

CMSC 451 - Algorithm Design

Lecture 21 - Subset Sum

Subset Sum (SS):

Given a set of positive integers $S = \{x_1, \dots, x_n\}$ + target value t , does there exist a subset $S' \subseteq S$, s.t.

$$\sum_{x \in S'} x = t \quad ?$$

Example: $S = \{3, 6, 9, 12, 15, 23, 32\}$ $t = 33$

→ "yes" $S' = \{6, 12, 15\}$ $6 + 12 + 15 = 33 \checkmark$

(but "no" if $t = 34$)

Application:

- A simplified version of 0-1 Knapsack and various packing problems.

This lecture:

- (1) SS is NP-complete
- (2) Approximation algorithm
PTAS - ϵ -error, running grows with $1/\epsilon$.

SS is NP-Complete:

(i) $SS \in NP$ (easy)

- certificate: indices of elements in S'
- verification: add up and check = t

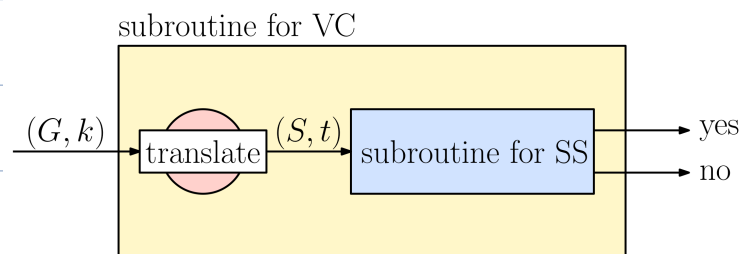
(ii) SS is NP-hard

- We'll show $VC \leq_p SS$

- Give poly-time function f :

$$f(G, k) \longrightarrow (S', t), \text{ s.t.}$$

G has VC of size k iff S' has subset sums to t



Intuition:



- Why is computing sums like building a cover?

- Numbers

$\equiv$ bit strings

$\downarrow \equiv$ sets (bit vector)

Addition

$\approx$ boolean-or

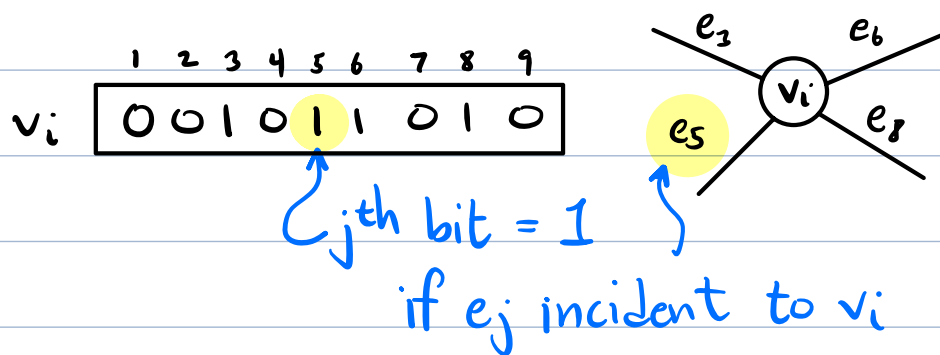
$\downarrow \approx$ set union

First Attempt: Given graph $G = (V, E)$ and k , does G have a vertex cover of size k ?

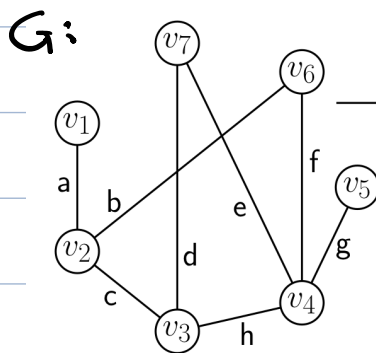
- Let: $n = |V|$ $m = |E|$

$V = \{v_1, \dots, v_n\}$ $E = \{e_1, \dots, e_m\}$

- For each $v_i \in V$ create bit vector:



Example:

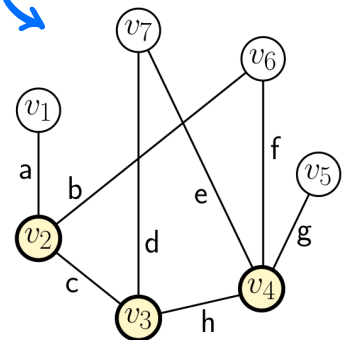


$k=3$

	a	b	c	d	e	f	g	h
v_1	1	0	0	0	0	0	0	0
v_2	1	1	1	0	0	0	0	0
v_3	0	0	1	1	0	0	0	1
v_4	0	0	0	0	1	1	1	1
v_5	0	0	0	0	0	0	1	0
v_6	0	1	0	0	0	1	0	0
v_7	0	0	0	1	1	0	0	0

