CS170Hw11

Trustin Nguyen

November 15, 2024

Study Group

Decision vs Search vs Optimization

Recall that a vertex cover is a set of vertices in a graph such that every edge is adjacent to at least one vertex in this set.

The following are three formulations of the VERTEX COVER problem:

- As a *decision problem*: Given a graph G, return TRUE if it has a vertex cover of size at most b, and FALSE otherwise.
- As a *search problem*: Given a graph G, find a vertex cover of size at most b (that is, return the actual vertices), or report that none exists.
- As an *optimization problem*: Given a graph G, find a minimum vertex cover.

At first glance, it may seem that search should be harder than decision, and that optimization should be even harder. We will show that if any one can be solved in polynomial time, so can the others.

(a) Suppose you are handed a black box that solves VERTEX COVER (DECISION) in polynomial time. Give an algorithm that solves VERTEX COVER (SEARCH) in polynomial time.

Answer. The vertex cover problem is in correspondence to 2-SAT. If an edge (u, v) is in the graph, we add the clause $(u \vee v)$ to the chain of clauses. Then the problem reduces to finding b vertices $v_1, v_2, \ldots, v_b$ such that setting the $v_1, \ldots, v_b$ to true satisfies all clauses.

This gives a natural way of solving the problem. We pick an edge of the graph. Say that this edge is (u', v'). Pick one vertex of that edge at random, lets say v'. We remove that vertex and all its adjacent edges (this corresponds to the fact that all clauses containing v' will be satisfied if $v' = \mathrm{True}$).

Now we run VERTEX COVER(DECISION) on the remaining graph with $b - 1$. If it returns true, we know that v' is in a vertex cover, otherwise, we say that u' is in our vertex cover. Repeat until we have no more clauses in our 2-SAT. The vertices we have chosen will make up the vertex cover of size at most b.

(b) Similarly, suppose you are handed a black box that solves VERTEX COVER (SEARCH) in polynomial time. Give an algorithm that solves VERTEX COVER (OPTIMIZATION) in polynomial time.

Answer. We can start with $b = 0$ and run the search algorithm, increasing b by 1 every time until the VERTEX COVER(SEARCH) reports that a vertex cover of size at most b exists. The first instance will be the minimum vertex cover. A speed up would be binary searching.

Some Sums

Given an array $A = [a_1, a_2, \ldots, a_n]$ of non negative integers, consider the following problems:

1 **Partition**: Determine whether there is a subset $S \subseteq [n]$ ($[n] := \{1, 2, \ldots, n\}$) such that $\sum_{i \in S} a_i = \sum_{j \in ([n] \setminus S)} a_j$. In other words, determine whether there is a way to partition A into two disjoint subsets such that the sum of the elements in each subset are equal.

2 **Subset Sum**: Given some integer x, determine whether there is a subset $S \subseteq [n]$ such that $\sum_{i \in S} a_i = x$. In other words, determine whether there is a subset of A such that the sum of its elements is x.

3 **Knapsack**: Given some set of items each with weight w_i and value v_i, and fixed numbers W and V, determine whether there is some subset $S \subseteq [n]$ such that $\sum_{i \in S} w_i \leqslant W$ and $\sum_{i \in S} v_i \geqslant V$.

For each of the following clearly describe your reduction and justify its correctness.

(a) Find a linear time reduction from SUBSET SUM to PARTITION.

Answer. To turn our subset sum to a partition problem, consider the operation of adding the element $2x - \sum_i a_i$ to our set A to get A', and calling

$$\text{PARTITION}(A')$$

The claim is that the truth value of $\text{SUBSET\_SUM}(A, x)$ is the same as the truth value of $\text{PARTITION}(A')$.

– Suppose that we have a valid partition of A'. Then we split A' into sets L, R such that $\sum L = \sum R$. We know that $2x - \sum_i a_i$ can only be in one of L or R. Wlog, say it is not in L. This means that

$$L \subseteq A$$

But

$$\sum L = \frac{1}{2} \cdot \sum A' = \sum A + 2x - \sum_i a_i = \frac{1}{2} \cdot 2x = x$$

so we have found a subset of A that adds to x.

– Suppose there is a subset of A that adds to x. We will show that there is a partition of A'. Let that subset be called $L \subseteq A$. Then we know that:

$$\sum L = x$$

and

$$\sum A' \setminus L = x$$

as $\sum A' = 2x$. So there is a partition of A'.

(b) Find a linear time reduction from SUBSET SUM to KNAPSACK.

