CS120: Intro. to Algorithms and their Limitations Anshu & Hesterberg

Sender–Receiver Exercise 5: Reading for Senders

Harvard SEAS - Fall 2023 *2023-11-16*

The goals of this exercise are:

- to develop your skills at understanding, distilling, and communicating proofs and the conceptual ideas in them,
- to reinforce the definition of NP and practice NP-completeness proofs

Sections 1 is also in the reading for receivers. Your goal will be to communicate the *proof* of Theorem 1.2 (i.e. the content of Section 2) to the receivers.

1 The Result

So far, we have seen examples of NP-complete problems in logic (e.g. SAT) and graph theory (e.g. Independent Set). Here you will see an example of a numerical NP-complete problem.

Input	: Natural numbers $v_0, v_1, \ldots, v_{n-1}, t$
Output	: A subset $S \subseteq [n]$ such that $\sum_{i \in S} v_i = t$, if such a subset S exists

Computational Problem SubsetSum

Example SubsetSum instance: Given the input $(v_0, v_1, \ldots, v_5, t) = (1, 5, 3, 8, 6, 2, 7)$, a solution would be the subset $S = \{0, 4\}$ since $v_0 + v_4 = 1 + 6 = 7 = t$.

Theorem 1.1. *SubsetSum is* $\mathsf{NP}_{\mathsf{search}}$*-complete.*

Actually, we will focus on the *vector* version of the problem, where we replace the v_i's with vectors having $\{0, 1\}$ entries and t with a vector of natural numbers.

Input	: Vectors $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_{n-1} \in \{0,1\}^d$, $\vec{t} \in \mathbb{N}^d$
Output	: A subset $S \subseteq [n]$ such that $\sum_{i \in S} \vec{v}_i = \vec{t}$, if such a subset S exists

Computational Problem VectorSubsetSum

We will use the notation $\vec{v}[j]$ to denote the j'th entry of vector $\vec{v}$, so the condition $\sum_{i \in S} \vec{v}_i = \vec{t}$ means that for every $j = 0, 1, \ldots, d-1$, we have $\sum_{i \in S} \vec{v}_i[j] = \vec{t}[j]$.

Example VectorSubsetSum instance: Consider the 3 vectors $\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3$ and target $\vec{t}$ written in the table below:

	0	1	2	3
$\vec{v}_0$	1	0	0	1
$\vec{v}_1$	1	0	0	1
$\vec{v}_2$	0	1	0	0
$\vec{v}_3$	0	1	1	0
$\vec{t}$	2	1	1	2

A solution is $S = \{0, 1, 3\}$ since $\vec{v}_0 + \vec{v}_1 + \vec{v}_3 = \vec{t}$.

Theorem 1.2. *VectorSubsetSum is* $\mathsf{NP}_{\mathsf{search}}$*-complete.*

Last week's section showed how to derive Theorem 1.1 from Theorem 1.2. In today's SRE, we will prove Theorem 1.2.

2 The Proof

To show that VectorSubsetSum is $\mathsf{NP}_{\mathsf{search}}$-complete, we need to prove that (a) it is in $\mathsf{NP}_{\mathsf{search}}$, and (b) that every problem in $\mathsf{NP}_{\mathsf{search}}$ reduces to VectorSubsetSum in polynomial time.

For (a), we need to show that the solutions to VectorSubsetSum are of polynomial length and that all solutions are verifiable in polynomial time. We observe that solutions (the set S) can be described in n bits, which is shorter than the length of the input $(\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_{n-1}, \vec{t})$, and therefore certainly polynomially bounded. A verifier for solutions $V((\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_{n-1}, \vec{t}), S)$ just needs to check that $\sum_{i \in S} \vec{u}_i = \vec{t}$, which can be done in time $O(nd)$ by summing the bits of the $\vec{u}_i$ and comparing them to $\vec{t}$, which is polynomial in the length of the input.

For (b), it suffices to show that any other $\mathsf{NP}_{\mathsf{search}}$-complete problem reduces to VectorSubset-Sum. We will give a mapping reduction from 3-SAT to VectorSubsetSum.

Let $\varphi(x_0, \ldots, x_{n-1}) = C_0 \wedge C_1 \wedge \cdots \wedge C_{m-1}$ be a 3-SAT instance with n variables and m clauses. Without loss of generality, we assume that all clauses are simplified, so no variable or literal appears more than once in any clause. We will construct our VectorSubsetSum instance $R(\varphi)$ as follows. We will use the following formula with $n = 3$ and $m = 4$ as a running example:

$$\varphi(x_0, x_1, x_2) = (x_0 \vee \neg x_1 \vee x_2) \wedge (\neg x_0 \vee x_1) \wedge (x_1 \vee \neg x_2) \wedge (x_2),$$

You should use a new example when explaining the reduction during the SRE; doing so will solidify your understanding of the reduction.

