### Approximation Algorithms Recap

An algorithm $A$ is an $\alpha$-approximation for problem $P$ if,

- $A$ runs in polynomial time
- $A$ always outputs a solution with value $s$ within an $\alpha$ factor of $Opt$

- Here $P$ is an optimisation problem with optimal solution of value $Opt$

- If $P$ is a maximisation problem, $\frac{Opt}{\alpha} \leq s \leq Opt$
- If $P$ is a minimisation problem, $Opt \leq s \leq \alpha \cdot Opt$

We have seen:

- A 2-approximation for Max Sat
- A $\frac{1}{2}$-approximation for Bin Packing
- A 2-approximation for $k$-centers
- A $\frac{3}{2}$-approximation for scheduling multiple machines

---

### The Subset Sum problem

Let $S$ be a multi-set of positive integers and $t$ be a positive integer.

**Decision Problem** Is there a subset, $S' \subseteq S$ with size $t$?

**Optimisation Problem** Find the size of the largest subset of $S$ which is no larger than $t$

The optimisation version is NP-hard and the decision version is NP-complete.

---

### An exact solution

Let $S = \{s_1, s_2, s_3, \ldots, s_m\}$ be the set of items and $S_i = \{s_1, s_2, \ldots, s_i\}$

Let $L_i$ be the set of sizes of all $S' \subseteq S_i$ which are not larger than $t$

- $S = \{2, 4, 4, 7, 2, 3\}$
- $t = 12$

- $S_4 = \{2, 4, 4, 7, 2, 3\}$
- $L_4 = \{0, 2, 4, 6, 7, 8, 9, 11\}$

- The largest subset of $S$ (of size at most $t$) is the largest number in $L_m$
- We compute $L_i$ from $L_{i-1}$:
  - $L_i = L_{i-1} \cup (L_{i-1} + s_i)$
  - $L_5 = \{0, 2, 4, 6, 7, 8, 9, 10, 11\}$

---

### An exact solution

The algorithm:

- For $i = 1 \ldots m$:
  - Compute ($L_{i-1} + s_i$) from $L_{i-1}$
  - Compute $L_i = L_{i-1} \cup (L_{i-1} + s_i)$
  - Output the largest number in $L_m$

- $L_0 = \{0\}$
- $|L_{i-1}|$ time
- $O(|L_i|)$ time
- $O(|L_m|)$ time

- $n$ is the length of the input (measured in words)
- $a$ is bit word
- $n$ words
- $\Theta(\log n)$

---

The overall time complexity is therefore $O(mt)$.
**An exact solution**

\[ |S| = m \]

The algorithm

- Let \( L_0 = \{0\} \)
- For \( i = 1 \) \( \ldots \) \( m \):
  - Compute \( L_{i-1} + s_i \) from \( L_{i-1} \)
  - Compute \( L_i = L_{i-1} \cup (L_{i-1} + s_i) \)
- Output the largest number in \( L_m \)

**Pseudo-polynomial time algorithms**

- We say that an algorithm is pseudo-polynomial time if it runs in poly-time when all the numbers are integers \( \leq n^c \) for some constant \( c \).
- The algorithm for Subset Sum gives \( O(nt) \in O(n^{c+1}) \) time in this case.
- We say that an NP-complete problem is weakly NP-complete if there is a pseudo-polynomial time solution for it.
- We say that an NP-complete problem is strongly NP-complete if it is NP-complete when all the numbers are integers \( \leq n^c \).
- So Subset Sum is weakly NP-complete.
- Bin packing is strongly NP-complete.

