT IS, PROBLEMS THAT MIGHT NOT ADMIT POLYTIME ALGORITHM?

Consider the following algorithms we studied:

- we proved that the shortest path problem on graphs with positive weights can be solved in $O((n+n)\log n)$ with DIJKSTRA ALGORITHM;
- we proved that sorting an array can be done in $O(n \log n)$ through HEAPSORT or MERGESORT.
- we proved that interval scheduling can be solved in $O(n \log n)$
- ...

Each of the above problems can be solved efficiently, in polytime.

We also solved some generalizations of the above probs:

- Shortest path on general weighted graphs can be solved in $O(n^3)$;
- Weighted int. scheduling can also be solved in $O(n \log n)$.
- ...

There are many possible generalizations of the above problems:

- INDEPENDENT SET

  If $G(V,E)$ is a graph, then $S \subseteq V$ is an independent set of the graph iff there are NO edges between the nodes of S.

THE I.S. PROBLEM:

Given $G(V,E)$, find a largest independent set.

INTERVAL SCHEDULING

independent set nodes
vertex cover nodes.

That is, if we could solve independence set, we could also solve interval scheduling.

Thus, independence set is more general than interval scheduling.

- VERTEX COVER

A vertex cover of an und. graph $G(V,E)$ is a set $S \subseteq V$ such that each edge $e \in E$ is incident on at least one node of S.

THE VC PROBLEM:

- Given $G(V,E)$ find a smallest VC.

As we will see, these two problems are "hard" (we believe they cannot be solved in polytime).

In other words, here, we are looking for negative answers to the question "does there exists an efficient algorithm?"

Negative answers are useful for a number of reasons:

1. A negative answer can stop you from wasting time;
2. A " " " can give you a reason to simplify your problem, or to use heuristics.

The most important tool CS has for indicating that no efficient algorithms exist is the "reduction".

Reductions are based on the following idea:

- Suppose that there exists an **oracle** (a black-box, or a function) that, magically, in polynomial time solves instances of problem X;
- Suppose that you devise a **reduction** (an algorithm) that, with polynomially many calls to the **oracle** for X, plus some additional polynomial time computation, solves generic instances of problem Y.

Then, using your reduction together with the oracle for X, you can efficiently solve Y.

DEF: For a pair of problems X and Y as above, we write $Y \leq_p X$ if the above reduction exists.

We read "$Y \leq_p X$" as "Y is polynomial-type reducible to X", or "X is at least as hard as Y (w.r.t. polynomial time".

L1: If $Y \leq_p X$, and X can be solved in polynomial time, then Y = = = = = =

P: Each call to the X-oracle can be answered in polynomial time (according to L1). That is, $\exists\, c \geq 0$ s.t. for an instance I of problem X the call to "oracle(I)" can be answered in time $O(|I|^c)$.

(If $n = |I|$, then $O(|I|^c) = O(n^c)$).

The reduction (which exists since $Y \leq_p X$) makes at most $O(n^{c'})$ calls to the oracle, for some $c' \geq 0$.

The reduction might also make some other computation, for a time of $O(n^{c''})$, for $c'' \geq 0$.

Then everything can be implemented with a runtime of

$$O(n^{c'}) \cdot O(n^c) + O(n^{c''}) = O(n^{c'+c} + n^{c''}) = O(n^{\max(c'+c,\, c'')}) \quad \blacksquare$$

L2: If $Y \leq_p X$, and Y cannot be solved in polynomial time, then X " " " " " time.

P: Apply L1.

Let us use this idea on independent set and vertex cover.

Up until now, we have no idea how hard they are. We will show that they share the same "hardness".

L: Independent Set $\leq_p$ Vertex Cover

L: Vertex Cover $\leq_p$ Independent Set

T: Let $G(V,E)$ be an undirected graph.
Then, $S \subseteq V$ is an Independent Set IFF $V - S$ is a Vertex Cover.

P: First, assume that S is and Ind. Set.

$\nexists \{u,v\} \in E$ s.t. $\{u,v\} \subseteq S$.

That is, $\forall \{u,v\} \in E$, either $u \notin S$, or $v \notin S$ or $u, v \notin S$.

If $w \in V$ is s.t. $w \notin S$ then $w \in V-S$.

Thus, $\forall \{u,v\} \in E$, either $u \in V-S$, or $v \in V-S$, or $u, v \in V-S$
Thus, $V-S$ is a vertex cover.

Now, assume that $V-S$ is a vertex cover.

Then, if $u, v \notin V-S$, then $\{u,v\} \notin E$.
And, $u, v \in V-S$ is equivalent to $u, v \in S$.

That is, for any two $u, v \in S$, $\{u,v\} \in E$ - then S is an Ind. Set ■

L: Independent Set $\leq_p$ Vertex Cover

P: Suppose we have an oracle for vertex cover. Then, to determine whether $G(V,E)$ has an independent set of k nodes, I can ask the oracle whether $G(V,E)$ has a vertex cover of $|V| - k$ nodes. ■

L: Vertex Cover $\leq_p$ Independent Set

P: Omitted (same as above.)

$\min VC \leq K \iff \max I.S. \geq n-k$

Vertex Cover and Independence Set are equivalent. (w.r.t. polynomial time)