$t = 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 = v_2 \vee v_3 \vee v_4$

Vertex cover $\{v_2, v_3, v_4\}$



Every edge must be covered
 $\Rightarrow$ logical-or has 1 in every position

Observe:

G has vertex cover of size k
iff

There are k bit vectors whose "or" is $1111\dots 1$

Not what we want:

- ① logical-or $\neq$ sum
- ② subset sum has no limit on number of items in sum

Second Attempt: (Correct)

- Addition has carries

$$\begin{array}{r} 1101 \\ \vee 0011 \\ \hline 1111 \end{array} \neq \begin{array}{r} 1101_2 \\ + 0011_2 \\ \hline 10000_2 \end{array}$$

Solution - Use larger base to avoid carries

$$\begin{array}{r} 1101 \\ \vee 0011 \\ \hline 1111 \end{array} \quad \begin{array}{r} 1101_4 \\ + 0011_4 \\ \hline 1112_4 \end{array} \quad \leftarrow \text{Base 4 } \{0,1,2,3\}$$

$\longleftrightarrow$
(closer)

- Each edge may covered once or twice
 $\Rightarrow$ final sum has $1_i + 2_i$: 121122_4

Solution: Create a set of slack values with 1 digit in each. Use with discretion to make $\text{sum} = 222\dots 2_4$

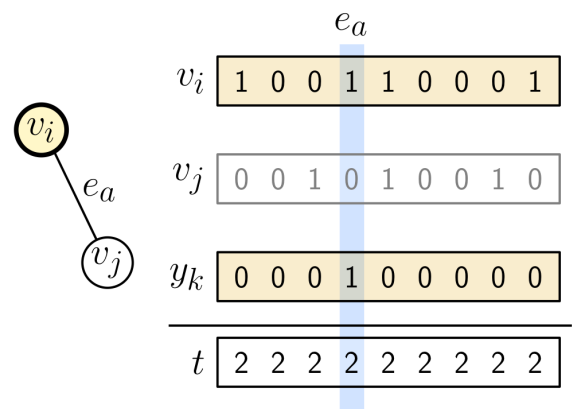
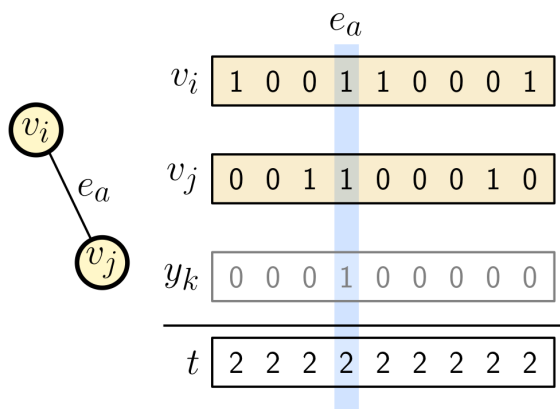
vertex sum: 1 2 1 1 2 2₄

slacks:

y_1	1	0	0	0	0	0	Select to bump 1-digits up to 2
	0	1	0	0	0	0	
y_2	0	0	1	0	0	0	
y_3	0	0	0	1	0	0	
y_4	0	0	0	0	1	0	
y_5	0	0	0	0	0	1	

$t = 2\ 2\ 2\ 2\ 2\ 2_4$

Example:



- How to control size k of vertex cover?