**Polynomial time approximation schemes**

A polynomial time approximation scheme (PTAS) for problem \( P \)

- A PTAS is a family of algorithms:
  - For any constant \( \epsilon > 0 \) there is an algorithm in the family, \( A_\epsilon \)
  - such that \( A_\epsilon \) is a \((1+\epsilon)\)-approximation for \( P \)
- If we had a PTAS for Subset Sum,
  - Let \( \epsilon = 0.1 \) and so \( A_{0.1} \) runs in polynomial time and
    - outputs a subset of size at least \( \text{Out} \geq 0.9 \cdot \text{Opt} \)
  - Let \( \epsilon = 0.01 \) and so \( A_{0.01} \) also runs in polynomial time and
    - outputs a subset of size at least \( \text{Out} \geq 0.99 \cdot \text{Opt} \)
  - Let \( \epsilon = 0.001 \) and so \( A_{0.001} \) also runs in polynomial time and
    - outputs a subset of size at least \( \text{Out} \geq 0.999 \cdot \text{Opt} \)

A polynomial time approximation scheme (PTAS) for problem \( P \)

- A PTAS for problem \( P \) is a family of algorithms:
  - For any constant \( \epsilon > 0 \) there is an algorithm in the family, \( A_\epsilon \)
  - such that \( A_\epsilon \) is a \((1+\epsilon)\)-approximation for \( P \)
- If we had a PTAS for Subset Sum,
  - Let \( \epsilon = 0.1 \) and so \( A_{0.1} \) runs in polynomial time and
    - outputs a subset of size at least \( \text{Out} \geq 0.9 \cdot \text{Opt} \)
  - A PTAS does not have to have running time polynomial in \( 1/\epsilon \)
  - A fully PTAS (FPTAS) is also polynomial in \( 1/\epsilon \)
    - i.e. the time complexity is \( O((\epsilon/n)^c) \) for some constant \( c \)
  - In our example \( O((100m)^c) = O((100n)^c) = O((10000n)^c) = O(n^c) \)

A PTAS for Subset Sum

Let \( L_i \) be the sets of sizes of all \( S' \subseteq S \), which are not larger than \( i \)

- The exact algorithm for Subset Sum was slow (in general) because
  each list of possible subset sizes \( L_i \) could become very large

**Key Idea**

- Constructed a **trimmed** version of \( L_i \) (denoted \( L'_i \)) so that
  - The length of \( L'_i \) is polynomial in the input length
  - \( L'_i \subseteq L_i \) and for every \( y \in L_i \), there is a \( z \in L'_i \) which
    is almost as big
  - \( L'_i = \{0, 2, 4, 6, 7, 8, 9, 10, 11\} \)

**Trim(\( L_i, \delta \))**

- Include \( L_i[j] \) in \( L'_i \) if \( L_i[j] > (1 + \delta) \cdot \text{prev} \)
  - where \( \text{prev} \) is the previous thing we included in \( L'_i \)
  - for \( \delta = 1 \) we include \( L'_2 = \{0, 2, 6\} \)
  - we will pick \( \delta \) later (and it will be much smaller than 1)
## S = \{4, 2, 4, 7, 2, 3\} \quad t = 11

A PTAS for Subset Sum

\[ S_4 = (4, 2, 4) \]

Let \( L_t \) be the set of sizes of all \( S' \subseteq S_t \) which are not larger than \( t \)

- \( L_t' \) is the trimmed version of \( L_t \)

### Algorithm

- Let \( L_0' = \{0\}, \delta = \epsilon/(2m) \)
- For \( i = 0 \ldots m: \)
  - Compute \( (L_{i-1}') + s_i \) from \( L_{i-1}' \)
  - Compute \( L_i = L_{i-1}' \cup (L_{i-1}' + s_i) \)
  - Let \( L_i' = \text{Trim}(L_i, \delta) \)
  - Output the largest number in \( L_i' \)

- This process throws away some subsets, but it outputs a valid solution

- Two questions remain... How big is \( |L_t'| \)? How good is the solution given?

