Lecture 18:

One more reduction

Approximation Algorithms
Exact Cover

Input
Finite set \( X \), \( |X| = m \)
Subsets \( S_1, \ldots, S_n \) \( \subseteq \) each \( S_i \subseteq X \)

Question
is there \( S' \subseteq S \) such that
each elt of \( X \) is in exactly one \( S_i \in S' \)?

e.g.
\( X = \{0, 1, 2, 3\} \)

\( S_1' = \{1, 2, 3\}, \{2, 3\}, \{1, 3\}, \{0, 2, 3\} \) \( \text{NO} \)

\( S_2' = \{1, 2, 3\}, \{2, 3\}, \{1, 3\}, \{0, 3\} \) \( \text{YES} \)
**Subset Sum**

**Input** \( w_1, \ldots, w_n \) \( \geq \) integers \( B \)

**Question** is there \( S \subseteq \{1, \ldots, n\} \)
\[ \text{s.t. } \sum_{i \in S} w_i = B \] ?

*e.g.* \( 1, 5, 9, 3, 27 \)
\( B = 31 \)
\( S = \{1, 4, 5\} \) \( \text{i.e., } 1 + 4 + 27 = 31 \)

**Exact Cover \( \leq \) Subset Sum:**

**Diagram:**

\[ S_1, \ldots, S_m \]
\[ R \]
\[ \begin{array}{c}
\frac{w_1 \ldots w_n}{B} \\
\text{Subset} \\
\text{Sum}
\end{array} \]

pick a large integer \( b > n \) \( \text{w.l.o.g. assume these are ints} \)

For each \( S_i \),

\[ w_i = \sum_{x \in S_i} p^x \]

\[ B = \sum_{x=0}^{n-1} p^x = \frac{p^n - 1}{p - 1} \]

\[ \text{represent } S_i \text{ in p-ary notation} \]

**Note:** The diagram and additional text are hand-drawn and may not be perfectly clear. The objective is to convey the concepts of Subset Sum and Exact Cover within a mathematical context.
e.g. \( X = \{0,1,2,3\} \)

\[
B \geq 1000^0 + 1000^1 + 1000^2 + 1000^3 = \sum B
\]

\[
= 1,001,001,001
\]

\[
S_1 = \{1,2,3\} \Rightarrow \quad 1000^1 + 1000^2 + 1000^3 = \sum_{S_1} = W_1
\]

\[
= 1,001,001,000
\]

\[
S_2 = \{2,3\} \Rightarrow \quad 1000^1 = \sum_{S_2} = W_2
\]

\[
= 1,001,000,000
\]

\[
S_3 = \{1,3\} \Rightarrow \quad 1000^0 = \sum_{S_3} = W_3
\]

\[
= 1,000
\]

\[
S_4 = \{0,3\} \Rightarrow \quad 1000^0 = \sum_{S_4} = W_4
\]

\[
= 1
\]

**Note:** \( W_1 + W_3 = 2 \cdot 1000^1 + 1000^2 + 1000^3 \)

Since \( S_1 \cap S_3 \neq \emptyset \)

**Fact:** Adding \( W_i \)'s corresponds to "union" of \( S_i \)'s

Coefficients of sums of \( W_i \)'s will remain 0/1 exactly when corresponding \( S_i \)'s are non-intersecting

\( p \) large enough so that no carries

\( \Rightarrow \) coeff of all \( p \times \) terms are 1 only when \( S_i \)'s cover \( X \)

**Claim:** There is exact cover of \( X \) by \( S \)

iff

\[
\text{There is } S \subseteq S \text{ s.t. } \sum_{i \in S} W_i = B
\]

Also note: Reduction is poly time!
Lecture 18:

Approximation Algorithms

We saw a large class of problems for which we know no efficient way of solving them. What can we do? Exponential time algorithms, heuristics...

Find a "pretty good" solution instead of the "best" solution

Specify Optimization problem:
- Specify set of valid outputs (solutions) for each input
- Cost function $c(y)$ for each output
- Minimize or maximize?