- Add column of 1's

- Adjust t so this column sums to k

$\Rightarrow k$ values selected $\Rightarrow k$ vertices selected

Here's the whole reduction: Given $G=(V,E)$, k

- Let $n=|V| + m=|E|$

- Create n vertex values in base 4, $\{v_1, \dots, v_n\}$

For v_i : $m+1$ digits

- Leading digit = 1

- If v_i incident to edge e_j set j^{th} digit to 1 (else 0)

- Create m slack values in base 4, $\{y_1, \dots, y_m\}$

For y_j : $m+1$ digits

- Leading digit = 0

- Set j^{th} digit to 1 + all others = 0

- Set t to $m+1$ digits in base 4

- Leading digit = k

- Remaining m digits = 2

- Convert numbers from base 4 to base 10

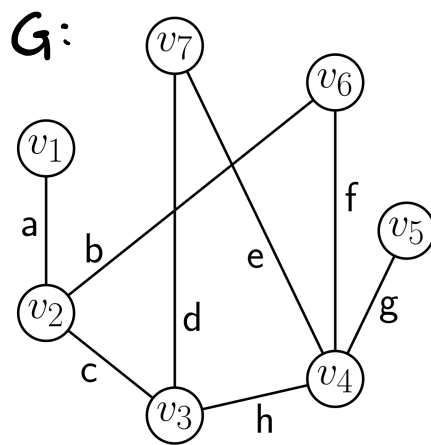
(or whatever subset sum expects) $\rightarrow (S, t)$

Time - Total of $n+m$ numbers, each with $m+1$ digits

$\Rightarrow O(m(n+m))$ - polynomial



Example:



$k=3$

Vertex cover
 $= \{v_2, v_3, v_4\}$

	a	b	c	d	e	f	g	h
v_1	1	1	0	0	0	0	0	0
v_2	1	1	1	1	0	0	0	0
v_3	1	0	0	1	1	0	0	1
v_4	1	0	0	0	0	1	1	1
v_5	1	0	0	0	0	0	1	0
v_6	1	0	1	0	0	0	1	0
v_7	1	0	0	0	1	1	0	0
y_a	0	1	0	0	0	0	0	0
y_b	0	0	1	0	0	0	0	0
y_c	0	0	0	1	0	0	0	0
y_d	0	0	0	0	1	0	0	0
y_e	0	0	0	0	0	1	0	0
y_f	0	0	0	0	0	0	1	0
y_g	0	0	0	0	0	0	0	1
y_h	0	0	0	0	0	0	0	1
$t =$	3	2	2	2	2	2	2	2

vertex values

slack values

k

All numbers in base 4

	a	b	c	d	e	f	g	h
v_1	1	1	0	0	0	0	0	0
v_2	1	1	1	1	0	0	0	0
v_3	1	0	0	1	1	0	0	1
v_4	1	0	0	0	0	1	1	1
v_5	1	0	0	0	0	0	1	0
v_6	1	0	1	0	0	0	1	0
v_7	1	0	0	0	1	1	0	0
y_a	0	1	0	0	0	0	0	0
y_b	0	0	1	0	0	0	0	0
y_c	0	0	0	1	0	0	0	0
y_d	0	0	0	0	1	0	0	0
y_e	0	0	0	0	0	1	0	0
y_f	0	0	0	0	0	0	1	0
y_g	0	0	0	0	0	0	0	1
y_h	0	0	0	0	0	0	0	1
$t =$	3	2	2	2	2	2	2	2

take as needed to make 2's

Counts number of vertices in cover

Correctness:

Lemma: G has a vertex cover of size k iff S has a subset that sums to t .

Proof: ($\Rightarrow$) Suppose G has vertex cover V' , $|V'| = k$

To construct subset sum S' :

- for each $v_i \in V'$:

 - add vertex value v_i to S'

- for each edge $e_j \in E$:

 - if only one endpt in V' , add y_j to S'

- Claim: S' sums to $t = [k \ 2 \ 2 \ 2 \dots 2_q]$

 - Since $|V'| = k$, k values contribute 1 to leading digit position

 - Every edge e_j is covered:

 - If once - take vertex + $y_j = 2 \checkmark$

 - If twice - take both vertices = $2 \checkmark$

($\Leftarrow$) Suppose S has subset S' sums to t .

- To match leading digit, there are k vertex values $\rightarrow V'$

- The only way to get $222\dots 2$ is for each column to contribute

 - one or two vertices to V' . Why?

 - slacks can only increase counts by $+1$

 - no carries possible because only two 1-digits per column.

□

Polynomial-Time Approximation to Subset Sum

- We'll present a poly-time algorithm for SS which given ϵ , where $0 < \epsilon < 1$, returns a subset $S' \subseteq S$ whose sum z , satisfies

$$(1-\epsilon) z^* \leq z \leq z^* \leq t$$

where z^* is the optimum sum.

