1 What is the Subset Sum Problem?

An instance of the *Subset Sum problem* is a pair (S, t), where $S = \{x_1, x_2, ..., x_n\}$ is a set of positive integers and t (the target) is a positive integer. The decision problem asks for a *subset* of S whose *sum* is as large as possible, but not larger than t.

This problem is NP-complete.

This problem arises in practical applications. Similar to the knapsack problem we may have a truck that can carry at most t pounds and we have n different boxes to ship and the i^{th} box weighs x_i pounds.

The naive approach of computing the sum of the elements of every subset of S and then selecting the best requires exponential time. Below we present an exponential time exact algorithm.

2 An Exact Algorithm for the Subset-Sum Problem

In iteration i, we compute the sums of all subsets of $\{x_1, x_2, ..., x_i\}$, using as a starting point the sums of all subsets of $\{x_1, x_2, ..., x_{i-1}\}$. Once we find the sum of a subset S' is greater than t, we ignore that sum, as there is no reason to maintain it. No superset of S' can possibly be the optimal solution.

Notation: If L is a list of positive integers and x is another positive integer, then we let $L + x$ denote the list of integers derived from L by increasing each element of L by x. For example, if $L = \langle 1, 2, 4\rangle$ then $L + 2 = \langle 3, 4, 6\rangle$

$MergeLists(L, L')$ returns the sorted list that is the merge of two sorted input lists L and L'

Algorithm 1: EXACT-SUBSET-SUM(S, t)

$n \longleftarrow |S|$
$L_0 \longleftarrow \langle 0\rangle$
for i = 1 to n **do**
    $L_i \longleftarrow MergeLists(L_{i-1}, L_{i-1} + x_i)$
    remove from L_i every element greater than t
return the largest element in L_n

Analysis:
It can be shown by induction that above algorithm is correct. EXACT-SUBSET-SUM(S, t) is an exponential time algorithm in general since the length of L_i can be as high as 2^i.
Example: Let $S = \{1, 4, 5\}$ then.......
$L_0 = \{0\}$
$L_1 = \{0, 1\}$
$L_2 = \{0, 1, 4, 5\}$
$L_3 = \{0, 1, 4, 5, 6, 9, 10\}$ So if the target was 9 we would have removed 10 from the last list.

3 FPTAS for the Subset-Sum Problem

3.1 What is a PTAS / FPTAS?

A polynomial time approximation scheme (PTAS) is an algorithm that takes as input not only an instance of the problem but also a value $\epsilon > 0$ and approximates the optimal solution to within a ratio bound of $1 + \epsilon$. For any choice of ϵ the algorithm has a running time that is polynomial in n, the size of the input.
Example: a PTAS may have a running time bound of $O(n^{2/\epsilon})$

A fully polynomial-time approximation scheme (FPTAS) is a PTAS with a running time that is polynomial not only in n but also in $1/\epsilon$.
Example: a PTAS with a running time bound of $O((1/\epsilon)^2 n^3)$ is an FPTAS

3.2 Trim Subroutine for Approximate Subset Sum Algorithm

Before we explain the approximation algorithm, we explain how to trim our list L_i using the parameter δ. If several values in L are close to each other, maintain only one of them,i.e we trim each list L_i after it is created.

Given a parameter δ, where $0 < \delta < 1$, element z approximates element y if $y/(1+\delta) \leq z \leq y$. To trim a list L_i by δ, remove as many elements as possible such that every element that is removed is approximated by some remaining element in the list.

Example:

If $\delta = 0.1$ and
$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$
will be trimmed to
$L = \langle 10, 12, 15, 20, 23, 29 \rangle$
since 11 approximates 12, 20 approximates 21 and 22 and similarly 23 approximates 24.

Algorithm 2: TRIM(L, δ)

$m \longleftarrow |L|$
$L' \longleftarrow \langle 0 \rangle$
$last \longleftarrow y_1$
for i = 2 to n **do**
  if $y_i > last.(1+\delta)$ **then**
    append y_i onto the end of L'
    $last \longleftarrow y_i$
return L'

The running time for above algorithm is $\Theta(m)$. We now present the approximate subset sum algorithm that uses Trim and MergeLists.

3.3 Approximate Subset Sum Algorithm

Algorithm 3: APPROX-SUBSET-SUM(S, t, ϵ)

$n \longleftarrow |S|$
$L_0 \longleftarrow \langle 0 \rangle$
for i = 1 to n **do**
  $L_i \longleftarrow MergeLists(L_{i-1}, L_{i-1} + x_i)$
  $L_i \longleftarrow Trim(L_i, \epsilon/2n)$
  remove from L_i every element greater than t;
return the largest element in L_n

3.4 Analysis of Approximate Subset Sum Algorithm

Theorem: For $0 < \epsilon < 1$, APPROX-SUBSET-SUM(S, t, ϵ) is a FPTAS for the subset sum problem.

Proof:
We need to show that

1. The solution returned is within a factor of $1 + \epsilon$ of the optimal solution.
2. The running time is polynomial in both n and $1/\epsilon$

Let z^* be the value returned by APPROX-SUBSET-SUM(S, t, ϵ) and let y^* be an optimal solution.

As $z^* \leq y^*$, we need to show that $\frac{y^*}{z^*} \leq 1 + \epsilon$

For every element $y \leq t$ that is the sum of a subset of the first i numbers in S, there is a $z \in L_i$, such that

$$\frac{y}{(1 + \epsilon/2n)^i} \leq z \leq y$$

Taking $i = n$ and using the fact that z^* is the largest element in L_n,

$$\frac{y^*}{z^*} \leq (1 + \epsilon/2n)^n \leq e^{\epsilon/2} \leq 1 + \epsilon$$

Therefore,

$$\frac{y^*}{z^*} \leq 1 + \epsilon \tag{1}$$

Now lets look at the number of elements in L_i:
After trimming, successive elements z and z' in L_i must differ by at least $1 + \epsilon/2n$
The number of elements in L_i is at most

$$2 + \log_{1+\epsilon/2n} t = 2 + \frac{\ln t}{\ln(1 + \epsilon/2n)}$$

$$\leq 2 + \frac{4n \ln t}{\epsilon} \tag{2}$$

This implies that this above value is polynomial in size of the input(i.e $\log t$ plus some polynomial in n) and in $1/\epsilon$.

Therefore since the running time of Approx-Subset-Sum is polynomial in size of the length of L_i, from (1) and (2) we have shown that Approx-Subset-Sum is an FPTAS.