Understanding unsolvable problem

Jonathan Irvin GUNAWAN

National University of Singapore

jonathan.irvin@yahoo.com

IOI 2016 - August 14, 2016

Introduction

Problem 'Friend' in IOI 2014

- Find a set of vertices with maximum total weight, such that no two vertices in the set are sharing a common edge.
- Classical Weighted Maximum Independent Set problem.
- Weighted Maximum Independent Set problem is NP-hard by reduction from 3-SAT.
- No one has been able to propose a solution to any NP-hard problem in polynomial time since 4 decades ago.
- None of the Indonesian IOI 2014 team were aware of the fact that Maximum Independent Set is an NP-hard problem, and thus were stuck trying to solve a general Maximum Independent Set problem.
- Similar case also occurs in IOI 2008 'Island' problem.

Introduction

- Generally, some contestants spend too much thinking time trying to solve something that is believed to be unsolvable.
- If only they realise that their attempt is intractable, they may try a different approach and find a special case of this problem.

Identifying Intractability of a Problem through Reduction

We would like to know that the problem that we are attempting is unlikely to have an immediate solution.

- The most common way is to apply a well-known technique called reduction.
- Suppose we know that problem X is impossible to solve, and we also know that we can solve problem X by using problem Y as a black-box[1].
- If we can solve problem Y, then we can solve problem X as well.
- Therefore, problem Y is also impossible to solve.

[1] We say that problem X is reducible to problem Y

Identifying Intractability of a Problem through Reduction

We are using NP-hard problems for illustration.

- MIN-VERTEX-COVER is a graph problem that involves finding a minimum subset of nodes such that for every edge, at least one of its endpoint is in the subset.
- MAX-INDEPENDENT-SET is a graph problem of finding a maximum subset of nodes such that for every edge, at most one of its endpoint is in the subset.
- Suppose we already know that MIN-VERTEX-COVER is a NP-hard problem.
- Therefore, we can show that MAX-INDEPENDENT-SET is also a NP-hard problem by reducing a MIN-VERTEX-COVER problem into a MAX-INDEPENDENT-SET problem.

Identifying Intractability of a Problem through Reduction

Lemma (1)

If $S \subseteq V$ is an INDEPENDENT-SET of graph $G(V, E)$, then $V - S$ is a VERTEX-COVER of the graph $G(V, E)$

Proof.

Let us assume that $V - S$ is not a VERTEX-COVER. Therefore, there are two vertices $A, B \notin V - S$ and there is an edge connecting A and B. Since $A, B \notin V - S$, we have $A, B \in S$. As A and B are connected by an edge, we note that S is not an INDEPENDENT-SET. This is a contradiction. Therefore $V - S$ is a VERTEX-COVER. □

Identifying Intractability of a Problem through Reduction

Lemma (2)

If $S \subseteq V$ is a VERTEX-COVER of graph $G(V, E)$, then $V - S$ is an INDEPENDENT-SET of the graph $G(V, E)$

Proof.

The proof is actually similar to the previous lemma. Let us assume that $V - S$ is not a INDEPENDENT-SET. Therefore, there are two vertices $A, B \in V - S$ and there is an edge connecting A and B. Since $A, B \in V - S$, we have $A, B \notin S$. As A and B are connected by an edge, we note that S is not a VERTEX-COVER. This is a contradiction. Therefore $V - S$ is an INDEPENDENT-SET. □

Identifying Intractability of a Problem through Reduction

Theorem (1)

If $S \subseteq V$ is a MAX-INDEPENDENT-SET of graph $G(V, E)$, then $V - S$ is a MIN-VERTEX-COVER of the graph $G(V, E)$.

Proof.

We know that $V - S$ is a vertex cover by lemma 1. The only thing that remains for us to prove is its minimality. Suppose $V - S$ is not minimum vertex cover. Then, there is another vertex cover $V - S'$ where $|V - S'| < |V - S|$, which implies that $|S'| > |S|$. By lemma 2, S' is an independent set. Therefore, S is not a maximum independent set. This is a contradiction. Therefore $V - S$ is the minimum vertex cover. □

From the above theorem, we conclude that a MIN-VERTEX-COVER can be easily constructed if we have a MAX-INDEPENDENT-SET.

