Lecture 16: NP-Completeness

Introduction

- NP-hardness and NP-completeness
- 3SAT
  - ↳ Super Mario Brothers
  - ↳ 3 Dimensional Matching
    - ↳ Subset Sum *(weak)*
      - ↳ Partition *(weak)*
        - ↳ Rectangle Packing *(weak)*
    - ↳ 4-Partition *(strong)*
      - ↳ Rectangle Packing *(strong)*
        - ↳ Jigsaw Puzzles

NP-Hard and NP-Complete problems

Today, we discuss NP-Completeness.

Recall from 6.006:

- **P** = the set of problems that are solvable in polynomial time. If the problem has size n, the problem should be solved in $n^{O(1)}$.
- **NP** = the set of decision problems solvable in nondeterministic polynomial time. The output of these problems is a YES or NO answer. Nondeterministic refers to the fact that a solution can be guessed out of polynomially many options in $O(1)$ time. If any guess is a YES instance, then the nondeterministic algorithm will make that guess.

In this model of nondeterminism, we can assume that all guessing is done first. This is equivalent to finding a polynomial-time verifier of polynomial-size certificates for YES answers.

Note that there is an asymmetry between YES and NO inputs

- A problem X is **NP-complete** if $X \in$NP and X is NP-hard.

- A problem X is NP-hard if every problem $Y \in$NP reduces to X.
  - If P $\neq$ NP, then $X \notin$ P.
- A **reduction** from problem A to problem B is a polynomial-time algorithm that converts inputs to problem A into equivalent inputs to problem B. Equivalent means that both problem A and problem B must output the same YES or NO answer for the input and converted input.
  - If $B \in$ P, then $A \in$ NP
  - If $B \in$ NP, then $A \in$ NP
  - If A is NP-hard, then B is NP-hard.

We can show that problems are NP-complete via the following steps.

1. **Show $X \in$ NP**. Show that $X \in$ NP by finding a nondeterministic algorithm, or giving a valid verifier for a certificate.
2. **Show X is NP-hard**. Reduce from a known NP-complete problem Y to X. This is sufficient because all problems $Z \in$ NP can be reduced to Y, and the reduction demonstrates inputs to Y can be modified to become inputs to X, which implies X is NP-hard. We must demonstrate the following properties for a complete reduction.
   (a) Give an polynomial-time conversion from Y inputs to X inputs.
   (b) If Y's answer is YES, then X's answer is YES.
   (c) If X's answer is YES, then Y's answer is YES.

Finally, a **gadget** transforms features in an input problem to a feature in an output problem.

3SAT

The 3SAT was discovered to be NP-complete by Cook in 1971.

Definition 1. ***3SAT:*** *Given a boolean formula of the form:*

$$(x_1 \vee x_3 \vee \bar{x_6}) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_7}) \wedge \ldots$$

is there an assignment of variables to True and False, such that the entire formula evaluates to True?

We note that a literal is of the form $\{x_i, \bar{x}_i\}$, and both forms of the literal correspond to the variable x_i. A clause is made up of the OR of 3 literals, and a formal is the AND of clauses.

3SAT $\in$ NP because we can create a verifier for a certificate. For a given instance of 3SAT, a certificate corresponds to a list of assignments for each variable, and a verifier can compute whether the instances is satisfied, or can be evaluated to true. Thus the verifier is polynomial time, and the certificate has polynomial length.

It is important to note that this verifier only guarantees that a 3SAT instance is verifiable. To ensure that a 3SAT instance is not verifiable, the algorithm would have to check every variable assignment, which cannot be done in polynomial time.

3SAT is also NP-hard. We give some intuition for this result. Consider any problem in NP. Because it belongs in NP, a nondeterministic polynomial time algorithm exists to solve this problem, or a verifier to check a solution. The verifier is an algorithm that can be implemented as a circuit. Now, the circuit consists of AND, OR and NOT gates, which can be represented as a formula. This formula can be converted to 3SAT form, where each clause has 3 literals, which is equivalent to the original formula. Thus all problems in NP can be converted to 3SAT, and the inputs to the original problem are equivalent to the converted inputs to 3SAT, thus 3SAT is NP-complete.

