Algorithm Design Techniques

- Induction
- Reduction

Self-Reduction

Reduction: $A \propto B$

Some reductions we've seen
- Sorting $\propto$ Find-max
- General Selection $\propto$ Find-median
**Inductive Design to solve A**

- One Stage
  - Self-Reduction

- Two Stage
  - Define B
  - Reduce A to B
  - Solve B using self-reduction

**Strengthening the inductive hypothesis**

---

**Dominating Set**

- A dominating set of a graph G is a subset W of the vertices of G such that every vertex in G is either in W or adjacent to a vertex in W.

**Examples**

---

**Dominating Set in Trees with n or fewer nodes**

*Doesn’t seem to work*

---

**Dominating Set in Trees with n or fewer nodes: Define new problem**

*Is self-reducible*
What do you want to know about the subtrees of node v?

Caveat: You have to reproduce that info for the subtree rooted at v.

Visualize a solution ...

1. v is in the DS
2. v is not in the DS
   A) a child of v is in the DS
   B) the parent of v is in the DS

What do you want to know about a child w?

1. Smallest dominating set that includes w.
2. Smallest dominating set that does not include w.
3. Smallest dominating set on the subtrees rooted at the children of w. (Note: w need not be covered.)

Definitions

1. I(w): Smallest dominating set of the subtree rooted at w that includes w.
2. E(w): Smallest dominating set of the subtree rooted at w that does not include w.
3. C(w): Smallest dominating set on the subtrees rooted at the children of w. (Note: w need not be covered.)

Caveat

Compute I(v), E(v) and C(v) if we have I(w), E(w) and C(w) for each child w of v?

I(v)=
E(v)=
C(v)=

Base Case

v is a leaf:
I(v)=
E(v)=
C(v)=
Dominating Set in Trees with \( n \) or fewer nodes

\[ T \xrightarrow{\text{transform}} T_r \xrightarrow{\text{Compute LRC}} I(r) \xrightarrow{\text{transform}} W \]

**Example**

DS-tree algorithm

- Is it correct?
- Is it efficient?

Longest Increasing Subsequence

- Input: Sequence of integers \( X: x_1, x_2, \ldots, x_n \)
- Output: Longest increasing subsequence of \( X \); i.e., a subsequence \( Z: z_1, z_2, \ldots, z_k \) such that \( z_i < z_{i+1} \) for each \( i:1..k-1 \).

Example

- 1, -3, 2, 10, 8, 23, -2, 17, 5

LIS\(_{n+1}\) \( \propto \) LIS\(_n\)

Don't know how to do it!!!
To solve A

• Define B (Strengthen the inductive hypothesis)
• Reduce A to B
• Solve B using self-reduction

LIS and Modified LIS

• Input: Sequence of integers X: \( x_1, x_2, \ldots, x_n \)
• Output: Longest increasing subsequence

• Input: Sequence of integers X: \( x_1, x_2, \ldots, x_n \)
• Output: For each \( i:1 \ldots n \), a longest increasing subsequence of \( x_1, \ldots, x_i \) that ends in \( x_i \)

MLIS\((x_1, \ldots, x_n)\)

\[
MLIS(x_1, \ldots, x_n) = \\
1. \text{LIS of } x_1 \text{ that ends in } x_1 \\
2. \text{LIS of } x_1, x_2 \text{ that ends in } x_2 \\
\vdots \\
n-1. \text{LIS of } x_1, \ldots, x_{n-1} \text{ that ends in } x_{n-1} \\
n. \text{LIS of } x_1, \ldots, x_n \text{ that ends in } x_n
\]

Example

• 1, -3, 2, 10, 8, 23, -2, 17, 5

To solve A

• Define B
• Reduce A to B
• Solve B using self-reduction

LIS \( \preceq \) MLIS

\[ X \xrightarrow{\text{Transform}} \text{Algorithm for MLIS} \xrightarrow{\text{Transform}} \text{MLIS}(?) \xrightarrow{\text{Transform}} \text{LIS}(X) \]
**LIS ∝ MLIS**

**Algorithm for LIS**

- Choose longest subsequence

**Algorithm for MLIS**

1. LIS of $x_1$ that ends in $x_1$
2. LIS of $x_1, x_2$ that ends in $x_2$
   ...
$n-1. \text{LIS of } x_1, \ldots, x_{n-1} \text{ that ends in } x_{n-1}$
$n. \text{LIS of } x_1, \ldots, x_n \text{ that ends in } x_n$

**To solve A**

- Define B
- Reduce A to B
- Solve B using self-reduction

**MLIS Self-Reduction**

**MLIS($x_1, \ldots, x_n$)**

1. LIS of $x_1$ that ends in $x_1$
2. LIS of $x_1, x_2$ that ends in $x_2$
   ...
$n-1. \text{LIS of } x_1, \ldots, x_{n-1} \text{ that ends in } x_{n-1}$
$n. \text{LIS of } x_1, \ldots, x_n \text{ that ends in } x_n$

**MLIS($x_1, \ldots, x_n$)**

1. LIS of $x_1$ that ends in $x_1$
2. LIS of $x_1, x_2$ that ends in $x_2$
   ...
$n-1. \text{LIS of } x_1, \ldots, x_{n-1} \text{ that ends in } x_{n-1}$
$n. \text{LIS of } x_1, \ldots, x_n \text{ that ends in } x_n$
Construct MLIS($x_1, \ldots, x_n$)

$\text{MLIS}(x_1, \ldots, x_n) =$
1) $\text{MLIS}(x_1, \ldots, x_{n-1})$ plus
2) Choose longest $\text{LIS}(x_1, \ldots, x_j)$ ending in $x_j$ ($j<n$) such that $x_j < x_n$. Append $x_n$.

LIS algorithm

- Is it correct?
- Is it efficient?

Recap: To solve A

- Define B
- Reduce A to B
- Solve B using self-reduction

Grocery Bags

How should we pack $n$ items weighing $w_1, w_2, \ldots, w_n$ ($w_i \leq W$) in two bags so as to minimize the difference in the weights of the bags?

Or even simpler: What is the smallest possible weight difference?

Self-Reduction

I don’t know how to make this work!

Self-Reduction

Strengthen the induction hypothesis
Problem B

- Input: Weights $w_1, w_2, \ldots, w_n$
- Output: A binary vector $T$:
  - $T[i] = 1$ if some subset of the weights sum to $i$
  - $T[i] = 0$ otherwise
  - for $i=0, \ldots, nW$

Transform

$\begin{align*}
T_0 & \quad T_1 & \quad T_2 & \quad T_3 & \quad \ldots & \quad T_{(n-1)W} \\
\downarrow & & & & & \\
T_{0} & \quad T_{1} & \quad T_{2} & \quad T_{3} & \quad \ldots & \quad T_{nW}
\end{align*}$

Self-Reduction: Problem B

What are the base cases?

Reduction: $A \propto B$

- Is it correct?
- Is it efficient?

Grocery Bag algorithm
Algorithm A

Use Algorithm B to compute $t[0]...t[nW]$
Let $S = \sum w_i$
(Note: $t[0..S]$ is symmetric about $S/2$)
Let $j$ be the closest index to $S/2$ such that $t[j] = 1$
Return $|j - S/2|$

Grocery Bags

How should we pack $n$ items weighing $w_1, w_2, ..., w_n$ ($w_i \leq W$) in two bags so as to minimize the difference in the weights of the bags?

Or even simpler: What is the smallest possible weight difference?