Algorithms

Lecture Topic: Approximation Algorithms (Part 4)

Anxiao (Andrew) Jiang

Roadmap of this lecture:

1. Understand approximation algorithms by solving the "Subset-Sum Problem".
   - 1.1 Define "Subset-Sum Problem".
   - 1.2 Define "Fully Polynomial-Time Approximation Scheme (FPTAS)".
   - 1.3 An exponential-time exact algorithm for "Subset-Sum Problem".
   - 1.4 FPTAS for "Subset-Sum Problem".
   - 1.5 Prove the correctness of the FPTAS.

The Subset-Sum Problem

Subset-Sum Problem (As a decision problem)

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: Does S have a subset that adds up exactly to the target value t?

The Subset-Sum Problem

Subset-Sum Problem (As a decision problem)

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: Does S have a subset that adds up exactly to the target value t?

Example: $S = \{1,2,7\}$, $t = 8$.

Answer: YES.

$S' = \{1,7\}$.

The Subset-Sum Problem

Subset-Sum Problem (As a decision problem)

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: Does S have a subset that adds up exactly to the target value t?

Example: $S = \{1,2,7\}$, $t = 6$.

Answer: NO.

The Subset-Sum Problem

Subset-Sum Problem (As a decision problem)

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: Does S have a subset that adds up exactly to the target value t?

Example: $S = \{1,2,7,14,49,98,343,686,2409,2793,16808,17206,117705,117993\}$

$t = 138457$

The Subset-Sum Problem

Subset-Sum Problem (As a decision problem)

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: Does S have a subset that adds up exactly to the target value t?

Example: $S = \{1,2,7,14,49,98,343,686,2409,2793,16808,17206,117705,117993\}$

$t = 138457$

Answer: Yes.

The Subset-Sum Problem

Subset-Sum Problem (As a decision problem) $\in NPC$.

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: Does S have a subset that adds up exactly to the target value t?

The Subset-Sum Problem

Subset-Sum Problem (As a decision problem) $\in NPC$.

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: Does S have a subset that adds up exactly to the target value t?

Subset-Sum Problem (As an optimization problem)

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

The Subset-Sum Problem

Subset-Sum Problem (As a decision problem) $\in NPC$.

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: Does S have a subset that adds up exactly to the target value t?

Subset-Sum Problem (As an optimization problem)

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.

A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An example application: load goods onto a truck that has a weight capacity constraint.

Quiz question:

1. Can the optimization problem for the "Subset-Sum Problem" be solved in polynomial time?

Roadmap of this lecture:

1. Understand approximation algorithms by solving the "Subset-Sum Problem".
   - 1.1 Define "Subset-Sum Problem".
   - 1.2 Define "Fully Polynomial-Time Approximation Scheme (FPTAS)".
   - 1.3 An exponential-time exact algorithm for "Subset-Sum Problem".
   - 1.4 FPTAS for "Subset-Sum Problem".
   - 1.5 Prove the correctness of the FPTAS.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Roadmap of this lecture:

1. We show an exponential-time algorithm to compute the optimal solution for this problem.
2. We modify it to a fully polynomial-time approximation scheme (FPTAS).

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Roadmap of this lecture:

1. We show an exponential-time algorithm to compute the optimal solution for this problem.
2. We modify it to a fully polynomial-time approximation scheme (FPTAS).

Approximation Scheme: An approximaiton scheme for an optimization problem is an approximation algorithm that takes as input not only an instance of the problem, but also a value $\epsilon > 0$ such that for any fixed ϵ, the scheme is a $(1+\epsilon)$-approximation algorithm.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Roadmap of this lecture:

1. We show an exponential-time algorithm to compute the optimal solution for this problem.
2. We modify it to a fully polynomial-time approximation scheme (FPTAS).

Approximation Scheme: An approximaiton scheme for an optimization problem is an approximation algorithm that takes as input not only an instance of the problem, but also a value $\epsilon > 0$ such that for any fixed ϵ, the scheme is a $(1 + \epsilon)$-approximation algorithm.

