CS120: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Active Learning Exercise 8: Reading for Receivers

Harvard SEAS - Fall 2021 *Dec. 2, 2021*

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

To prepare for this exercise as a receiver, you should try to understand the problem definitions and theorem statements in Section 4 below, and review the material on NP-completeness covered in class on November 30. Your partner sender will communicate the proof of Theorem 4.2. Section 6 contains a sketch of the proof of Theorem 4.1 for you and your receiver to think about if you finish the active learning exercise early; there is no need to prepare anything in advance for that.

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 $\sum_{i \in S} v_i = t$, or $\perp$ if no such subset exists.

Computational Problem SubsetSum

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

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

Input	: Vectors $v_0, v_1, \ldots, v_{n-1} \in \{0,1\}^d$, $t \in \mathbb{N}^d$
Output	: A subset $S \subseteq [n]$ such that $\sum_{i \in S} v_i = t$, or $\perp$ if no such subset exists.

Computational Problem VectorSubsetSum

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

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

Section 6 includes a sketch of how to derive Theorem 4.1 from Theorem 4.2, for you and your partner to discuss if you have extra time.

2 The Proof

VectorSubsetSum is in $\mathsf{NP}_{\mathsf{search}}$**:**

Reduction to show VectorSubsetSum is $\mathsf{NP}_{\mathsf{search}}$-hard, and example:

Analysis of the reduction:

3 Food for Thought

If you and your partner(s) finish early, you can think about how to reduce VectorSubsetSum to SubsetSum, and thus prove SubsetSum $\mathsf{NP}_{\mathsf{search}}$-complete. The idea is to map each d-dimensional vector v to a number by treating its entries as digits in a base b representation for a large value of b. That is, we map vector v to the number $v[0] + v[1] \cdot b + v[2] \cdot b^2 + \cdots + v[d-1] \cdot b^{d-1}$, and similarly for the target vector t. Now we claim that if the vectors $v_0, \ldots, v_{n-1}$ have $\{0, 1\}$ entries, and we take the base b to be $n+1$, then the SubsetSum instance has a valid solution S if and only if S is also a valid solution to the VectorSubsetSum instance.

Try to convince yourselves of this claim. Why do we take the base to be $n+1$? Which direction might go wrong if we just use base $b = 2$?

CS120: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Active Learning Exercise 8: Reading for Senders

Harvard SEAS - Fall 2021 *Dec. 2, 2021*

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 4 and 6 are also in the reading for receivers. Your goal will be to communicate the *proof* of Theorem 4.2 (i.e. the content of Section 5) to the receivers. Section 6 contains a sketch of the proof of Theorem 4.1 for you and your receiver to think about if you finish the active learning exercise early; there is no need to prepare anything in advance for that.

4 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 $\sum_{i \in S} v_i = t$, or $\perp$ if no such subset exists.

Computational Problem SubsetSum

Theorem 4.1. *SubsetSum is* $\mathsf{NP}_{\mathsf{search}}$*-complete.*

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

Input	: Vectors $v_0, v_1, \ldots, v_{n-1} \in \{0,1\}^d$, $t \in \mathbb{N}^d$
Output	: A subset $S \subseteq [n]$ such that $\sum_{i \in S} v_i = t$, or $\perp$ if no such subset exists.

Computational Problem VectorSubsetSum

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

Theorem 4.2. *VectorSubsetSum is* $\mathsf{NP}_{\mathsf{search}}$*-complete.*

Section 6 includes a sketch of how to derive Theorem 4.1 from Theorem 4.2, for you and your partner to discuss if you have extra time.

5 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 observe that solutions (the set S) can be described in n bits, which is shorter than the length of the input $(v_0, v_1, \ldots, v_{n-1}, t)$, and thus certainly polynomially bounded. A verifier for

solutions $V((v_0, v_1, \ldots, v_{n-1}, t), S)$ just needs to check that $\sum_{i \in S} v_i = t$, which can be done in time $O(nd)$, 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.

Let $\varphi(x_0, \ldots, x_{n-1}) = C_0 \wedge C_1 \wedge \cdots \wedge C_{m-1}$ be a 3-SAT instance. We will construct our VectorSubsetSum instance $R(\varphi)$ as follows.

1. Variable gadgets: for each variable x_i in φ, we will have two *variable-vectors* v_i^T and v_i^F.
   (a) Including v_i^T in the solution set S will correspond to setting $x_i = 1$ and including v_i^F in the solution set S will correspond to setting $x_i = 0$.
   (b) To enforce that exactly one of v_i^T and v_i^F is included in S, we will use the i'th coordinate of the vector, setting $v_i^T[i] = v_i^F[i] = t[i] = 1$, and setting $v_j^T[i] = v_j^F[i] = 0$ for all $j \neq i$.
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 $v_i^T[n + k] = 1$ if variable x_i appears positively in the k'th clause and set $v_i^T[n + k] = 0$ otherwise.
   (b) For each $k \in [m]$, we will set $v_i^F[n + k] = 1$ if variable x_i appears negated in the k'th clause and set $v_i^F[n + k] = 0$ otherwise.
   (c) With these definitions, the sum of the vectors chosen in S will be zero if the clause is not satisfied and will be a positive integer if the clause is satisfied.
   (d) To enforce that the clause is satisfied, we set $t[n+k]$ to equal the length L_k of the clause C_k (i.e. the number of distinct literals in the clause), and add $L_k - 1$ *clause-padding* vectors $c_{k,0}, \ldots, c_{k,L_k-2}$ that have a 1 in the $(n + k)$'th coordinate (and zero in all other coordinates).
   (e) Since the number of clause-padding vectors for the k'th clause is smaller than $t[n + k]$, the only way to satisfy the VectorSubsetSum condition in the $(n + k)$'th coordinate is to select at least one of the variable-vectors corresponding to a literal in the clause.

