

SubsetSum is NP-complete and some related questions

This document discusses the **SubsetSum** decision problem, shows that it is NP-complete by showing that $3SAT \leq_p \text{SubsetSum}$

1 Problem Definitions

We define SubsetSum and 3SAT.

SubsetSum

Input: $w_1, \dots, w_n, t \in \mathbb{N}$

Question: Is there a set $S \subseteq \{1, \dots, n\}$ such that $\sum_{i \in S} w_i = t$?

3SAT

Input: A set of clauses $C = \{c_1, \dots, c_m\}$ over literals $x_1, \dots, x_n, \overline{x_1}, \dots, \overline{x_n}$ such that each c_i has size 3. C represents a propositional formulae in CNF (Conjunctive Normal Form).

Question: Is there some truth assignment τ such that $C^\tau = \top$? That is, is there a truth assignment τ such that

$$\bigwedge_{i=1}^m (\bigvee_{l \in c_i} l)^\tau = \top$$

Exercises:

1. Come up with an instance I_1 of the SubsetSum problem such that $I_1 \notin \text{SubsetSum}$.
2. Come up with an instance I_2 of the SubsetSum problem such that $I_2 \in \text{SubsetSum}$.

2 Discussion

The description of the encoding used for inputs to the SubsetSum problem is in the Encoding section below.

In general we are interested in the complexity of solving problems. To get a feel for this problem, consider the following algorithm which finds a set S such that $\sum_{i \in S} w_i = t$. We assume that weights w_1, \dots, w_n appear in the array $W[]$.

```
SubsetSum(W[],t){
  for each S (subset of {1,...,n}) {
    total=0
    for(i in S){
      total=total+W[i]
    }
    if(total==t)return(S)
  }
  return(reject);
}
```

This simple algorithm actually has terrible running time. The above algorithm has running time worse than 2^n . If we had 500 weights, the above algorithm would take a LONG time to complete!

3 Proofs

Theorem: SubsetSum is NP-complete

Proof: Follows from the 2 theorems below.

Theorem: SubsetSum \in NP

Proof:

SubsetSum \in NP since

$$\text{SubsetSum} = \{x \in \{0, 1, \text{comma}\}^* : \exists y \in \{0, 1\}^* (|y| \leq c|x|^d \text{ and } R(x, y))\}$$

where

1. **Describe R :** R accepts the pair (x, y) if x codes a set of weights and target w_1, \dots, w_n, t and y codes a subset S of $\{1, \dots, n\}$ and $\sum_{i \in S} w_i = t$. Otherwise R rejects.

2. **Show that R runs in polytime:** R operates by taking inputs x (encoding w_1, \dots, w_n, t) and y (encoding some $S \subseteq \{1, \dots, n\}$) and

(a) checks that $\sum_{i \in S} w_i = t$

R accepts if this is the case and rejects otherwise. From the description above, it should be clear that R can operate in polytime.

3. **Argue that $|y|$ is polynomial in $|x|$:** Finally, from the description of our encoding of subsets of $\{1, \dots, n\}$ (see below), we can take $c = 1$ and $d = 1$. In this case, $|y| \leq |x|$.

Theorem: SubsetSum is NP-hard

Proof: To show that SubsetSum is NP-hard (by a Theorem we proved in class) it is enough to show that $3\text{SAT} \leq_p \text{SubsetSum}$. Note that we are assuming that 3SAT is NP-complete. By definition, to show that $3\text{SAT} \leq_p \text{SubsetSum}$ we need to find an $f \in FP$ (polytime computable function) such that

$$(\forall x)(x \in 3\text{SAT} \Leftrightarrow f(x) \in \text{SubsetSum})$$

Describe f : We describe f by saying what it does to an instance $C = \{c_1, \dots, c_m\}$ of 3SAT.

$f(C) = w_1, \dots, w_s, t$ we think of each w_i as the string of bits representing w_i in binary. Logically, each w_i consists of a set of bits broken down into the following sections $(fw)(ts)(c_1)(c_2) \dots (c_m)$. The (fw) section is used to make sure that exactly one of the filler weights (defined below) is chosen for each clause. Similarly, the (ts) section of bits is used to make sure that exactly one of the truth assignment weights is chosen for each variable. Each of the (c_i) sections of bits corresponds to the i th clause c_i of the 3SAT problem.

$(fw) = (fw_1fw_2 \dots fw_m)$ consists of m sections (one section for each clause). Each fw_i consists of 3 bits. $(ts) = (x_1x_2 \dots x_n)$ consists of n bits, each bit corresponds to 1 variable. Finally, each of the (c_m) sections has 3 bits, 1 bit for each literal appearing in clause c_i . So in total, a single weight will have $3m + n + 3m$ bits.