Answer. Let $W = V = x$. For each item, let $w_i = v_i = a_i$. Then the claim is that

$$\text{KNAPSACK}(W, V, \{w\}_i, \{v\}_i) = \text{KNAPSACK}(x, x, A, A)$$

is the same as $\text{SUBSET\_SUM}(A, x)$.

– Suppose that we have determined the value of KNAPSACK(x, x, A, A) to be true. Then we know there is some subset $S \subseteq [n]$ such that

$$\sum_{i \in S} a_i \geqslant x$$

and

$$\sum_{i \in S} a_i \leqslant x$$

so

$$\sum_{i \in S} a_i = x$$

– Suppose that we have determined the value of SUBSET_SUM(A, x) to be true. Then we know there is a subset:

$$\sum_{i \in S} a_i = x$$

so it follows

$$\sum_{i \in S} a_i \geqslant x$$

and

$$\sum_{i \in S} a_i \leqslant xl$$

so KNAPSACK(x, x, A, A) is true as well.

Reduction to 3-Coloring

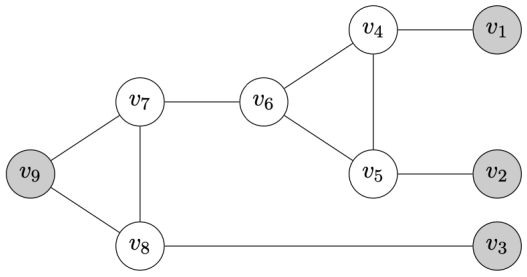
Given a graph $G = (V, E)$, a valid 3-coloring assigns each vertex in the graph a color from {red, green, blue} such that for any edge (u, v), u and v have different colors. In the 3-coloring problem, our goal is to find a valid 3-coloring if one exists. In this problem, we will give a reduction from 3-SAT to the 3-coloring problem. Since we know that 3-SAT is NP-Hard (there is a reduction to 3-SAT from every NP problem), this will show that 3-coloring is NP-Hard (there is a reduction to 3-coloring from every NP problem).

In our reduction, the graph will start with three special vertices, labeled v_{TRUE}, v_{FALSE}, and v_{BASE}, as well as the edges (v_{TRUE}, v_{FALSE}), (v_{TRUE}, v_{BASE}), and (v_{FALSE}, v_{BASE}).

(a) For each variable x_i in a 3-SAT formula, we will create a pair of vertices labeled x_i and $\neg x_i$. How should we add edges to the graph such that in any valid 3-coloring, one of x_i, $\neg x_i$ is assigned the same color as v_{TRUE} and the other is assigned the same color as v_{FALSE}?

Answer. For each pairing $x_i, \neg x_i$, create a vertex y_i and assign it the color v_{BASE}. Now add edges so that the three vertices $x_i, \neg x_i, y_i$ form a triangle. This ensures that either x_i or $\neg x_i$ is assigned the color v_{TRUE} and the other is assigned the color v_{FALSE}.

(b) Consider the following graph, which we will call a "gadget":



Consider any valid 3-coloring of this graph that does *not* assign the color red to any of the gray vertices (v_1, v_2, v_3, v_9). Show that if v_9 is assigned the color blue, then at least one of $\{v_1, v_2, v_3\}$ is assigned the color blue.

Answer. We will prove the contrapositive. If v_1, v_2, v_3 are all non blue, then they must be some other color green. We know then the v_4, v_5 are non green, so v_6 must be green. Now v_7, v_8 are non green so v_9 must be green. So v_9 is non blue.

(c) We have now observed the following about the graph we are creating in the reduction:

(i) For any vertex, if we have the edges (u, v_{FALSE}) and (u, v_{BASE}) in the graph, then in any valid 3-coloring u will be assigned the same color as v_{TRUE}

(ii) Through brute force one can also show that in a gadget, if all the following hold:

(1) All gray vertices are assigned the color green or blue.

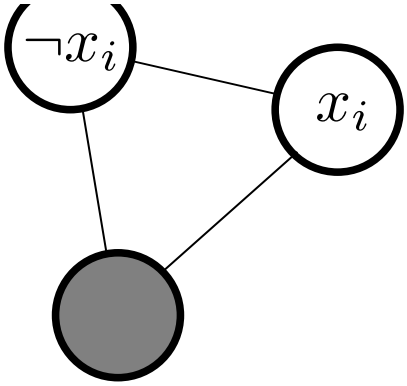
(2) v_9 is assigned the color blue.

(3) At least one of $\{v_1, v_2, v_3\}$ is assigned the color blue.

