# Part I

## Subset Sum

## Subset Sum

**Subset Sum**

**Instance**: $X = \{x_1, \ldots, x_n\}$ – $n$ integer positive numbers, $t$ - target number
**Question:** Is there a subset of $X$ such the sum of its elements is $t$?

$M$: Max value input numbers.

R.T. $O\left(Mn^2\right)$.

```
SolveSubsetSum (X, t, M)
    b[0...Mn] - boolean array init to false.
        // b[x] is true if x can be realized
        // as a subset of X.
    b[0] ← true.
    for i = 1,...,n do
        for j = Mn down to x_i do
            b[j] ← b[j − x_i] ∨ b[j]
    return b[t]
```

## Subset Sum

Efficient algorithm???

1. Algorithm solving **Subset Sum** in $O(Mn^2)$.
2. $M$ might be prohibitly large...
3. if $M = 2^n \implies$ algorithm is not polynomial time.
4. **Subset Sum** is **NPC**.
5. Still want to solve quickly even if $M$ huge.
6. Optimization version:

**Subset Sum Optimization**

**Instance**: $(X, t)$: A set $X$ of $n$ positive integers, and a target number $t$.
**Question:** The largest number $\gamma_{\text{opt}}$ one can represent as a subset sum of $X$ which is smaller or equal to $t$.

## Subset Sum

Two-approximation

### Lemma

1. $(X, t)$; *Given instance of* ***Subset Sum***. $\gamma_{\text{opt}} \leq t$: *Opt.*
2. $\implies$ *Compute legal subset with sum* $\geq \gamma_{\text{opt}}/2$.
3. *Running time* $O(n \log n)$.

### Proof.

1. Sort numbers in $X$ in decreasing order.
2. Greedily - add numbers from largest to smallest (if possible).
3. $s$: Generates sum.
4. $u$: First rejected number. $s'$: sum before rejection.
5. $s' > u > 0$, $s' < t$, and $s' + u > t \implies$
   $t < s' + u < s' + s' = 2s' \implies s' \geq t/2$.

$\square$

## Polynomial Time Approximation Schemes

### Definition (PTAS)

**PROB**: **Max**imization problem.
$\varepsilon > 0$: approximation parameter.
$\mathcal{A}(I, \varepsilon)$ is a ***polynomial time approximation scheme*** (**PTAS**) for **PROB**:

1. $\forall I$: $(1-\varepsilon)\left|\mathbf{opt}(I)\right| \leq \left|\mathcal{A}(I,\varepsilon)\right| \leq \left|\mathbf{opt}(I)\right|$,
2. $|\mathbf{opt}(I)|$: opt price,
3. $|\mathcal{A}(I,\varepsilon)|$: price of solution of $\mathcal{A}$.
4. $\mathcal{A}$ running time polynomial in $n$ for fixed $\varepsilon$.

For minimization problem:
$|\mathbf{opt}(I)| \leq |\mathcal{A}(I,\varepsilon)| \leq (1+\varepsilon)|\mathbf{opt}(I)|$.

## Polynomial Time Approximation Schemes

1. Example: Approximation algorithm with running time $O(n^{1/\varepsilon})$ is a PTAS.
   Algorithm with running time $O(1/\varepsilon^n)$ is not.
2. Fully polynomial...

### Definition (FPTAS)

An approximation algorithm is ***fully polynomial time approximation scheme*** (**FPTAS**) if it is a PTAS, and its running time is polynomial both in $n$ and $1/\varepsilon$.

3. Example: PTAS with running time $O(n^{1/\varepsilon})$ is not an FPTAS.
4. Example: PTAS with running time $O(n^2/\varepsilon^3)$ is an FPTAS.

## Approximating Subset Sum

### Subset Sum Approx

**Instance**: $(X, t, \varepsilon)$: A set $X$ of $n$ positive integers, a target number $t$, and parameter $\varepsilon > 0$.
**Question:** A number $z$ that one can represent as a subset sum of $X$, such that $(1-\varepsilon)\gamma_{\mathrm{opt}} \leq z \leq \gamma_{\mathrm{opt}} \leq t$.

## Approximating Subset Sum

Looking again at the exact algorithm

```
ExactSubsetSum(S, t)
    n ← |S|
    P_0 ← {0}
    for i = 1...n do
        P_i ← P_{i-1} ∪ (P_{i-1} + x_i)
        Remove from P_i all elements > t
    return largest element in P_n
```

1. $S = \{a_1, \ldots, a_n\}$
   $x + S = \{a_1 + x, a_2 + x, \ldots a_n + x\}$
2. Lists might explode in size.

## Trim the lists...

```
Trim(L', δ)
    L ← Sort(L')
    L = ⟨y_1 ... y_m⟩
    curr ← y_1
    L_out ← {y_1}
    for i = 2 ... m do
        if y_i > curr · (1 + δ)
            Append y_i to L_out
            curr ← y_i
    return L_out
```

### Definition

For two positive real numbers $z \leq y$, the number $y$ is a $\delta$-approximation to $z$ if $\frac{y}{1+\delta} \leq z \leq y$.

