subset sum

- **Construction**: Given 3-SAT instance \( \Phi \) with \( n \) variables and \( k \) clauses, form 
  
  \[ 2n + 2k \text{ decimal integers, each of } n+k \text{ digits, as illustrated below.} \]

- **Claim**: \( \Phi \) is satisfiable iff there exists a subset that sums to \( W \).

- **Proof**: No carries possible.

```
C_1 = \bar{x} \vee y \vee z
C_2 = x \vee y \vee z
C_3 = \bar{x} \vee \bar{y} \vee \bar{z}

\[
\begin{array}{cccccc}
1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
```

- \( \Phi \text{ is satisfiable iff } \exists \text{ subset } W \text{ such that } \sum W = W \text{ in binary encoding.} \)

- **Claim**: \( 3\text{-SAT} \leq_p \text{SUBSET-SUM} \).

- **Pf**: Given an instance \( \Phi \) of 3-SAT, we construct an instance of SUBSET-SUM that has solution iff \( \Phi \) is satisfiable.

**xkcd**
scheduling with release times

- **SCHEDULE-RELEASE-TIMES.** Given a set of n jobs with processing time \( t_i \), release time \( r_i \), and deadline \( d_i \), is it possible to schedule all jobs on a single machine such that job \( i \) is processed with a contiguous slot of \( t_i \) time units in the interval \( [r_i, d_i] \)?

- **Claim: SUBSET-SUM ≤\_p SCHEDULE-RELEASE-TIMES.**
  - **Pf.** Given an instance of SUBSET-SUM \( w_1, \ldots, w_n \) and target \( W \),
    - Create \( n \) jobs with processing time \( t_i = w_i \), release time \( r_i = 0 \), and no deadline \( (d_i = 1 + \sum w_j) \).
    - Create job 0 with \( t_0 = 1 \), release time \( r_0 = W \), and deadline \( d_0 = W+1 \).

  Can schedule jobs 1 to n anywhere but \( [W, W+1] \)

### NP-completeness

**Candy Crush is NP-hard**

**Title:** Candy Crush is NP-hard

**Authors:**

Sven Žnidarsic, Varnika Gupta, J.J. Dongarra

**Abstract:**

We prove that playing Candy Crush is at least as difficult as the hardest problems in the class NP. Our proof is based on a reduction from 3-SAT, a well-known NP-complete problem.

**Introduction:**

Candy Crush is a popular casual game that combines elements of puzzle-solving and strategy. In the game, players must match candies of the same color to make them disappear, earning points and clearing levels. This game has gained immense popularity due to its accessibility and addictive gameplay.

Candy Crush is NP-hard, which means it is at least as hard as the hardest problems in the class NP. This classification is significant because it implies that Candy Crush is at least as difficult as the hardest problems in the class NP, and that there is no known algorithm that can solve it in polynomial time.

**Coping with NP-completeness**

**Approximation.**

Euclidean TSP
coping with NP-completeness

Approximation.

Euclidean TSP

coping with NP-completeness

Average case inputs: E.g. random 3-SAT

\[ \Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_2 \lor x_4 \lor \neg x_5) \land (\neg x_3 \lor x_6 \lor \neg x_9) \land \ldots \]

new algorithmic frontiers