Super Mario Brothers [Aloupis, Demaine, Guo, Viglietta 2014]

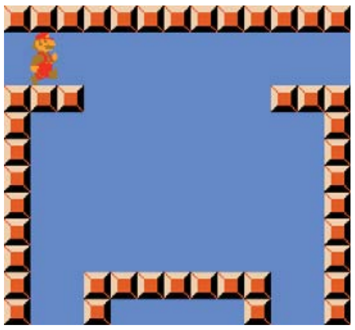
We show that Super Mario Brothers is NP-hard by giving a reduction from 3SAT. This version of Super Mario Brothers is generalized to an arbitrary screen size of $n \times n$, so we remove limits on the number of items on screen. We have the following problem definition

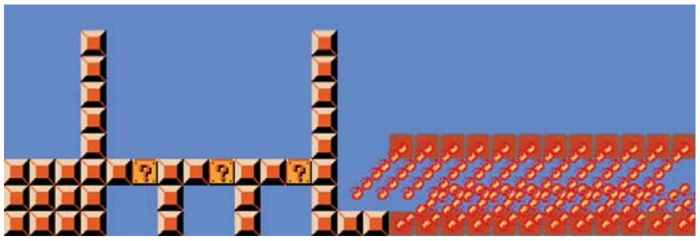
Definition 2. ***Super Mario Brothers:*** *Given a level of Super Mario Brothers, can we advance to the next level?*

Because we reduce from 3SAT, we are given a 3SAT instance, and we must generate a level of Super Mario Brothers that corresponds to that 3SAT instance.

We construct the level by constructing gadgets for each of the variables in the 3SAT formula, as pictured in Figure 1. Mario jumps down from the ledge, and cannot jump back up. He can fall to the left or right, corresponding to assigning the variable to True or False. The remainder of the level is set up such that this choice cannot be reversed.

We also create the following gadget for clauses. After choosing the assignment for a given variable, Mario visits all of the clause gadgets with the same literal value,


(a) Variable gadget


(b) Clause gadget

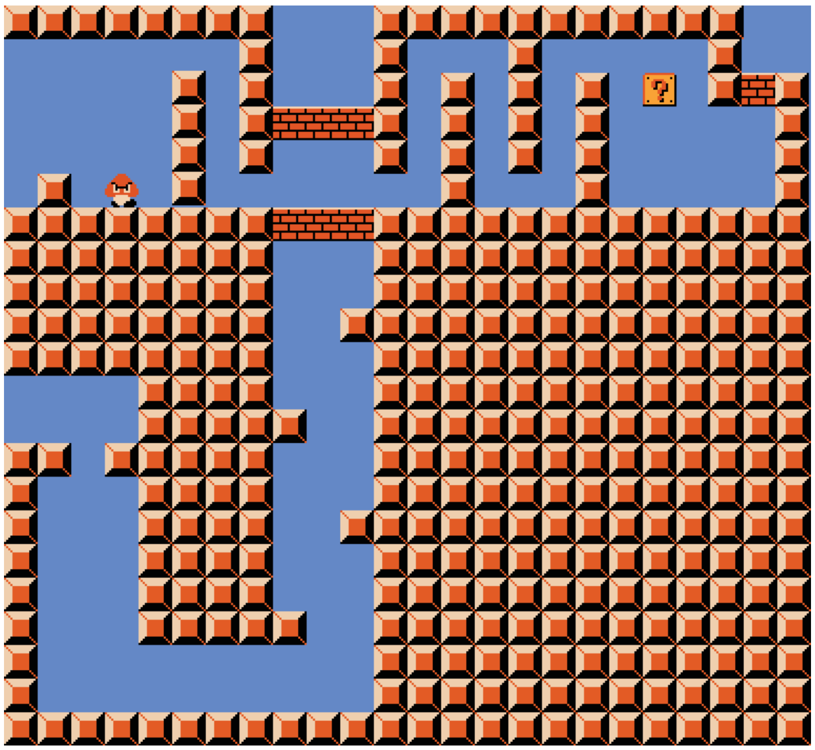

(c) Crossover gadget

Figure 1: Gadgets for Super Mario Brothers.

then moves to the next variable gadget. By visiting a clause, Mario can release a star.