### Lemma

For any \( y \in L_t \), there is an \( z \in L_t' \) with \[
\frac{y}{(1+\delta)^t} \leq z \leq y
\]

- By setting \( i = m \) and \( \delta = \epsilon/2m \) we have that,
  - For any \( y \in L_m \), there is a \( z \in L_m' \) with \[
  \frac{y}{(1+\delta)^m} \leq z \leq y
\]
  - Further, \( \text{Opt} \in L_m \), meaning there is a \( z \in L_m' \) with
    \[
    \frac{\text{Opt}}{(1+\delta)^m} \leq z \leq \text{Opt}
    \]

We only need to show that \( (1 + \frac{\epsilon}{2m})^m \leq 1 + \epsilon \ldots \)

### Lemma

For any \( y \in L_i \), there is an \( z \in L_i' \) with \[
\frac{y}{(1+\delta)^i} \leq z \leq y
\]

- By setting \( i = m \) and \( \delta = \epsilon/2m \) we have that,
  - For any \( y \in L_m \), there is a \( z \in L_m' \) with \[
  \frac{y}{(1+\delta)^m} \leq z \leq y
\]
  - Further, \( \text{Opt} \in L_m \), meaning there is a \( z \in L_m' \) with
    \[
    \frac{\text{Opt}}{(1+\delta)^m} \leq z \leq \text{Opt}
    \]

We only need to show that \( (1 + \frac{\epsilon}{2m})^m \leq 1 + \epsilon \ldots \)

### Proof (by induction)

**Base Case:** \( L_0 = \{0\} \)

**Inductive step:** Assume that the lemma holds for \( (i-1) \)

As \( y \in L_i \) we have that either \( y \in L_{i-1} \) or \( (y-s_i) \in L_{i-1} \)

- If \( y \in L_{i-1} \) then there is a \( x \in L_{i-1}' \) with
  \[
  \frac{y}{(1+\delta)^{i-1}} \leq x \leq y
  \]
  By the definition of Trim there is some \( z \in L_i' \) with \[
  z \leq x \leq \frac{y}{(1+\delta)^i} \leq z \leq y
  \]
  i.e. that there is an \( z \in L_i' \) with \[
  \frac{y}{(1+\delta)^i} \leq z \leq y
  \]

The case that \( (y-s_i) \in L_{i-1} \) is almost identical (we omit it for brevity)

### Lemma

We need to show that \( (1 + \frac{\epsilon}{2m})^m \leq 1 + \epsilon \) for \( 0 < \epsilon \leq 1 \)

\[
\left(1 + \frac{\epsilon}{2m}\right)^m \leq e^{\epsilon/2} \leq 1 + \frac{\epsilon}{2} + \left(\frac{\epsilon}{2}\right)^2 \leq 1 + \epsilon
\]

This follows from:

\[
e^x \geq (1 + \frac{x}{m})^m \text{ for all } x, m > 0
\]

\[
e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} \leq 1 + x + x^2
\]

### How big is \( |L_t'| ? \)

The time complexity depends on \( |L_t'| \)

- By the definition of Trim we have that, any two successive elements, \( z, z' \) of \( L_t' \) have
  \[
  \frac{z'}{z} \geq 1 + \delta = 1 + \frac{\epsilon}{2m}
  \]
  - Further, all elements are no greater than \( t \)
  - So \( L_t' \) contains at most \( O(\log(1+\delta) t) \) elements

\[
\ln(1 + x) > \frac{x}{2m} \quad \text{(here } x = \epsilon/2m) \]

\[
\log(1+\delta) t = \frac{\ln t}{\ln(1 + (\epsilon/2m))} \leq \frac{2m(1 + (\epsilon/2m))}{\ln t} \leq 2m \frac{\ln t}{\epsilon} = O\left(\frac{m \log t}{\epsilon}\right)
\]
A PTAS for Subset Sum

The algorithm

- Let $L'_0 = \{0\}, \delta = \epsilon/(2m)$
- For $i = 0, \ldots, m$
  - Compute $(L'_{i-1} + s_i)$ from $L'_{i-1}$
  - Compute $U = L'_{i-1} \cup (L'_{i-1} + s_i)$
  - Let $L'_i = \text{Trim}(U, \delta)$
- Output the largest number in $L'_m$

- As $|L'_i| \in O(m \log t/\epsilon)$, the algorithm runs in $O(m \log t/\epsilon) \in O(n^3/\epsilon)$ time
- The solution outputted is at least $\frac{\text{Opt}}{1+\epsilon} \leq z \leq \text{Opt}$
- So this is in fact an FPTAS for Subset Sum