Exact (Exponential time) Algorithm

- For $0 \leq i \leq n$, $L_i =$ set of all possible sums of $\{x_1, \dots, x_i\}$

E.g. $S = \{1, 4, 6, \dots\}$ $L_0 = \langle 0 \rangle$

$$L_1 = \langle 0, 1 \rangle$$

$$L_2 = \langle 0, 1, 4, 5 \rangle$$

$$L_3 = \langle 0, 1, 4, 5, 6, 7, 10, 11 \rangle$$

$$|L_i| \leq 2^i$$

- Final answer: Largest in L_n that is $\leq t$.

$$z^* = \max \{ z \in L_n \mid z \leq t \}$$

- Approach:

- Compute L_i for $i = 0, 1, \dots, n$

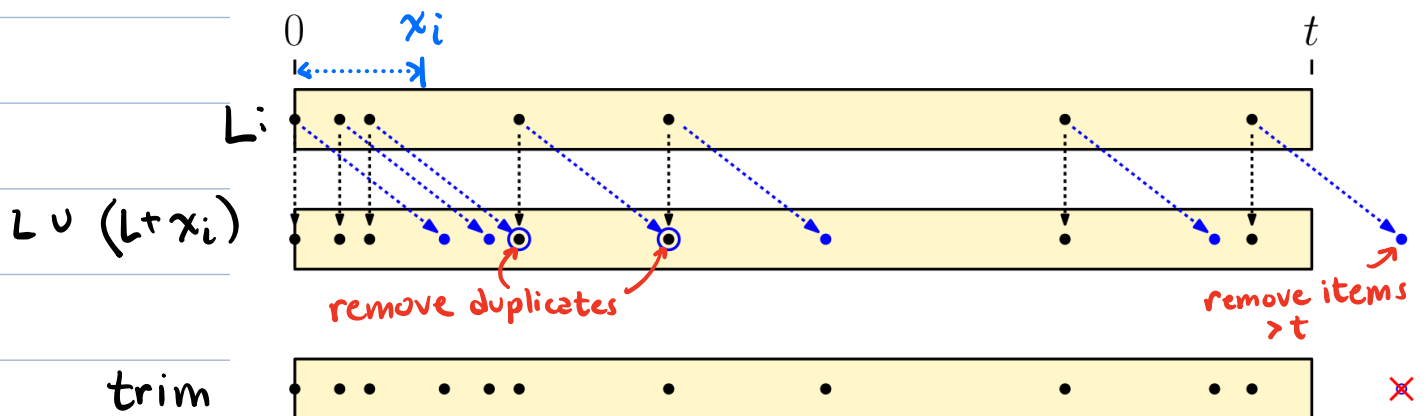
Define: $L + x = \text{add } x \text{ to all } z \in L$

E.g. $L = \langle 0, 3, 5 \rangle \rightarrow L + 4 = \langle 4, 7, 9 \rangle$

$L_i \leftarrow L_{i-1} \cup (L_{i-1} + x_i) \} \text{merge}(L, L + x_i)$

- Remove duplicates } $\text{trim}(L, t)$
- Remove items $> t$

- Pictorially:



exact-subset-sum ($x[1..n], t$) // exact subset sum

$L \leftarrow \langle 0 \rangle$

// basis

for ($i \leftarrow 1$ to n)