To understand what this reduction is doing, we strongly suggest working out a small example, say with 3 variables and 4 clauses of lengths $3, 2, 2, 1$, and writing down all the vectors in a table like the following:

	0	1	2	3	4	5	6
t							
v_0^T							
v_0^F							
v_1^T							
v_1^F							
v_2^T							
v_2^F							
$c_{0,0}$							
$c_{0,1}$							
$c_{1,0}$							
$c_{2,0}$							

Example. Given the following 3SAT formula:

$$\phi(x_0, x_1, x_2) = (x_0 \vee \neg x_1 \vee x_2) \wedge (\neg x_0 \vee x_1) \vee (x_1 \vee \neg x_2) \wedge (x_3),$$

we build the table as follows:

	0	1	2	3	4	5	6
v_0^T	1	0	0	1	0	0	0
v_0^F	1	0	0	0	1	0	0
v_1^T	0	1	0	0	1	1	0
v_1^F	0	1	0	1	0	0	0
v_2^T	0	0	1	1	0	0	1
v_2^F	0	0	1	0	0	1	0
$c_{0,0}$	0	0	0	1	0	0	0
$c_{0,1}$	0	0	0	1	0	0	0
$c_{1,0}$	0	0	0	0	1	0	0
$c_{2,0}$	0	0	0	0	0	1	0
t	1	1	1	3	2	2	1

Now let's analyze the reduction:

1. R is polynomial time: $R(\varphi)$ produces $n' = 2n + L_0 + L_1 + \cdots + L_{m-1} + 1$ 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 non-$\perp$ solution, there is a non-bot 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:

   - v_i^T for each i such that $\alpha_i = 1$.
   - v_i^F for each i such that $\alpha_i = 0$.
   - $c_{k,0}, c_{k,1}, \ldots, c_{k,L_k - s_k - 1}$ 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 $t[i] = 1$ because we include exactly one of v_i^T and v_i^F. For each coordinate $n + k$ for $k = 0, \ldots, m-1$, they sum to $t[n+k] = L_k$, 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 non-$\perp$ solutions to $R(\varphi)$ to non-$\perp$ 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 $v_i^T \in S$. The construction of α from S can be done in linear time. By the fact that the vectors sum to $t[i] = 1$ on coordinates $i = 0, \ldots, n-1$, we also have $\alpha_i = 0$ iff $v_i^F \in S$. 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 v_i^T or v_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 thus would total at most $L_k - 1$ (the maximum contribution from the clause-padding vectors), falling short of $t[n+k] = L_k$.

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

$A(\varphi)$:
Input : A CNF formula $\varphi(x_0, \ldots, x_{n-1}) = C_0 \wedge C_1 \wedge \cdots \wedge C_{m-1}$
Output : A satisfying assignment α to φ if one exists, else $\perp$
Construct the sequence of vectors $R(\varphi) = (v_0^T, v_0^F, \ldots, v_{n-1}^T, v_{n-1}^F, c_{0,0}, \ldots, c_{m-1,L_{m-1}-2})$ as described above;
Feed $R(\varphi)$ to the VectorSubsetSum oracle and receive back S;
if $S = \perp$ **then return** $\perp$;
else
    Define assignment α by $\alpha_i = 1$ iff $v_i^T \in S$;
    return α

Algorithm 1: The Reduction from SAT to VectorSubsetSum

6 Food for Thought

If you and your partner(s) finish early, you can think about how to reduce VectorSubsetSum to SubsetSum, and thus prove SubsetSum $\mathsf{NP}_{\mathsf{search}}$-complete. The idea is to map each d-dimensional vector v to a number by treating its entries as digits in a base b representation for a large value of b. That is, we map vector v to the number $v[0] + v[1] \cdot b + v[2] \cdot b^2 + \cdots + v[d-1] \cdot b^{d-1}$, and similarly for the target vector t. Now we claim that if the vectors $v_0, \ldots, v_{n-1}$ have $\{0, 1\}$ entries, and we take the base b to be $n+1$, then the SubsetSum instance has a valid solution S if and only if S is also a valid solution to the VectorSubsetSum instance.

Try to convince yourselves of this claim. Why do we take the base to be $n+1$? Which direction might go wrong if we just use base $b = 2$?