Polynomial-Time Approximation Scheme (PTAS): An approximaiton scheme is a polynomial-time approximation scheme if for any fixed $\epsilon > 0$, the scheme runs in time polynomial in the size n of its input.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Roadmap of this lecture:

1. We show an exponential-time algorithm to compute the optimal solution for this problem.
2. We modify it to a fully polynomial-time approximation scheme (FPTAS).

Approximation Scheme: An approximaiton scheme for an optimization problem is an approximation algorithm that takes as input not only an instance of the problem, but also a value $\epsilon > 0$ such that for any fixed ϵ, the scheme is a $(1 + \epsilon)$-approximation algorithm.

Polynomial-Time Approximation Scheme (PTAS): An approximaiton scheme is a polynomial-time approximation scheme if for any fixed $\epsilon > 0$, the scheme runs in time polynomial in the size n of its input.

Fully Polynomial-Time Approximation Scheme (FPTAS): An approximaiton scheme is a fully polynomial-time approximation scheme if its running time is polynomial in both $1/\epsilon$ and the size n of the input instance.

Example: $O((1/\epsilon)^2 n^3)$

Quiz question:

1. What is the difference between an Approximation Scheme, a PTAS, and an FPTAS?

Roadmap of this lecture:

1. Understand approximation algorithms by solving the "Subset-Sum Problem".
   1.1 Define "Subset-Sum Problem".
   1.2 Define "Fully Polynomial-Time Approximation Scheme (FPTAS)".
   1.3 An exponential-time exact algorithm for "Subset-Sum Problem".
   1.4 FPTAS for "Subset-Sum Problem".
   1.5 Prove the correctness of the FPTAS.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:
1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:

1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.
3. To compute the subset sums from $\{x_1, x_2, \cdots, x_{i+1}\}$, we can use the subset sums from $\{x_1, x_2, \cdots, x_i\}$.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:

1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.
3. To compute the subset sums from $\{x_1, x_2, \cdots, x_{i+1}\}$, we can use the subset sums from $\{x_1, x_2, \cdots, x_i\}$.
4. There is no need to memorize the subset sums larger than t.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:

1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.
3. To compute the subset sums from $\{x_1, x_2, \cdots, x_{i+1}\}$, we can use the subset sums from $\{x_1, x_2, \cdots, x_i\}$.
4. There is no need to memorize the subset sums larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:
1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.
3. To compute the subset sums from $\{x_1, x_2, \cdots, x_{i+1}\}$, we can use the subset sums from $\{x_1, x_2, \cdots, x_i\}$.
4. There is no need to memorize the subset sums larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$ $\quad$ L_0 : the list of subset-sums from { }
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:

1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.
3. To compute the subset sums from $\{x_1, x_2, \cdots, x_{i+1}\}$, we can use the subset sums from $\{x_1, x_2, \cdots, x_i\}$.
4. There is no need to memorize the subset sums larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

L_0 : the list of subset-sums from $\{\ \}$

L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:
1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.
3. To compute the subset sums from $\{x_1, x_2, \cdots, x_{i+1}\}$, we can use the subset sums from $\{x_1, x_2, \cdots, x_i\}$.
4. There is no need to memorize the subset sums larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. Remove from L_i every element that is greater than t
5. return the largest element in L_n

L_0 : the list of subset-sums from $\{\ \}$

L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t

$L_{i-1} + x_i$: add x_i to each subset-sum in L_{i-1}

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:
1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.
3. To compute the subset sums from $\{x_1, x_2, \cdots, x_{i+1}\}$, we can use the subset sums from $\{x_1, x_2, \cdots, x_i\}$.
4. There is no need to memorize the subset sums larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$ — L_0 : the list of subset-sums from $\{\ \}$
2. for $i = 1$ to n — L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$ — $L_{i-1} + x_i$: add x_i to each subset-sum in L_{i-1}
4. $\quad$ Remove from L_i every element that is greater than t — no need to memorize such subset-sums
5. return the largest element in L_n