### Observation

*If $x \in L'$ then there exists a number $y \in L_{out}$ such that $y \leq x \leq y(1+\delta)$, where $L_{out} \leftarrow \mathrm{Trim}(L', \delta)$.*

## Trim the lists...

```
Trim(L', δ)
    L ← Sort(L')
    L = ⟨y_1 ... y_m⟩
    curr ← y_1
    L_out ← {y_1}
    for i = 2 ... m do
        if y_i > curr · (1 + δ)
            Append y_i to L_out
            curr ← y_i
    return L_out
```

```
ApproxSubsetSum(S, t)
    // S = {x_1, ..., x_n},
    //        x_1 ≤ x_2 ≤ ... ≤ x_n
    n ← |S|, L_0 ← {0}, δ = ε/2n
    for i = 1 ... n do
        E_i ← L_{i-1} ∪ (L_{i-1} + x_i)
        L_i ← Trim(E_i, δ)
        Remove from L_i elems > t.
    return largest element in L_n
```

## Analysis

1. $E_i$ list generated by algorithm in $i$th iteration.
2. $P_i$: list of numbers (no trimming).

### Claim

*For any $x \in P_i$ there exists $y \in L_i$ such that $y \leq x \leq (1+\delta)^i y$.*

### Proof

1. If $x \in P_1$ then follows by observation above.
2. If $x \in P_{i-1} \implies$ (induction) $\exists y' \in L_{i-1}$ s.t. $y' \leq x \leq (1+\delta)^{i-1} y'$.
3. By observation $\exists y \in L_i$ s.t. $y \leq y' \leq (1+\delta) y$. Therefore

$$y \leq y' \leq x \leq (1+\delta)^{i-1} y' \leq (1+\delta)^i y.$$

## Proof continued

### Proof continued

1. If $x \in P_i \setminus P_{i-1} \implies x = \alpha + x_i$, for some $\alpha \in P_{i-1}$.
2. By induction, $\exists \alpha' \in L_{i-1}$ s.t. $\alpha' \leq \alpha \leq (1+\delta)^{i-1} \alpha'$.
3. Thus, $\alpha' + x_i \in E_i$.
4. $\exists x' \in L_i$ s.t. $x' \leq \alpha' + x_i \leq (1+\delta) x'$.
5. Thus, $x' \leq \alpha' + x_i \leq \alpha + x_i = x \leq (1+\delta)^{i-1}\alpha' + x_i \leq (1+\delta)^{i-1}(\alpha' + x_i) \leq (1+\delta)^i x'$. ■

## Running time of ApproxSubsetSum

### Lemma

*For $x \in [0, 1]$, it holds $e^{x/2} \leq (1 + x)$.*

### Lemma

*For $0 < \delta < 1$, and $x \geq 1$, we have*

$$\log_{1+\delta} x \leq \frac{2 \ln x}{\delta} = O\left(\frac{\ln x}{\delta}\right).$$

See notes for a proof of lemmas.

## Running time of ApproxSubsetSum

### Observation

*In a list generated by **Trim**, for any number $x$, there are no two numbers in the trimmed list between $x$ and $(1 + \delta)x$.*

### Lemma

$|L_i| = O\big((n/\varepsilon^2) \log n\big)$, *for* $i = 1, \ldots, n$.

## Running time of ApproxSubsetSum

### Proof.

1. $L_{i-1} + x_i \subseteq [x_i, ix_i]$.
2. Trimming $L_{i-1} + x_i$ results in list of size

$$\log_{1+\delta} \frac{ix_i}{x_i} = O\left(\frac{\ln i}{\delta}\right) = O\left(\frac{\ln n}{\delta}\right),$$

3. Now, $\delta = \varepsilon/2n$, and

$$|L_i| \leq |L_{i-1}| + O\left(\frac{\ln n}{\delta}\right) \leq |L_{i-1}| + O\left(\frac{n \ln n}{\varepsilon}\right)$$
$$= O\left(\frac{n^2 \log n}{\varepsilon}\right).$$

□

## Running time of ApproxSubsetSum

### Lemma

*The running time of **ApproxSubsetSum** is* $O\big(\frac{n^3}{\varepsilon} \log^2 n\big)$.

### Proof.

1. Running time of **ApproxSubsetSum** dominated by total length of $L_1, \ldots, L_n$.
2. Above lemma implies $\sum_i |L_i| = O\left(\frac{n^3}{\varepsilon} \log n\right)$.
3. **Trim** sorts lists. $i$th iteration R.T. $O(|L_i| \log |L_i|)$.
4. Overall, R.T. $O(\sum_i |L_i| \log |L_i|) = O\big(\frac{n^3}{\varepsilon} \log^2 n\big)$.

□

## ApproxSubsetSum

**Theorem**

**ApproxSubsetSum** *returns* $u \le t$, *s.t.*
$\frac{\gamma_{\text{opt}}}{1+\varepsilon} \le u \le \gamma_{\text{opt}} \le t$,
$\gamma_{\text{opt}}$*: opt solution.*
*Running time is* $O\left((n^3/\varepsilon)\log^2 n\right)$.

**Proof.**

1. Running time from above.
2. $\gamma_{\text{opt}} \in P_n$: optimal solution.
3. $\exists z \in L_n$, such that $z \le \mathbf{opt} \le (1+\delta)^n z$
4. $(1+\delta)^n = (1+\varepsilon/2n)^n \le e^{\varepsilon/2} \le 1+\varepsilon$, since $1 + x \le e^x$ for $x \ge 0$.
5. $\gamma_{\text{opt}}/(1+\varepsilon) \le z \le \mathbf{opt} \le t$.

$\square$