$f(C)$ consists of two types of weights, the truth setting weights $TSW = \{w_{x_1}, w_{\overline{x_1}}, \dots, w_{x_n}, w_{\overline{x_n}}\}$ and the filler weights FW . We will discuss the filler weights in a second.

w_{x_i} is the bit string with a 1 in the (ts) section corresponding to x_i and a 1 in bit k of section (c_j) if the k th bit of (c_j) corresponds to x_i . All other bits of w_{x_i} are 0. Similarly, $w_{\overline{x_i}}$ is the bit

string with a 1 in the (ts) section corresponding to x_i and a 1 in bit k of section (c_j) if the k th bit of (c_j) corresponds to $\overline{x_i}$. All other bits of w_{x_i} are 0.

For example, if our original clause set was

$$\{\{x_1, x_2, \overline{x_4}\}, \{\overline{x_1}, x_3, x_4\}\}$$

then each weight produced by f would consist of $(fw_1fw_2)(x_1x_2x_3x_4)(c_1)(c_2)$. Each fw_i consists of 3 bits, each x_i consists of 1 bit, each c_i consists of 3 bits. We assume the association between bits in (c_1) and literals in $c_1 = \{x_1, x_2, \overline{x_4}\}$ is from left to right so that the high order bit in (c_1) corresponds to x_1 , the middle bit in (c_1) corresponds to x_2 and the low order bit of c_1 corresponds to $\overline{x_4}$.

The truth setting weights are $TSW =$

$$\begin{aligned} \{w_{x_1} &= (000000)(1000)(100)(000), \\ w_{\overline{x_1}} &= (000000)(1000)(000)(100), \\ w_{x_2} &= (000000)(0100)(010)(000), \\ w_{\overline{x_2}} &= (000000)(0100)(000)(000), \\ w_{x_3} &= (000000)(0010)(000)(010), \\ w_{\overline{x_3}} &= (000000)(0010)(000)(000), \\ w_{x_4} &= (000000)(0001)(000)(001), \\ w_{\overline{x_4}} &= (000000)(0001)(001)(000)\} \end{aligned}$$

Notice that no matter which subset of TSW we choose, we can never get a carry out of a clause section. Also, if we are interested in achieving a target with an all 1 (ts) section then we must choose exactly 1 of w_{x_i} or $w_{\overline{x_i}}$.

Each clause has a set of 7 filler weights. The 7 filler weights for c_i each have a 1 as the low order bit of (fw_i) and all non 111 bit strings in the (c_i) section. So corresponding to c_1 we have the 7 filler weights

$$\begin{aligned} \{fw_1^0 &= (001000)(0000)(000)(000), \\ fw_1^1 &= (001000)(0000)(001)(000), \\ fw_1^2 &= (001000)(0000)(010)(000), \\ fw_1^3 &= (001000)(0000)(011)(000), \\ fw_1^4 &= (001000)(0000)(100)(000), \\ fw_1^5 &= (001000)(0000)(101)(000), \\ fw_1^6 &= (001000)(0000)(110)(000)\} \end{aligned}$$

Similarly, for c_2 we have the 7 filler weights

$$\begin{aligned} \{fw_2^0 &= (000001)(0000)(000)(000), \\ fw_2^1 &= (000001)(0000)(000)(001), \\ fw_2^2 &= (000001)(0000)(000)(010), \\ fw_2^3 &= (000001)(0000)(000)(011), \\ fw_2^4 &= (000001)(0000)(000)(100), \\ fw_2^5 &= (000001)(0000)(000)(101), \\ fw_2^6 &= (000001)(0000)(000)(110)\} \end{aligned}$$

Notice that if S is a subset of the filler weights for (c_i) then $\sum_{w \in S} w$ has no carries out of the fw_i section of bits. So if we are interested in a target of the form (001001001) ... then exactly 1

filler weight for each clause section can be chosen.

The target weight is $t = (001001)(1111)(111)(111)$. In general, the target weight consists of a 1 in the low order bit of each fw_i section as well as an all 1 (ts) section and all 1 clause section.

So there is a total of $2n$ truth setting weights and $7m$ filler weights and 1 target weight.

Argue that f is computable in polytime: It should be clear that f is computable in polytime. f is not doing anything complicated.

Show that f really works: Finally, we show that $x \in 3SAT \Leftrightarrow f(x) \in SubsetSum$

As above, we interpret x as $C = \{c_1, \dots, c_m\}$ and $f(x) = f(C) = (G = (V, E), m)$.