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

An exponential-time exact algorithm

Basic idea:
1. For each subset S' of S, compute the sum of its elements.
2. Then select the subset whose sum is as large as possible but no more than t.
3. To compute the subset sums from $\{x_1, x_2, \cdots, x_{i+1}\}$, we can use the subset sums from $\{x_1, x_2, \cdots, x_i\}$.
4. There is no need to memorize the subset sums larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$ — L_0 : the list of subset-sums from { }
2. for $i = 1$ to n — L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$ — $L_{i-1} + x_i$: add x_i to each subset-sum in L_{i-1}
4. $\quad$ Remove from L_i every element that is greater than t — no need to memorize such subset-sums
5. return the largest element in L_n — largest subset-sum from S no more than t

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

Example: $L = \langle 1, 2, 3, 5, 9 \rangle$

$$L + 2 = \langle 3, 4, 5, 7, 11 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

Example:

$$\begin{array}{rcl} L & = & \langle 1,\ 2,\ 3,\ 5,\ 9 \rangle \\ & & \quad \downarrow \ \ \downarrow \ \ \downarrow \ \ \downarrow \ \ \downarrow \quad +2 \\ L + 2 & = & \langle 3,\ 4,\ 5,\ 7,\ 11 \rangle \end{array}$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

Example:

$$\begin{array}{rcl} L & = & \langle 1,\ 2,\ 3,\ 5,\ 9 \rangle \\ & & \quad \downarrow\ \ \downarrow\ \ \downarrow\ \ \downarrow\ \ \downarrow \quad +2 \\ L+2 & = & \langle 3,\ 4,\ 5,\ 7,\ 11 \rangle \end{array}$$

$$S + x = \{s + x \mid s \in S\}$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. Remove from L_i every element that is greater than t
5. return the largest element in L_n

Each L_i is a sorted list.

MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$ has time complexity $O(L_{i-1})$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t

P_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t

P_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$

Example:

$S = \{\ 1,\ 4,\ 5\ \}$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t

P_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$

Example:

$S = \{\ 1,\ 4,\ 5\ \}$

$P_1 = \langle 0,\ 1 \rangle$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t

P_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$

Example:

$S = \{\ 1,\ 4,\ 5\ \}$

$P_1 = \langle 0,\ 1 \rangle$

$P_2 = P_1 \cup (P_1 + 4) = \langle 0,1 \rangle \cup \langle 4,5 \rangle = \langle 0,\ 1,\ 4,\ 5 \rangle$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. Remove from L_i every element that is greater than t
5. return the largest element in L_n

L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t

P_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$

Example:

$$S = \{\ 1,\ 4,\ 5\ \}$$

$$P_1 = \langle 0,\ 1 \rangle$$

$$P_2 = P_1 \cup (P_1 + 4) = \langle 0,1 \rangle \cup \langle 4,5 \rangle = \langle 0,\ 1,\ 4,\ 5 \rangle$$

$$P_3 = P_2 \cup (P_2 + 5) = \langle 0,1,4,5 \rangle \cup \langle 5,6,9,10 \rangle = \langle 0,\ 1,\ 4,\ 5,\ 6,\ 9,\ 10 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad$ Remove from L_i every element that is greater than t
5. return the largest element in L_n

L_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$ no more than t

P_i : the list of subset-sums from $\{x_1, x_2, \cdots, x_i\}$

Removing the elements greater than t from P_i would give us L_i.