$L \leftarrow \text{merge}(L, L + x[i])$ // $L \leftarrow L \cup (L + x_i)$

$L \leftarrow \text{trim}(L, t)$ // rem. dups & $> t$

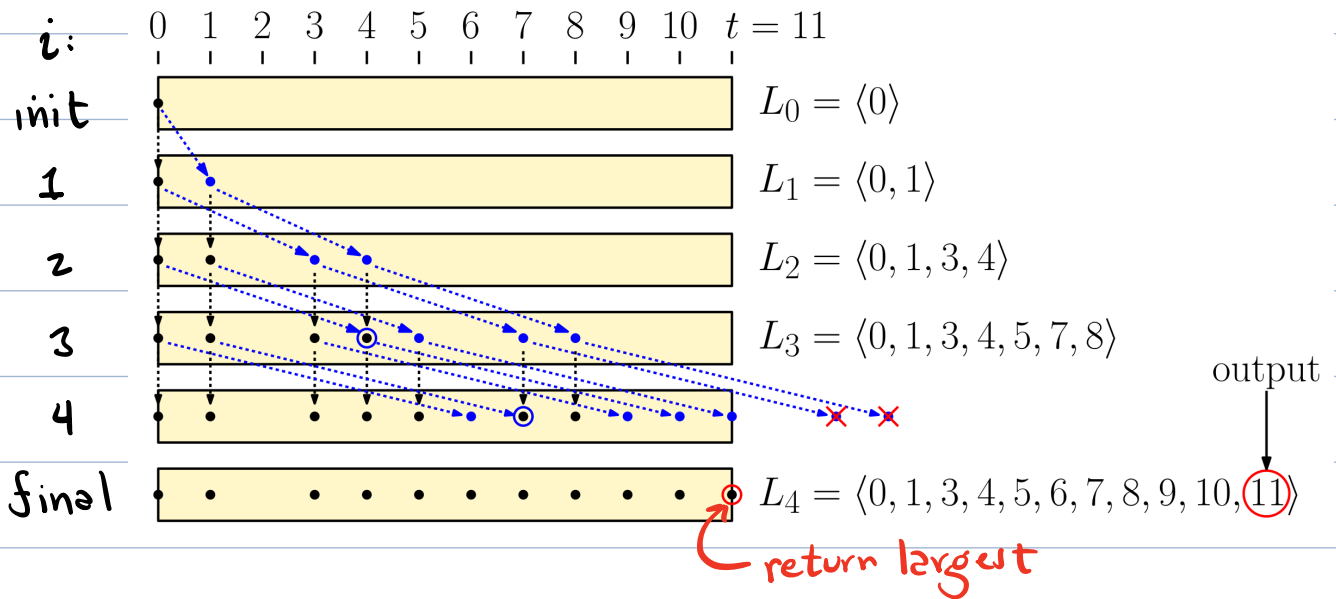
return largest in L

Correctness - Easy

Running Time = $O(2^n)$ exponential!



Example:

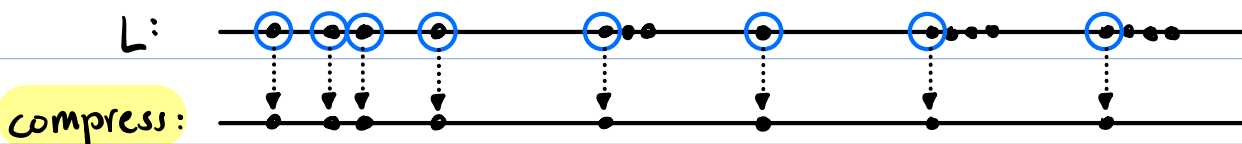


Approximation Algorithm

- Too many distinct sums (potentially 2^n)
- Idea: compress nearly equal sums

$L = \langle \dots, 9851, 9852, 9860, 9950, 9952, 10,301, \dots \rangle$

compress to 9851 compress to 9950



How large a difference can we tolerate?

prev = previous unpruned value

y = element under consideration

Cannot prune y if relative difference $> \delta$:

$$\frac{y - \text{prev}}{y} > \delta \iff y > \frac{\text{prev}}{1 - \delta}$$

```
compress(L,  $\delta$ , t) // compress sorted list L
├── L'  $\leftarrow \emptyset$ 
├── prev  $\leftarrow 0$ 
├── for each y  $\in$  L
│   ├── if (y > t) break // ignore values > t
│   ├── if (y > prev / (1 -  $\delta$ )) // different enough?
│   │   ├── append y to L' // add to L
│   │   └── prev  $\leftarrow$  y
└── return L' // final compressed list
```