$C \in 3SAT \Rightarrow \exists \tau$ a truth value assignment, such that $C^\tau = \top$ That is, $c_j^\tau = \top$ for $j = 1, \dots, m$. Consider the following initial set of weights (we will consider the weights themselves instead of the indices since all weights appearing in $f(C)$ are distinct). Initially $S = \{w_{x_i} : x_i^\tau = T\} \cup \{w_{\bar{x}_i} : \bar{x}_i^\tau = T\}$ then $\sum_{w \in S} w$ is a bit string with all 1s in the (ts) section and a 1 in at least one bit of each clause section (c_i) (since τ satisfies C). This almost matches the target bit string. To complete the set S , we add in one of the fillers for (c_1) so that the (c_1) section of the total sum consists of 111. We do the same for each of the other clause sections and end up in total with n members of TSW (one for each variable) and m members of FW (one for each clause). It should be clear that the resulting set S has

$$\sum_{w \in S} w = t$$

so $f(C) \in SubsetSum$. So $C \in 3SAT \Rightarrow f(C) \in SubsetSum$.

We make the above argument more concrete using the above example. Consider the truth assignment τ which sets $x_1^\tau = F, x_2^\tau = F, x_3^\tau = T, x_4^\tau = F$. Then $C^\tau = T$. So $C \in 3SAT$. Above we constructed the weights and target corresponding to $f(C)$. Consider

$$S = \{w_{\bar{x}_1}, w_{\bar{x}_2}, w_{x_3}, w_{\bar{x}_4}\}$$

Then

$$\sum_{w \in S} w = (000000)(1111)(001)(110)$$

We now add in the following fillers, for (c_1) we add in fw_1^4 and for (c_2) we add in fw_2^1 . In total,

$$S = \{w_{\bar{x}_1}, w_{\bar{x}_2}, w_{x_3}, w_{\bar{x}_4}, fw_1^4, fw_2^1\}$$

and

$$\sum_{w \in S} w = (001001)(1111)(111)(111) = t$$

so $f(C) \in SubsetSum$.

If $w_{x_1}, w_{\bar{x}_1}, \dots, w_{x_n}, w_{\bar{x}_n}, fw_1^0, \dots, fw_1^6, \dots, fw_m^0, \dots, fw_m^6 \in SubsetSum$ then there is a collection of weights S such that $\sum_{w \in S} w = t$, but by the comments made during the description of f , S contains exactly one of w_{x_i} or $w_{\bar{x}_i}$ for each $i \in \{1, \dots, n\}$. Similarly, S contains exactly one of

$\{fw_j^0, \dots, fw_j^6\}$ for each $j \in \{1, \dots, m\}$. We define a truth assignment τ by $x_i^\tau = T \Leftrightarrow w_{x_i} \in S$.

Exercise: Complete the remaining part of this proof: Show that $C^\tau = T$ so that $C \in 3SAT$.

So $f(C) \in SubsetSum \Rightarrow C \in 3SAT$.

Combining the two results, we have $x \in 3SAT \Leftrightarrow f(x) \in SubsetSum$ so that $SubsetSum$ is NP-hard.

Exercise:

1. Write up a clause set $C \in 3SAT$, write down $f(C)$. Find a truth value assignment satisfying C and (using the proof above) find the corresponding index set S such that $\sum_{i \in S} w_i = t$. Conclude that $f(C) \in SubsetSum$.
2. Write up a clause set C such that $f(C) \in SubsetSum$. Find any index set S such that $\sum_{i \in S} w_i = t$ and the associated truth value assignment τ (defined in the proof above) such that $C^\tau = \top$. Conclude that $C \in 3SAT$.

4 Encodings

We assume that w_1, \dots, w_n, t is encoded as a sequence of comma separated binary values. That is an $(n+1) \log_2(M)$ bit string of 0 and 1 and *commas* listing the weights in order and finally the target weight t . M is the maximum weight or target listed. For example 1101, 11, 10, 111 represents the instance with $w_1 = 13, w_2 = 3, w_3 = 2, t = 7$.

We will require an encoding for subsets of $\{1, \dots, n\}$. We will encode $S \subseteq \{1, \dots, n\}$ as an n -bit bit string such that the i th bit is 1 if and only if the $i \in S$. If x encodes some instance w_1, \dots, w_n, t of $SubsetSum$ and y encodes $S \subseteq \{1, \dots, n\}$ then $|y| \leq |x|$.

Exercises:

1. Explain why $|y| \leq |x|$ above.