Time complexity of algorithm: exponential in $n = |S|$

Example:

$$S = \{\ 1,\ 4,\ \ 5\ \}$$

$$P_1 = \langle 0,\ 1 \rangle$$

$$P_2 = \ P_1 \cup\ (P_1 + 4)\ =\ \langle 0,1 \rangle \cup \langle 4,5 \rangle\ =\ \langle 0,\ 1,\ 4,\ 5 \rangle$$

$$P_3 = \ P_2 \cup\ (P_2 + 5)\ =\ \langle 0,1,4,5 \rangle \cup \langle 5,6,9,10 \rangle\ =\ \langle 0,\ 1,\ 4,\ 5,\ 6,\ 9,\ 10 \rangle$$

Quiz questions:

1. What is the main idea of the above algorithm for “Subset-Sum Problem”?

2. Why does the above algorithm have exponential time complexity?

Roadmap of this lecture:

1. Understand approximation algorithms by solving the "Subset-Sum Problem".
    1.1 Define "Subset-Sum Problem".
    1.2 Define "Fully Polynomial-Time Approximation Scheme (FPTAS)".
    1.3 An exponential-time exact algorithm for "Subset-Sum Problem".
    1.4 FPTAS for "Subset-Sum Problem".
    1.5 Prove the correctness of the FPTAS.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \leq z \leq y$$

We can think of such a z as "representing" y in the new list L'.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \leq z \leq y$$

We can think of such a z as "representing" y in the new list L'.

Example: $\delta = 0.1$ $\quad L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \leq z \leq y$$

We can think of such a z as "representing" y in the new list L'.

Example: $\delta = 0.1$

$$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$$

$$L' = \langle 10,$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \leq z \leq y$$

We can think of such a z as "representing" y in the new list L'.

Example: $\delta = 0.1$

$$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$$

$$L' = \langle 10, 12,$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \le z \le y$$

We can think of such a z as "representing" y in the new list L'.

Example: $\delta = 0.1$

$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$

$L' = \langle 10, 12, 15,$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \leq z \leq y$$

We can think of such a z as "representing" y in the new list L'.

Example: $\delta = 0.1$

$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$

$L' = \langle 10, 12, 15, 20,$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \leq z \leq y$$

We can think of such a z as "representing" y in the new list L'.

Example: $\delta = 0.1$

$$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$$

$$L' = \langle 10, 12, 15, 20, 23,$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \le z \le y$$

We can think of such a z as "representing" y in the new list L'.

Example: $\delta = 0.1$

$$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$$

$$L' = \langle 10, 12, 15, 20, 23, 29 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Main idea: Trim each list L_i. (That is, remove subset-sums that are too "close" to each other.)

Specifically: let $0 < \delta < 1$.

When "trimming" a list L by δ, remove as many elements from L as possible, in such a way that if L' is the result of "trimming" L, then for every element y that was removed from L, some element z still in L' approximates y :

$$\frac{y}{1+\delta} \leq z \leq y$$

We can think of such a z as "representing" y in the new list L'.

Example: $\delta = 0.1$

$$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$$

$$L' = \langle 10, 12, 15, 20, 23, 29 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Exact-Subset-Sum(S,n,t)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. Remove from L_i every element greater than t
5. return the largest element in L_n

Fully Polynomial-Time Approximation Scheme:

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. Remove from L_i every element greater than t
6. return the largest element z^* in L_n

$$\frac{y}{1+\delta} \le z \le y$$

$$\delta = \frac{\epsilon}{2n}$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\quad t = 308$ $\quad \epsilon = 0.4$ $\quad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\quad t = 308$ $\quad \epsilon = 0.4$ $\quad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$L_0 = \langle 0 \rangle$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $L_i =$ TRIM$(L_i, \epsilon/2n)$
5. Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104, 102, 201, 101\}$ $\quad t = 308 \quad \epsilon = 0.4 \quad \frac{\epsilon}{2n} = \frac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0, 104 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\quad t = 308$ $\quad \epsilon = 0.4$ $\quad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0,\ 104 \rangle$$

$$L_2 = \langle 0,\ 102,\ 104,\ 206 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\quad t = 308$ $\quad \epsilon = 0.4$ $\quad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0,\ 104 \rangle$$