Example: $L = \langle 10, 11, 12, 15, 20, 21, 22, 51, 53, 54, 56, 80 \rangle$

$\delta = 0.1$ $t = 60$

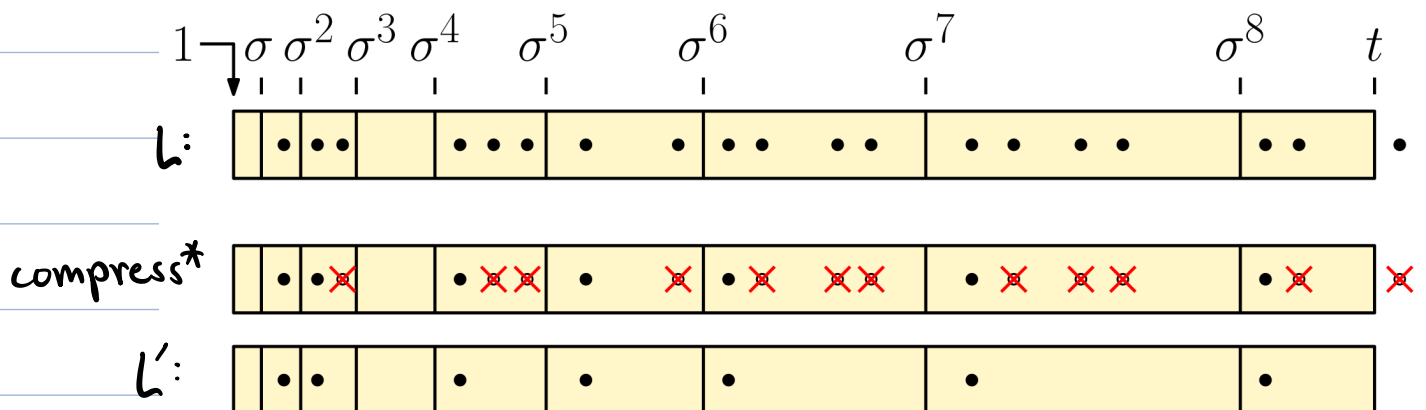
$L' = \langle \textcircled{10}, \textcircled{12}, \textcircled{15}, \textcircled{20}, \textcircled{51}, \textcircled{53}, \textcircled{54}, \textcircled{56}, \textcircled{80} \rangle$ $> t$

return $\langle 10, 12, 15, 20, 51 \rangle$

"Bucket" Perspective:

- A bit easier to visualize in terms of buckets of same relative size
- Let $\sigma = 1/(1-\delta)$ ($\sigma > 1$)
- Breakpoints: $1, \sigma, \sigma^2, \sigma^3, \dots, \sigma^i, \dots$
- Bucket j : $[\sigma^{j-1}, \sigma^j]$

$\text{compress}^*(L, \delta, t)$ - keep smallest in each bucket up to t



- Not exactly same as compress , but same properties. If y, y' in same bucket then $\sigma^{i-1} \leq y < y' \leq \sigma^i$

$$\frac{y' - y}{y'} = 1 - \frac{y}{y'} \leq 1 - \frac{\sigma^{i-1}}{\sigma^i} = 1 - \frac{1}{\sigma} = \delta$$

$$\sigma = 1/(1-\delta)$$

$\Rightarrow$ in compress , y would prune y'

$$\Rightarrow 1 - \frac{1}{\sigma} = \delta$$

Value of compression – Number of surviving values is polynomial, not exponential!

Lemma: After $\text{compress}(L, \delta, t)$ where $\delta = \epsilon/n$, number of nonzero items in L is $O((n \log t)/\epsilon)$

Proof: Each consecutive pair of survivors is separated by factor of $\geq \sigma$, where $\sigma = 1/(1-\delta)$

- Let k = number of nonzero survivors

- $L_{\text{final}} = \langle 0, y_1, y_2, \dots, y_k \rangle$ s.t. $y_i \geq \sigma \cdot y_{i-1}$

$$- y_1 \geq 1 \Rightarrow y_i \geq \sigma^{i-1} \cdot y_1 \geq \sigma^{i-1}$$

$$\Rightarrow y_k \geq \sigma^{k-1} \text{ but } y_k \leq t$$

$$\Rightarrow \sigma^{k-1} \leq t$$

$$\Rightarrow (k-1) \ln \sigma \leq \ln t$$

$$\begin{aligned} \Rightarrow k &\leq 1 + \ln t / \ln \sigma \\ &= 1 + \ln t / \ln 1/(1-\delta) \\ &= 1 + \ln t / (-\ln(1-\delta)) \end{aligned}$$

Standard inequality: $-\ln(1-x) \geq x$

$$\Rightarrow k \leq 1 + (\ln t) / \delta$$

$$\delta = \epsilon/n \Rightarrow k \leq 1 + (n \cdot \ln t) / \epsilon = O\left(\frac{n \cdot \log t}{\epsilon}\right) \quad \square$$

Assuming $x_i \leq t$, each number needs $O(\log t)$ bits

- Total input size = $O(n \log t)$

- List size $O((n \log t)/\epsilon) = \frac{1}{\epsilon} \cdot O(\text{input size})$

- Polynomial!

approx-subset-sum ($x[1..n], t, \epsilon$)

$$\delta \leftarrow \varepsilon / n$$

// pruning resolution

$$L \leftarrow \langle 0 \rangle$$

// basis case

for ($i \leftarrow 1$ to n)