Our vectors will have $d = n + m$ coordinates (so $d = 7$ in our example), and we will index the coordinates as $0, 1, \ldots, n + m - 1$. For readability, we will use different names for our vectors than $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_{t-1}$ and refer the solution S as a set of vectors rather than a set $S \subseteq [t]$ of indices.

1. Variable gadgets: for each variable x_i in φ, we will have two *variable-vectors* $\vec{u}_i^T$ and $\vec{u}_i^F$.
   (a) Including $\vec{u}_i^T$ in the solution set S will correspond to setting $x_i = 1$ and including $\vec{u}_i^F$ in the solution set S will correspond to setting $x_i = 0$.
   (b) To enforce that exactly one of $\vec{u}_i^T$ and $\vec{u}_i^F$ is included in S, we will use the i'th coordinate (entry) of the vector, setting $\vec{u}_i^T[i] = \vec{u}_i^F[i] = \vec{t}[i] = 1$, and setting $\vec{u}_j^T[i] = \vec{u}_j^F[i] = 0$ for all $j \in [n] - \{i\}$.

   In our example, our variable gadgets are as follows:

	0	1	2	3	4	5	6
$\vec{u}_0^T$	1	0	0				
$\vec{u}_0^F$	1	0	0				
$\vec{u}_1^T$	0	1	0				
$\vec{u}_1^F$	0	1	0				
$\vec{u}_2^T$	0	0	1				
$\vec{u}_2^F$	0	0	1				
$\vec{t}$	1	1	1				

2. Clause gadgets: We will use coordinates $n, \ldots, n+m-1$ to ensure that the choices of the variable-vectors satisfy all of the clauses.

   (a) For each $k \in [m]$, we will set $\vec{u}_i^T[n+k] = 1$ if variable x_i appears positively in the k'th clause C_k and set $\vec{u}_i^T[n+k] = 0$ otherwise.

   (b) For each $k \in [m]$, we will set $\vec{u}_i^F[n+k] = 1$ if variable x_i appears negated in the k'th clause and set $\vec{u}_i^F[n+k] = 0$ otherwise.

   In our example:

	0	1	2	3	4	5	6
$\vec{u}_0^T$	1	0	0	1	0	0	0
$\vec{u}_0^F$	1	0	0	0	1	0	0
$\vec{u}_1^T$	0	1	0	0	1	1	0
$\vec{u}_1^F$	0	1	0	1	0	0	0
$\vec{u}_2^T$	0	0	1	1	0	0	1
$\vec{u}_2^F$	0	0	1	0	0	1	0
$\vec{t}$	1	1	1				

   (c) With these definitions, the sum of the vectors chosen in S will be zero in coordinate $n+k$ if the k'th clause is not satisfied and will be a positive integer if the k'th clause is satisfied.

   In our example, if we sum the vectors in the set $S = \{\vec{u}_0^T, \vec{u}_1^F, \vec{u}_2^F\}$, in coordinate 3 we'll have sum 2 and in coordinate 4 we'll have sum 0, corresponding to the fact that the assignment $(1, 0, 0)$ satisfies the clause $(x_0 \vee \neg x_1 \vee x_2)$ but does not satisfy the clause $(\neg x_0 \vee x_1)$.

   (d) To enforce that the clause is satisfied, for each $k \in [m]$ we:

      i. Set $\vec{t}[n+k]$ to equal the length L_k of the clause C_k (i.e. the number of distinct literals in the clause), and

      ii. Add $L_k - 1$ *clause-padding* vectors $\vec{c}_{k,0}, \ldots, \vec{c}_{k,L_k-2}$ that have a 1 in the $(n+k)$'th coordinate and 0 in all other coordinates.

   (e) Since the number of clause-padding vectors for the k'th clause is smaller than $\vec{t}[n+k]$, the only way to satisfy the VectorSubsetSum condition in the $(n+k)$'th coordinate of $\vec{t}$ is to select at least one of the variable-vectors corresponding to a literal in the clause.