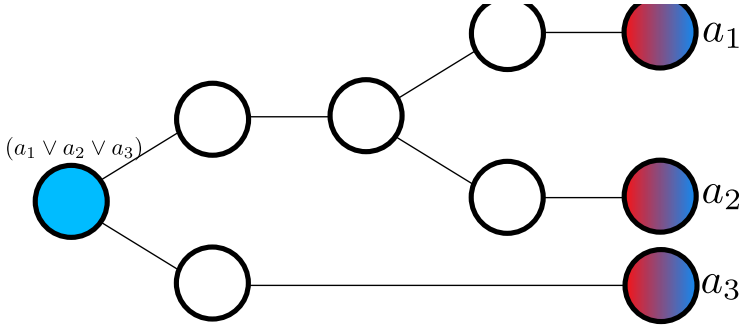
Then there is a valid coloring for the white vertices in the gadget.

Using these observations and your answers to the previous parts, give a reduction from 3-SAT to 3 coloring. Prove that your reduction is correct (you do not need to prove any of the observations above).

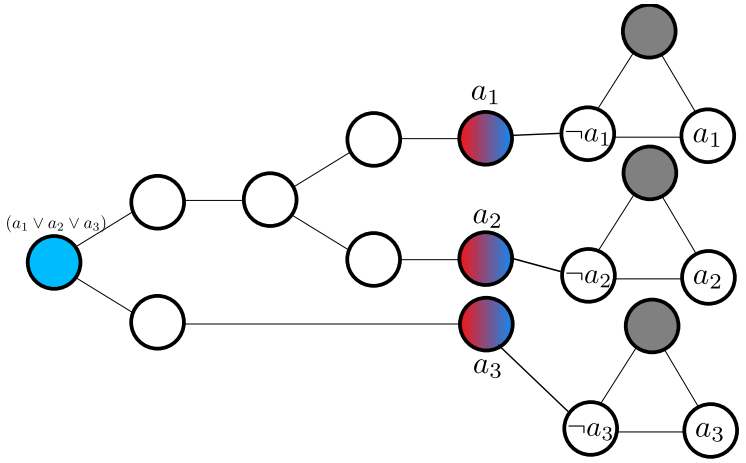
Answer. To solve this, for every variable x_i in our $3-SAT$, start constructing the graph by adding components of the form:



where the third node is fixed at some color (gray) other than red and blue. Now for each clause in our $3-SAT$, lets say $(a_1 \vee a_2 \vee a_3)$ add a gadget:



with the left most node fixed at color blue, and impose that the right nodes are either red or blue (not gray). Connect the rightmost nodes to their negation nodes in the triangle components that we made earlier. For example:



Now we claim that solving for a valid coloring of the graph is equivalent to solving the $3-SAT$.

We have described a way to get from 3-SAT to 3-coloring, and now to show that a solution on the 3-coloring gives a solution on the 3-SAT.

– Suppose there is a valid 3-coloring of the graph. Then for every gadget, the v_9 node is blue and one of the v_1, v_2, v_3 nodes are blue also. Then this defines a correspondence to the truth value of a clause.

  So if we have a valid 3-coloring, the 3-SAT is satisfied:

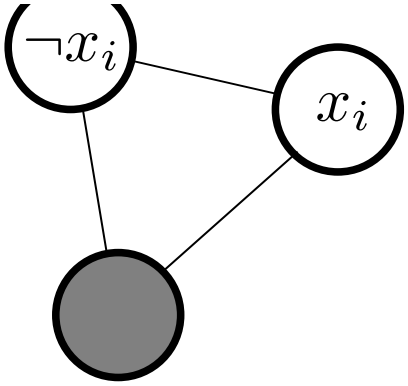
  1 Each clause is colored blue (Represents that the clause is evaluated to true)

  2 One of the three nodes v_1, v_2, v_3 are true in each gadget (Represents that at least one literal in the clause is true)

  3 None of the pairs $a_i, \neg a_i$ are the same color (Recall we connected each v_1, v_2, v_3 in the gadget to its negation, so they cannot be the same color).

– Now suppose that we are given a 3-SAT that evaluates to true and the truth values of each literal. We need to show that the corresponding graph we constructed has a valid 3-coloring.

  Assign the color red to mean the truth value of false and blue to represent true. By the brute force observation, all gadgets have a valid coloring. We also know that none of the pairs $a_i, \neg a_i$ are the same color. This means there is a valid coloring on every 3 component:



  Finally, there is a valid coloring on the union of the gadget graph and 3 component graph because x_i and $\neg x_i$ are different colors (same reason).

This concludes the proof.