$$L_2 = \langle 0,\ 102,\ 104,\ 206 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\qquad t = 308$ $\qquad \epsilon = 0.4$ $\qquad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0,\ 104 \rangle$$

$$L_2 = \langle 0,\ 102,\ 104,\ 206 \rangle$$

$$L_3 = \langle 0,\ 102,\ 201,\ 206,\ 303,\ 407 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\qquad t = 308$ $\qquad \epsilon = 0.4$ $\qquad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0,\ 104 \rangle$$

$$L_2 = \langle 0,\ 102,\ 104,\ 206 \rangle$$

$$L_3 = \langle 0,\ 102,\ 201,\ 206,\ 303,\ 407 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\quad t = 308$ $\quad \epsilon = 0.4$ $\quad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0,\ 104 \rangle$$

$$L_2 = \langle 0,\ 102,\ 104,\ 206 \rangle$$

$$L_3 = \langle 0,\ 102,\ 201,\ 206,\ 303,\ 407 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

$L_0 = \langle 0 \rangle$
for $i = 1$ to n
    $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
    $L_i =$ TRIM$(L_i, \epsilon/2n)$
    Remove from L_i every element greater than t
return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104, 102, 201, 101\}$ $t = 308$ $\epsilon = 0.4$ $\dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0, 104 \rangle$$

$$L_2 = \langle 0, 102, 104, 206 \rangle$$

$$L_3 = \langle 0, 102, 201, 206, 303, 407 \rangle$$

$$L_4 = \langle 0, 101, 102, 201, 203, 302, 303, 404 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\qquad t = 308$ $\qquad \epsilon = 0.4$ $\qquad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0,\ 104 \rangle$$

$$L_2 = \langle 0,\ 102,\ 104,\ 206 \rangle$$

$$L_3 = \langle 0,\ 102,\ 201,\ 206,\ 303,\ 407 \rangle$$

$$L_4 = \langle 0,\ 101,\ 102,\ 201,\ 203,\ 302,\ 303,\ 404 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\quad t = 308$ $\quad \epsilon = 0.4$ $\quad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0,\ 104 \rangle$$

$$L_2 = \langle 0,\ 102,\ 104,\ 206 \rangle$$

$$L_3 = \langle 0,\ 102,\ 201,\ 206,\ 303,\ 407 \rangle$$

$$L_4 = \langle 0,\ 101,\ 102,\ 201,\ 203,\ 302,\ 303,\ 404 \rangle$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Fully Polynomial-Time Approximation Scheme:

Example: $S = \{104,\ 102,\ 201,\ 101\}$ $\qquad t = 308$ $\qquad \epsilon = 0.4$ $\qquad \dfrac{\epsilon}{2n} = \dfrac{0.4}{8} = 0.05$

$$L_0 = \langle 0 \rangle$$

$$L_1 = \langle 0,\ 104 \rangle$$

$$L_2 = \langle 0,\ 102,\ 104,\ 206 \rangle$$

$$L_3 = \langle 0,\ 102,\ 201,\ 206,\ 303,\ 407 \rangle$$

$$L_4 = \langle 0,\ 101,\ 102,\ 201,\ 203,\ \underset{z^*}{302},\ 303,\ 404 \rangle$$

optimal answer: $307 = 104 + 102 + 101$

Quiz questions:

1. How does the above algorithm reduce its time complexity (compared to the previous exponential-time exact algorithm)?

2. Can you think of an instance for which the above algorithm outputs an optimal solution, and an instance for which it does not?

Roadmap of this lecture:

1. Understand approximation algorithms by solving the "Subset-Sum Problem".
   - 1.1 Define "Subset-Sum Problem".
   - 1.2 Define "Fully Polynomial-Time Approximation Scheme (FPTAS)".
   - 1.3 An exponential-time exact algorithm for "Subset-Sum Problem".
   - 1.4 FPTAS for "Subset-Sum Problem".
   - 1.5 Prove the correctness of the FPTAS.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

Claim: for every element $y \in P_i$ that is at most t, there exists an element $z \in L_i$ such that

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

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

Claim: for every element $y \in P_i$ that is at most t, there exists an element $z \in L_i$ such that

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

Proof: By induction.