Completing our example, this the final VectorSubsetSum $R(\varphi)$ instance:

	0	1	2	3	4	5	6
$\vec{u}_0^T$	1	0	0	1	0	0	0
$\vec{u}_0^F$	1	0	0	0	1	0	0
$\vec{u}_1^T$	0	1	0	0	1	1	0
$\vec{u}_1^F$	0	1	0	1	0	0	0
$\vec{u}_2^T$	0	0	1	1	0	0	1
$\vec{u}_2^F$	0	0	1	0	0	1	0
$\vec{c}_{0,0}$	0	0	0	1	0	0	0
$\vec{c}_{0,1}$	0	0	0	1	0	0	0
$\vec{c}_{1,0}$	0	0	0	0	1	0	0
$\vec{c}_{2,0}$	0	0	0	0	0	1	0
$\vec{t}$	1	1	1	3	2	2	1

Now let's analyze the reduction (encouraging you to check your understanding against the above example):

1. R is polynomial time: $R(\varphi)$ produces $n' = 2n + (L_0 - 1) + (L_1 - 1) + \cdots + (L_{m-1} - 1) + 1$ vectors of dimension $d = n + m$. We can fill in all of the nonzero entries of these vectors in linear time by making a pass over the formula φ, observing which literals are in each clause and counting the length of each clause. Filling in the zero entries of the vectors takes time $O(n'd)$, which is also polynomial in the length of the formula φ.

2. If φ has a solution, then there is a solution to $R(\varphi)$: Suppose φ has a satisfying assignment $\alpha \in \{0,1\}^n$. Then we can take the subset S to include the following vectors:

   - $\vec{u}_i^T$ for each i such that $\alpha_i = 1$.
   - $\vec{u}_i^F$ for each i such that $\alpha_i = 0$.
   - $\vec{c}_{k,0}, \vec{c}_{k,1}, \ldots, \vec{c}_{k,L_k - s_k - 1}$ for each clause k, for s_k to be the number of literals in C_k that are satisfied by α. Note that $s_k \geq 1$ because α is a satisfying assignment to φ.

   Let's check that the vectors in S sum to our target vector t. For coordinates $i = 0, \ldots, n-1$, they sum to $\vec{t}[i] = 1$ because we include exactly one of $\vec{u}_i^T$ and $\vec{u}_i^F$. For each coordinate $n + k$ for $k = 0, \ldots, m-1$, they sum to $\vec{t}[n+k] = L_k$ because we get a contribution of s_k from the variable-vectors corresponding to literals satisfied by α in the clause and we get a contribution of $L_k - s_k$ from the clause-padding vectors we included.

3. We can transform solutions to $R(\varphi)$ to solutions to φ in polynomial time: Given a subset S of the vectors in $R(\varphi)$ that sum to t, we can construct an assignment α by setting $\alpha_i = 1$ iff $\vec{u}_i^T \in S$. The construction of α from S can be done in linear time. By the fact that the vectors sum to $\vec{t}[i] = 1$ on coordinates $i = 0, \ldots, n-1$, we also have $\alpha_i = 0$ iff $\vec{u}_i^F \in S$. Now we need to argue that α satisfies every clause C_k in φ. That is, for at least one of the literals in C_k, we include the corresponding vector $\vec{u}_i^T$ or $\vec{u}_i^F$ in the set S (and hence α makes the corresponding literal true). If that weren't the case, the sum of the vectors in S in coordinate $n + k$ would have zero contribution from the variable vectors and would therefore total at most $L_k - 1$ (the maximum contribution from the clause-padding vectors), falling short of $\vec{t}[n+k] = L_k$.

Putting it together, the following is our reduction from SAT to VectorSubsetSum:

```
1 $A(\varphi)$:
  **Input**  : A CNF formula $\varphi(x_0, \dots, x_{n-1}) = C_0 \wedge C_1 \wedge \cdots \wedge C_{m-1}$
  **Output** : A satisfying assignment $\alpha$ to $\varphi$ if one exists, else $\perp$
2 Construct the sequence of vectors $R(\varphi) = (\vec{u}_0^T, \vec{u}_0^F, \dots, \vec{u}_{n-1}^T, \vec{u}_{n-1}^F, \vec{c}_{0,0}, \dots, \vec{c}_{m-1,L_{m-1}-2})$ as
   described above;
3 Feed $R(\varphi)$ to the VectorSubsetSum oracle and receive back either $\perp$ or a set $S$;
4 **if** *the oracle returned* $\perp$ **then return** $\perp$;
5 **else**
6     Define assignment $\alpha$ by $\alpha_i = 1$ iff $\vec{u}_i^T \in S$;
7     **return** $\alpha$
```

Algorithm 1: The Reduction from SAT to VectorSubsetSum