$L \leftarrow \text{merge}(L, L + x[i])$

```
// add in  $x_i$ 
```

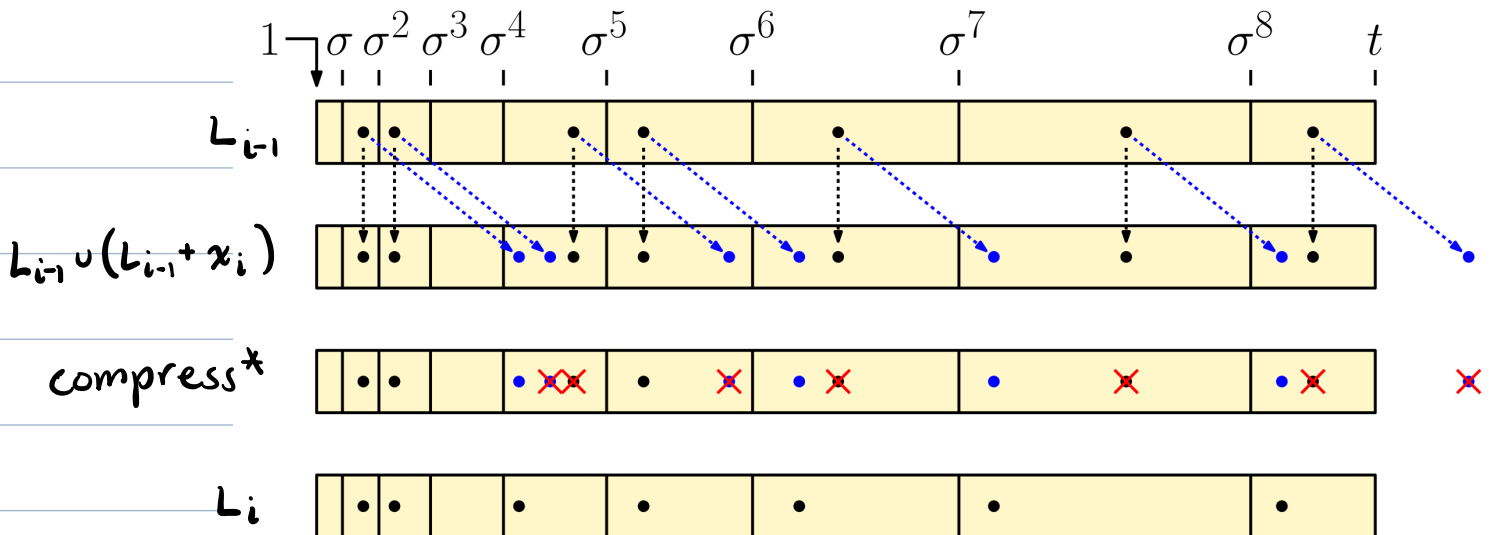
$L \leftarrow \text{compress}(L, f, t)$

// + compress

return largest item in L

// largest survivor

Example: (using `compress*` rather than `compress`)



Approximation Analysis:

Let Y^* = optimum subset sum value

Y_ϵ = algorithm output

→ Although we only output sum, there is a subset summing to Y_ϵ .

We know: $Y_\epsilon \leq Y^* \leq t$

want to show $Y_\epsilon \geq (1-\epsilon)Y^*$

Intuition:

- Algorithm has n stages
- We allowed an error of $\delta = \epsilon/n$ per stage
- Total error $\leq n \cdot \delta = \epsilon$ 😊
- But these are multiplicative, not additive errors 😞
- We'll show the intuition still works

Define:

L_i = algorithm's list after i^{th} iteration

L_i^* = exact list after i^{th} iteration

(no compression)

Claim that for $y \in L_i^*$ there is "close" $z \in L_i$

Lemma: For $0 \leq i \leq n$, for all $y \in L_i^*$, there exists $z \in L_i$, s.t.

$$(1-\delta)^i y \leq z \leq y,$$

where $\delta = \epsilon/n$.

called y 's representative

Proof: We'll just prove lower bound:

$$(1-\delta)^i y \leq z$$

(Upper bound is an easy extension)

By induction on i . Basis: $L_0 = L_0^* = \langle 0 \rangle \checkmark$

Let's assume lemma holds for $i-1$ & we'll show i

Every element of L_i^* arises in two ways:

$$y \in L_{i-1}^*$$

$$\text{or } y + x_i, \text{ where } y \in L_{i-1}^*$$

We'll show both y & $y + x_i$ have representatives

By ind. hypothesis, $\exists z \in L_{i-1}$ s.t.