If $i = 1$, thre exists $z \in L_1$ such that $\dfrac{y}{1+\epsilon/2n} \leq z \leq y$ by the definition of "trimming".

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

Claim: for every element $y \in P_i$ that is at most t, there exists an element $z \in L_i$ such that

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

Proof: Consider $i > 1$.

Case 1: $y \in P_{i-1}$ By induction, there exists $z' \in L_{i-1}$ s.t. $\dfrac{y}{(1+\epsilon/2n)^{i-1}} \leq z' \leq y$.

z' is in L_i before "trimming" it. So after "trimming" L_i, there exists $z \in L_i$ s.t. $\dfrac{z'}{1+\epsilon/2n} \leq z \leq z'$

By combining them, we get the conclusion.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

Claim: for every element $y \in P_i$ that is at most t, there exists an element $z \in L_i$ such that

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

Proof: Consider $i > 1$.

Case 2: $y = \hat{y} + x_i$, where $\hat{y} \in P_{i-1}$. By induction, there exists $z' \in L_{i-1}$ s.t. $\dfrac{\hat{y}}{(1+\epsilon/2n)^{i-1}} \leq z' \leq \hat{y}$.

$z' + x_i$ is in L_i before "trimming" it. So after "trimming" L_i, there exists $z \in L_i$ s.t. $\dfrac{z'+x_i}{1+\epsilon/2n} \leq z \leq z' + x_i$

By combining them, we get the conclusion.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

Claim: for every element $y \in P_i$ that is at most t, there exists an element $z \in L_i$ such that

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

There exists $z \in L_n$ such that

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

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

There exists $z \in L_n$ such that $\dfrac{y^*}{(1+\epsilon/2n)^n} \leq z \leq y^*$

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

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $L_i =$ TRIM$(L_i, \epsilon/2n)$
5. Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

There exists $z \in L_n$ such that $\dfrac{y^*}{(1+\epsilon/2n)^n} \leq z \leq y^*$

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

Since $z \leq z^* \leq y^*$, we have $\dfrac{y^*}{z^*} \leq \left(1 + \dfrac{\epsilon}{2n}\right)^n$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

There exists $z \in L_n$ such that $\dfrac{y^*}{(1+\epsilon/2n)^n} \leq z \leq y^*$

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

Since $z \leq z^* \leq y^*$, we have $\dfrac{y^*}{z^*} \leq \left(1 + \dfrac{\epsilon}{2n}\right)^n \leq (e^{\epsilon/2n})^n = e^{\epsilon/2} \leq 1 + \dfrac{\epsilon}{2} + \left(\dfrac{\epsilon}{2}\right)^2 \leq 1 + \epsilon$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Let $y^* \in P_n$ be the optimal solution.

$z^* \leq y^* \leq t$

There exists $z \in L_n$ such that $\dfrac{y^*}{(1+\epsilon/2n)^n} \leq z \leq y^*$

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

Since $z \leq z^* \leq y^*$, we have $\dfrac{y^*}{z^*} \leq \left(1 + \dfrac{\epsilon}{2n}\right)^n \leq (e^{\epsilon/2n})^n = e^{\epsilon/2} \leq 1 + \dfrac{\epsilon}{2} + \left(\dfrac{\epsilon}{2}\right)^2 \leq 1 + \epsilon$

Approximation ratio $\leq 1 + \epsilon$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Assume $L_i = \langle 0, z_1, z_2, \cdots, z_k \rangle$.

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Assume $L_i = \langle 0, z_1, z_2, \cdots, z_k \rangle$.