Identifying Intractability of a Problem through Reduction

- In the beginning of this section, we assume we know that MIN-VERTEX-COVER is a NP-hard problem.
- It is good to know as many NP-hard problem as possible.
- This is necessary so that if we encounter a new problem X, we can use any of the NP-hard problems that we know, reduce it to problem X, and thus prove that X is also NP-hard.

How to proceed

Suppose we already know that a problem is unsolvable (i.e. any known algorithm will not solve this problem in time).

- In competition, it is impossible to complain that "This is unsolvable, can you eliminate this problem?" to the judges, since the judges believe they have a solution.
- Such a request is absurd when there are already several contestants who have solved that problem.
- Also, in a major competition (e.g. ACM International Collegiate Programming Contest World Finals, IOI), it is unlikely that the judges have incorrect solution.

How to proceed - Approximation

In real life, when we cannot find the optimal solution, we can try to find the solution that is close to the optimal solution.

- A solution that is not larger than α ($\alpha > 1$) times the optimal solution for a minimisation problem.
- The most common approximation algorithm I found in textbooks is the 2-approximation MIN-VERTEX-COVER problem, which means that the algorithm will not choose more than twice the number of vertices than the optimal solution.
- Rarely occur in competitive programming (especially IOI) since to create this kind of problem, the judges have to know the optimal solution in order to verify that the contestants solution is indeed α-approximation. Generating the optimal solution takes a long time.
- We will not discuss this approach in detail.

How to proceed - Pruning

This approach is useful in some competitive programming problems.

- ACM International Collegiate Programming Contest (ICPC) World Finals 2010, problem I (Robots on Ice) required the contestant to count the number of Hamiltonian Paths with constraints, which is known to be NP-hard problem.
- While finding all possible paths is impossible, the solution for this problem is to prune the exponential algorithm we use to find all possible paths.
- If at some point we know that it is impossible to visit the rest of the unvisited points, then we can prune the path and backtrack immediately.
- Not very suitable for IOI. Usually, IOI problems require deep analysis from the contestant. It is rare that we can get Accepted by only "hacking" a complete search algorithm.
- We will also not discuss this approach in detail.

How to proceed - Finding special cases

- The most suitable approach in IOI, and thus is the main focus of this paper.
- The approach needed to solve IOI 2008 Island and IOI 2014 Friend.
- Find a special constraint in the problem such that this constraint allows the problem to be solvable in polynomial time.
- We can check whether an additional constraint causes a problem to be solvable in polynomial time using the reduction proof of the original problem (without the additional constraint), and check whether the proof still holds given the additional constraint to the problem.
- We will give one example of a special case in a NP-hard problem.

How to proceed - Finding special cases

- Suppose we have a function with the following formula

$$f(n) = \begin{cases} 1, & n = 1 \vee n = 2 \\ f(n-1) + f(n-2), & n > 2 \end{cases}$$

- We consider the sequence $F = \{f(n)\}_{n=1}^{\infty}$.
- We define $F(N)$ to be the first N terms of F.
- We want to know whether we can create a partition of $F(N)$ into two disjoint multisets A and B such that the sum of all elements in A is equal to the sum of all elements in B
- This looks like a classic PARTITION problem. PARTITION problem is NP-hard by reduction from SUBSET-SUM.
- However, this sequence is defined in a very special way, in the sense that F is defined using the aforementioned recurrence.
- Therefore, we should inspect the recurrence formula more closely.

How to proceed - Finding special cases

Lemma (3)

Any consecutive subsequence of F with length multiples of three can be partitioned into two multisets of equal sum.

How to proceed - Finding special cases

Proof.