Finally, after visiting all of the variable gadgets, Mario must re-traverse the clause gadgets. If the clause gadget was previously visited, a star is available, and he can pass through the fire. Otherwise, he will not be able to traverse the clause gadget and die.

Thus, winning the level is equivalent to passing through all the clause gadgets on the second pass through. Mario can only pass all the clause gadgets if they have all been satisfied by a variable assignment. The actions throughout the variable gadgets correspond to the solution to the 3SAT formula, so if Mario can pass the level, we have a solution to the 3SAT problem.

The final gadget needed is the crossover gadget. It ensures that Mario does not switch between variable and clause gadgets when it is not allowed. The total size of all these gadgets within the polynomial size required by the reduction.

Thus, Mario can win the level if and only if the original 3SAT formula could be satisfied. Therefore have a reduction from 3SAT, and Super Mario Brothers is NP-hard.

3 Dimensional Matching (3DM)

Definition 3. ***3DM:*** *Given disjoint sets X, Y, and Z, each of n elements and triples $T \subseteq X \times Y \times Z$ is there a subset $S \subseteq T$ such that each element $\in X \cup Y \cup Z$ is in exactly one $s \in S$?*

3DM is NP. Given a certificate, which lists a candidate list of triples, a verifier can check that each triple belongs to T and every element of $X \cup Y \cup Z$ is in one triple.

3DM is also NP-complete, via a reduction from 3SAT. We build gadgets for the variables and clauses.

The variable gadget for variable x_i is displayed in the picture. Only red or blue triangles can be chosen, where the red triangles correspond to the true literal, while the blue triangles correspond to the false literal. Otherwise, there will be overlap, or some inner elements will not be covered. There is an $2n_{x_i}$ "wheel" for each variable gadget, where n_{x_i} corresponds to the number of occurrences of x_i in the formula.

The clause gadget for the clause $x_i \wedge \bar{x}_j \wedge x_k$ is displayed in the picture. The dot that is unshared in each variable's triangle within the clause gadget is also the single dot within the variable gadget.

Then, if we set x_i to true, we take all of the red false triangles in the variable gadget, leaving a blue true triangle available to cover the clause gadget. However, this still potentially leaves $\bar{x}_j$ and x_k uncovered, so we need a garbage collection gadget, pictured below. There are $\sum_x n_x$ clauses of these gadgets, because there are n_x unnecessary elements of the variable gadget that won't be covered. However, of the remaining elements, one of these per clause will be covered, so the remaining need to be covered by the garbage collection clause.

Thus, if a solution exists to the 3SAT formula, we can find the solution to the 3DM problem by choosing the points in 3DM corresponding to the variable values. If we have a 3DM solution, we can map this back to the satisfying assignment for 3SAT. Thus, our reduction is complete and 3DM is NP-hard.

Subset Sum

Definition 4. ***Subset Sum*** *Given n integers $A = \{a_1, a_2, \ldots, a_n\}$ and a target sum t, is there a subset $S \subseteq A$ stuch that*

$$\sum S = \sum_{a_i \in S} a_i = t$$

The Subset Sum problem is NP-complete. It is in NP, because a verifier can simply check that the given subset is a subset of A and that its sum is equivalent to the target in polynomial time.

It is NP-hard via a reduction from 3DM. View the numbers in base $b = 1 + \max_i n_{x_i}$. Then the triple (x_i, x_j, x_k) can be written in base b in the form $000100100001000 = b^i + b^j + b^k$, where the first 1 corresponds to i, the second to j, and the third to k. The target sum $t = 111111111111111 = \sum_i b^i$.