Optimal Algorithm:

$\forall$ inputs $x$, outputs $OPT$ s.t. $c(OPT) = \min \{ c(x) : \text{all valid outputs} \}$
α-approximation algorithm: \((\alpha > 1)\) can depend on \(n\)

\[
\forall \text{ inputs, outputs APX st.} \\
\frac{C(\text{APX})}{C(\text{OPT})} \leq \alpha \\
\frac{C(\text{OPT})}{C(\text{APX})} \leq \alpha
\]

(minimization)

(maximization)

Strange thing:

Some NP-complete hard to approx, others easy

- e.g. Clique \(\alpha = \frac{n}{\log n}\) but not \(\alpha = n^e\) for \(0 < e < 1\)
- Vertex Cover \(\alpha = 2\)
- Set cover \(\alpha = \ln n\) but not better
Max-clique

Input: \( G, k \)

Problem: Find maximum size \( C \subseteq V \) s.t. \( C \) is a clique.

Recall: Decision problem is \( NP \)-complete.

Approximating Max Clique:

- \( \alpha = n \) is trivial.

A better algorithm:

- Divide vertices into \( K = O(\frac{n}{\log n}) \) blocks \( B_1, \ldots, B_k \) each of \( O(\log n) \) nodes.
- For each block, check all possible subsets of nodes to see if it is a clique.
- Output largest found clique.

Complexity: \( O(\frac{n}{\log n} \cdot n \cdot \log^2 n) = O(n^2 \log n) \)

- \#blocks
- \#subsets in each block
- Note each subset is of size \( O(\log n) \)
- \( \uparrow \) time to check if clique
Approximation Ratio:

\[ C^* \text{ is largest clique} \]

pigeon hole principle \( \Rightarrow \) \( \exists \) block \( B \) with \( n \leq \frac{|C^*|}{K} \) nodes of \( C^* \)

when algorithm considers \( B \), it will find

clique of size \( \frac{|C^*|}{K} \)

\[ \therefore \text{approximation ratio } = \frac{|C^*|}{\frac{|C^*|}{K}} = K = O(n/\log n) \]

Can we do better?

Then \( \text{NP-hard to approx clique to w/in } \alpha = n^\varepsilon \)

for any \( 0 < \varepsilon < 1 \)

\( \Rightarrow \) hardness of \( \text{NP} \)-approximation algorithm
**Set Cover**

**Input**
- set $V$ of size $m$
- family $F$ of $n$ subsets of $U$
  - i.e. $F = \mathcal{F}_1 \ldots \mathcal{F}_n$

**Problem**
- find min size $C \subseteq F$ which covers $U$
  - i.e. $U = \bigcup_{i=1}^{\mathcal{F}_i}$

**Example**

NP-hard!
Approximating Set Cover: (Greedy)

Initialize \( C = \emptyset \)
\[ W = U \]
While \( W \neq \emptyset \) do
   pick set \( S \) maximizing \( |S \cap W| \)
   \[ C = C \cup S \]
   \[ W = W - S \]
Return \( C \)

Approximation Ratio:
Thm. algorithm gives \( O(\ln m) \)-approximation

Proof.
Bound # rounds "while"-loop \( \Rightarrow \) bound size of \( C \)

\{ Compare to Optimal cover, but we don't know it! \}

Assume size of Opt cover = \( k \), we don't even know this!

Easy Fact: at any iteration, some set in Opt cover will cover at least \( \frac{1}{k} \) fraction of \( W \)

Why? Opt cover has size \( \leq k \) \( \Rightarrow \) covers \( W \leq U \)

Some set has to cover \( \geq \frac{|W|}{k} \) elts

Fact implies after \( i \) rounds of "while loop"

at most \( |W| \cdot \left(1 - \frac{1}{k}\right)^i \) points left.

+ After \( T = O(K \ln |U|) \) rounds, \( \leq |W| \cdot \left(1 - \frac{1}{k}\right)^{ck \ln |U|} \leq 1 \) points left, so done.