Pick any consecutive subsequence of F with length multiples of three, which we will denote by $F' = \{f(a), f(a+1), f(a+2), ...f(b)\}$ for some $a < b$. We can partition F' into

$$A = \{f(a+3k), 0 \leq k \leq \frac{b-a-2}{3}\} \cup \{f(a+1+3k), 0 \leq k \leq \frac{b-a-2}{3}\}$$

$$B = \{f(a+2+3k), 0 \leq k \leq \frac{b-a-2}{3}\}.$$

A and B will have the same sum, as for every $0 \leq k \leq \frac{b-a-2}{3}$

$$f(a+3k) + f(a+1+3k) = f(a+2+3k)$$

by the construction of the function. □

How to proceed - Finding special cases

Theorem (2)

If N is divisible by three, then $F(N)$ can be partitioned into two multisets of equal sum.

Proof.

If N is divisible by three, then $F(N)$ is a prefix of F with length multiples of three. By lemma 4, $F(N)$ can be partitioned into two multisets of equal sum. □

How to proceed - Finding special cases

Theorem (3)

If $N \equiv 1 \pmod 3$, $F(N)$ *cannot be partitioned into two multisets of equal sum.*

Proof.

Suppose $F'(N) = F(N) - f(1)$. Note that $F'(N)$ contains $N - 1$ elements. Since $N \equiv 1 \pmod 3$, we have$N - 1 \equiv 0 \pmod 3$. By lemma 4, $F'(N)$ can be partitioned into two multisets of equal sum. Therefore, the sum of all elements in $F'(N)$ is even. However, the sum of all elements in $F(N) = F'(N) + f(1)$, which is odd because $F'(N)$ is even while $f(1)$ is odd. Therefore, there is no way to partition $F(N)$. □

How to proceed - Finding special cases

Theorem (4)

If $N \equiv 2 \pmod 3$. $F(N)$ can be partitioned into two multisets of equal sum.

Proof.

Assign $f(1)$ to A and $f(2)$ to B. Since $f(1) = f(2)$, we are now trying to partition $F'(N) = F(N) - f(1) - f(2)$. $F'(N)$ will have $N - 2$ elements. Since $N \equiv 2 \pmod 3$, we obtain $N - 2 \equiv 0 \pmod 3$. By lemma 4, $F'(N)$ can be partitioned into two multisets of equal sum, which implies our theorem. □

How to proceed - Finding special cases

- Therefore, solving this problem is reduced to checking whether $N \equiv 1 \pmod 3$.
- We can solve this in $O(1)$.
- We will provide more examples of special cases in the following section.

Example of special cases - Planar Graph

Planar graph is a graph that can be drawn on a flat surface without having two edges crossing each other

- Due to the four color theorem, the number of maximum clique in planar graph is less than or equal to 4. We can have a naive $O(V^4)$ solution to find maximum clique, which is a polynomial. We can even improve it to $O(V^2)$.

Example of special cases - Bipartite Graph

Bipartite graph is a graph in which the vertices can be partitioned into two disjoint sets U and V such that all edges connect a vertex from U and a vertex from V.

- Some problems have a bipartite graph as an input although the problem statement does not explicitly state that the given input graph must be bipartite.
- Bipartite graph and bipartite matching was recently included in IOI 2015 syllabus.
- By Konig's Theorem, finding minimum vertex cover and maximum independent set in bipartite graph is equivalent to finding the maximum matching. Therefore, both problems can be solved in $O(V^3)$ which is polynomial.
- Finding the maximum matching in bipartite graph is much easier than in general graph.

Conclusion

- In conclusion, we can use a well-known reduction technique to prove that a problem that we are currently attempting to solve is impossible.
- In competitive programming (including IOI), understanding this technique is essential so that we will not be stuck at trying to solve an impossible problem, thus prompting us to find another way to solve the problem.
- To prove that a problem is NP-hard, it is good to know as many NP-hard problems as possible, so that we can reduce from any one of the problems that we know to the new problem.
- Some of the classic NP-hard problems include 3-SAT, Vertex Cover, Independent Set, and Subset Sum.

Conclusion

- After realizing that the problem is NP-hard, we must be able to find the special case that makes the problem solvable.
- We must be able to find a special property that breaks the reduction proof.
- Having a lot of practice on these kind of problems will help us to familiarize with the possibilities of a special case.
- Some of the common special cases include planar, bipartite, and directed acyclic graph.

Thank You