This prevents any 1s from colliding, which ensures that each element is used exactly once, because multiple 1s correspond to reusing an element, and the base is sufficiently large so that no smaller numbers can be summed to create the next power of b. This completes the reduction.

In fact, the Subset Sum problem is only **weakly NP-hard**. The number of digits in t is $O(n)$. This means that the values of the numbers used in the reduction are exponential in input, making this problem weakly NP-hard. Problems which are **strongly NP-hard** must only use number values that are polynomial in the size of the input.

This also implies that the problem can be solved by a pseudopolynomial algorithm.

Partition

Definition 5. ***Partition:*** *Given $A = \{a_1, a_2, \ldots, a_n\}$, is there a subset $S \subseteq A$ such that*

$$\sum S = \sum A \setminus S = \frac{1}{2} \sum A?$$

Partition is also weakly NP-complete. It is a special case of the Subset Sum problem, where we set $t = \frac{1}{2} \sum A$. In fact, we can reduce Partition to Subset Sum, though this is not the direction we want for the reduction.

We can reduce from Subset Sum to Partition as follows. Let $\sigma = \sum A$. Add elements $a_{n+1} = \sigma + t$ and $a_{n+2} = 2\sigma - t$ to A. Then a_{n+1} and a_{n+2} must be on different sides of the partition. In order to balance the two sides, $\sigma + t$ must be added to a_{n+1} and t must be added to a_{n+1}, so each subset has sum 2σ. Thus if we can solve Partition, we also have the subset of elements that sum to t, the target for the Subset Sum problem, completing our reduction.

Rectangle Packing

Definition 6. ***Rectangle Packing:*** *Given a set of rectangles R_i and a target rectangle T, can we pack the rectangles in T such that there is no overlap? Note that sum of the area of the rectangles R_i is equivalent to the area of the target rectangle or $\sum_i R_i = T$.*

Rectangle packing is weakly NP-hard via a reduction from Partition. For every element a_i in Partition, we create a rectangle R_i with height 1 and width $3a_i$. The target rectangle has height 2 and width $3t = \frac{3}{2} \sum A$. Because each rectangle has width at least 3, all rectangles must be packed horizontally. Thus to solve the Rectangle packing problem, we must separate the blocks into two groups with total width $3t$, which will correspond to two subsets with total sum t in the Partition problem.

Jigsaw Puzzles [Demaine & Demaine 2007]

Definition 7. *Given square tiles with no patterns, can these tiles be arranged to fit a target rectangular shape? Note that the tiles can have a side tab, pocket, or boundary, but tabs and pockets must have matching shapes.*

The most obvious reduction is from Partition. For every number, create a set of square rectangles with a unique tab and pocket, where the number of tiles is equivalent to the value of a_i, and the two end pieces of the rectangle have boundaries.

However, this reduction cannot be completed because the inputs to Partition can be exponentially large.

Instead, the reduction comes from 4-Partition.

Definition 8. ***4-Partition:*** *Given n integers $A = \{a_1, a_2, \ldots, a_n\} \in (\frac{t}{5}, \frac{t}{3})$, is there a partition into $\frac{n}{4}$ subsets of 4 elements, each with the same sum $t = \sum A / \frac{n}{4}$?*

4-Partition is a strongly NP-complete problem. We reduce from 4-Partition to Jigsaw Puzzles to show that Jigsaw Puzzles are NP-hard.

For each a_i, we create the following set of pieces. This ensures that we cannot mix pieces from different a_i.

These pieces are set into the following target board. There are $\frac{n}{4}$ rows, each of which will hold pieces corresponding to 4 a_i terms in the 4-Partition problem. The width of the target board is t, because each row must hold t pieces, so that the corresponding a_i form a group of sum t in the 4-Partition problem. Thus, the reduction is complete.