$$(1-\delta)^{i-1} y \leq z \quad \textcircled{a}$$

By adding x_i to both, we have

$$(1-\delta)^{i-1} y + x_i \leq z + x_i$$

$$\Rightarrow (1-\delta)^{i-1} (y + x_i) \leq z + x_i \quad (b) \text{ (since } 1-\delta < 1)$$

By compression, neither z nor $z + x_i$ in L_i^*

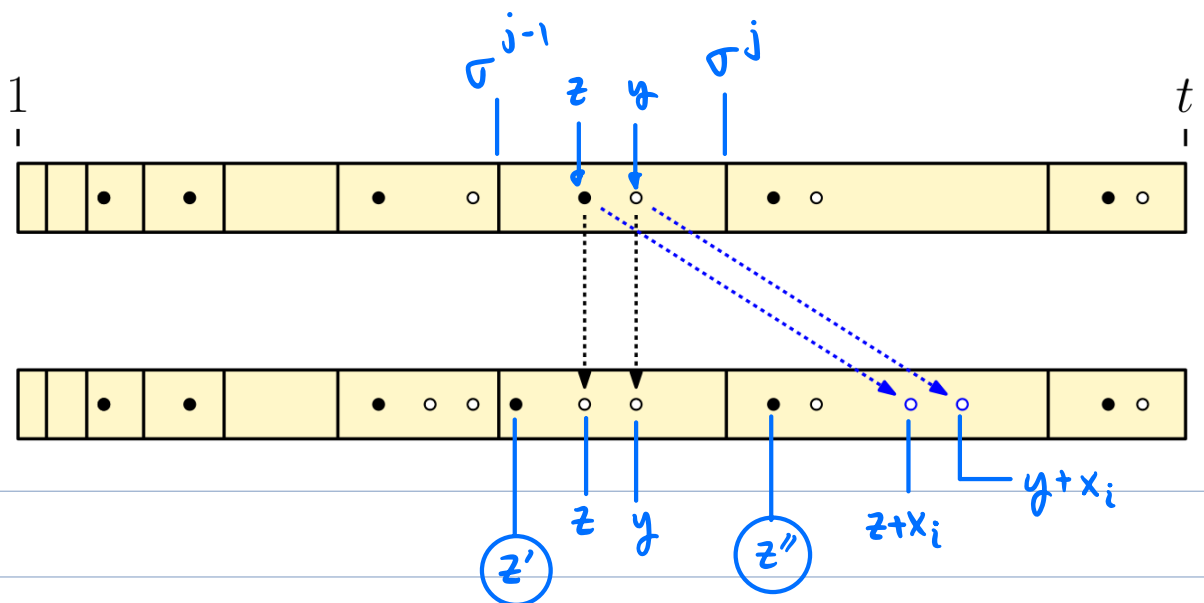
Let $z' + z''$ be elements that pruned $z + z + x_i$

By rules of compression:

$$(1-\delta)z \leq z'$$

$$(1-\delta)(z + x_i) \leq z''$$

Illustration in compress^{*}:



Combining (a), (b), (c):

$$(1-\delta)^{i-1} (1-\delta) y \leq z' \Rightarrow (1-\delta)^i y \leq z'$$

$$(1-\delta)^{i-1} (1-\delta) (y+x_i) \leq z'' \Rightarrow (1-\delta)^i (y+x_i) \leq z''$$

Since $z' + z'' \in L_i$, this completes proof $\square$

Applying lemma with: $i=n$, $y=Y^*$ (opt s.s.)
 $z=Y$ (approx s.s.)

we have:

$$\left(1 - \frac{\varepsilon}{n}\right)^n Y^* \leq Y \leq Y^*$$

Standard lemma: $\forall n > 0 + \forall a \in \mathbb{R}$

$$(1+a) \leq \left(1 + \frac{a}{n}\right)^n$$

Set $a = -\varepsilon$

$$(1-\varepsilon) \leq \left(1 - \frac{\varepsilon}{n}\right)^n$$

$$\Rightarrow (1-\varepsilon) Y^* \leq Y \leq Y^*$$

$\Rightarrow$ approx-subset-sum yields an ε -approximation $\checkmark$