$z_1 \geq 1$

$$z_2 > z_1(1 + \frac{\epsilon}{2n}) \geq (1 + \frac{\epsilon}{2n})$$

$$z_3 > z_2(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^2$$

$\cdots$

$$z_k > z_{k-1}(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^{k-1}$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Assume $L_i = \langle 0, z_1, z_2, \cdots, z_k \rangle$.

$$z_1 \geq 1$$

$$z_2 > z_1(1 + \frac{\epsilon}{2n}) \geq (1 + \frac{\epsilon}{2n})$$

$$z_3 > z_2(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^2$$

…

$$z_k > z_{k-1}(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^{k-1}$$

$z_k \leq t$, so $k - 1 \leq \lfloor \log_{1+\epsilon/2n} t \rfloor$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers. A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Assume $L_i = \langle 0, z_1, z_2, \cdots, z_k \rangle$.

$z_1 \geq 1$

$z_2 > z_1(1 + \frac{\epsilon}{2n}) \geq (1 + \frac{\epsilon}{2n})$

$z_3 > z_2(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^2$

…

$z_k > z_{k-1}(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^{k-1}$

$z_k \leq t$, so $k - 1 \leq \lfloor \log_{1+\epsilon/2n} t \rfloor$

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

$$\leq \frac{2n(1 + \epsilon/2n)\ln t}{\epsilon} + 2 < \frac{3n \ln t}{\epsilon} + 2$$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t .

Output: A subset of S whose sum is as large as possible but no larger than t .

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $\quad L_i =$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
4. $\quad L_i =$ TRIM$(L_i, \epsilon/2n)$
5. $\quad$ Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Assume $L_i = \langle 0, z_1, z_2, \cdots, z_k \rangle$.

$z_1 \geq 1$

$z_2 > z_1(1 + \frac{\epsilon}{2n}) \geq (1 + \frac{\epsilon}{2n})$

$z_3 > z_2(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^2$

…

$z_k > z_{k-1}(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^{k-1}$

$z_k \leq t$, so $k - 1 \leq \lfloor \log_{1+\epsilon/2n} t \rfloor$

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

$$\leq \frac{2n(1+\epsilon/2n)\ln t}{\epsilon} + 2 < \frac{3n \ln t}{\epsilon} + 2$$

Time complexity of algorithm: $O(\frac{n^2 \ln t}{\epsilon})$

Subset-Sum Problem

Input: A set $S = \{x_1, x_2, \cdots, x_n\}$ of n positive integers.
A positive integer t.

Output: A subset of S whose sum is as large as possible but no larger than t.

Approx-Subset-Sum (S, n, t, ϵ)

1. $L_0 = \langle 0 \rangle$
2. for $i = 1$ to n
3. $L_i =$ MERGE-LISTS$(L_{i-1},\ L_{i-1} + x_i)$
4. $L_i =$ TRIM$(L_i,\ \epsilon/2n)$
5. Remove from L_i every element greater than t
6. return the largest element z^* in L_n

Theorem: The algorithm is a fully polynomial-time approximation scheme for the subset-sum problem.

Proof: Assume $L_i = \langle 0,\ z_1,\ z_2,\ \cdots,\ z_k \rangle$.

$z_1 \geq 1$

$z_2 > z_1(1 + \frac{\epsilon}{2n}) \geq (1 + \frac{\epsilon}{2n})$

$z_3 > z_2(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^2$

...

$z_k > z_{k-1}(1 + \frac{\epsilon}{2n}) > (1 + \frac{\epsilon}{2n})^{k-1}$

$z_k \leq t$, so $k - 1 \leq \lfloor \log_{1+\epsilon/2n} t \rfloor$

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

$$\leq \frac{2n(1 + \epsilon/2n)\ln t}{\epsilon} \ + \ 2 \ < \ \frac{3n \ln t}{\epsilon} \ + \ 2$$

Time complexity of algorithm: $O(\frac{n^2 \ln t}{\epsilon})$

polynomial in input size (n and $\ln t$), and in $\frac{1}{\epsilon}$

Quiz question:

1. For the above FPTAS, how does its approximation ratio and its time complexity change as ϵ decreases?