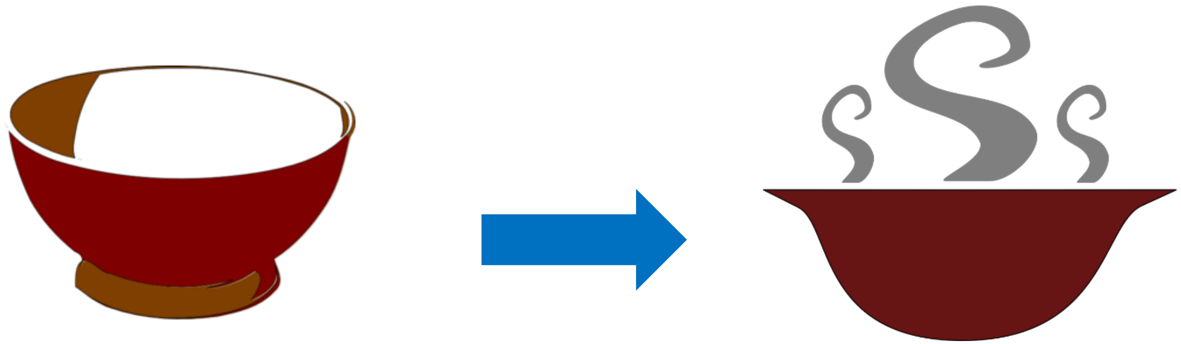
CSE 421: Introduction to Algorithms

Algorithm Design by Reduction/Induction

Shayan Oveis Gharan

Boiling Water Example

Q: Given an empty bowl, how do you make boiling water?



A: Well, I fill it with water, turn on the stove, leave the bowl on the stove for 20 minutes. I have my boiling water.

Q: Now, suppose you have a bowl of water, how do you make boiling water?

A: First, I pour water away, now
I have an empty bowl and
I have already solved this!

Lesson: Never solve a problem twice!

Using Induction in Algorithm Design

We can design an ALG by induction

We just need to guarantee:

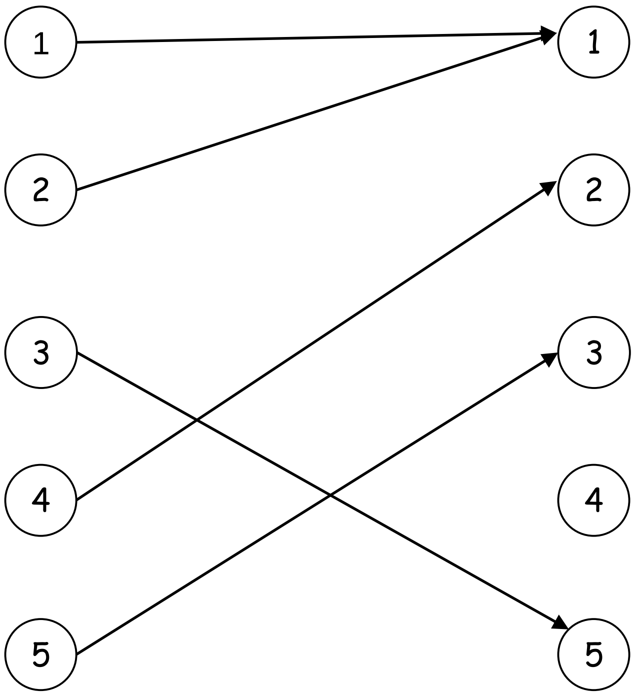
- (Base Case) It is possible to solve a small instance of problem
- (Induction Step) A solution to every instance can be constructed from solutions of smaller instances

Finding Maximum One-to-One Mapping

Problem: Given a function f that maps A to A.

Goal: Find a maximum set $S \subseteq A$ s.t.,

- f maps S to itself
- f is one-to-one on S, i.e., for all $i, j \in S$, $f(i) \neq f(j)$.



Properties of Optimum S

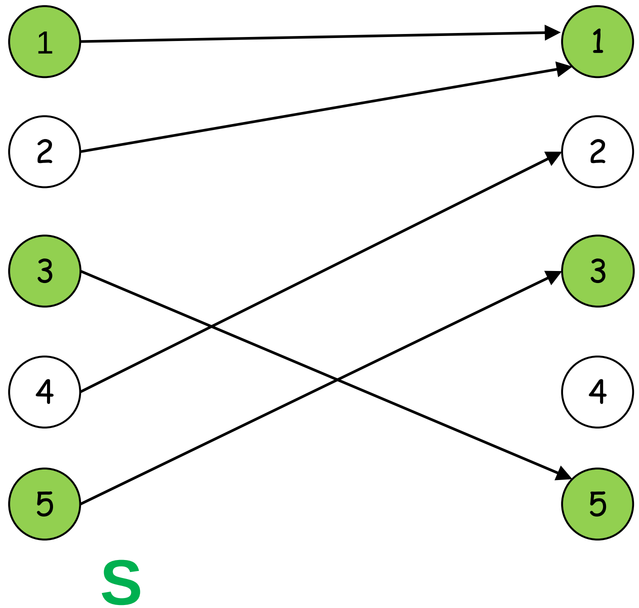
Claim: f defines a bijection on S, i.e., it is a perfect matching between elements of S.

Pf. Since of S one-to-one on S,
enough to show it is surjective.

Since f is one-to-one it maps S
to |S| different values

But f maps S to S.

So, it is surjective



Design by Induction

Problem: Given a function f that maps A to A. Find a maximum set $S \subseteq A$ s.t.,

- f maps S to itself
- f is one-to-one on S, i.e., for all $i, j \in S$, $f(i) \neq f(j)$.

If $|A| = 1$, then we let $S = A$.

Otherwise, suppose we can solve for sets B of size $|B| < n$.

How to reduce A to B?

- Idea 1: Find an element $i \in S$, Recursively find optimum S for $B = A - \{i\}$ and output $\{i\} \cup S$.
- Idea 2: Find an element $i \notin S$. Output the optimum S of $B = A - \{i\}$.

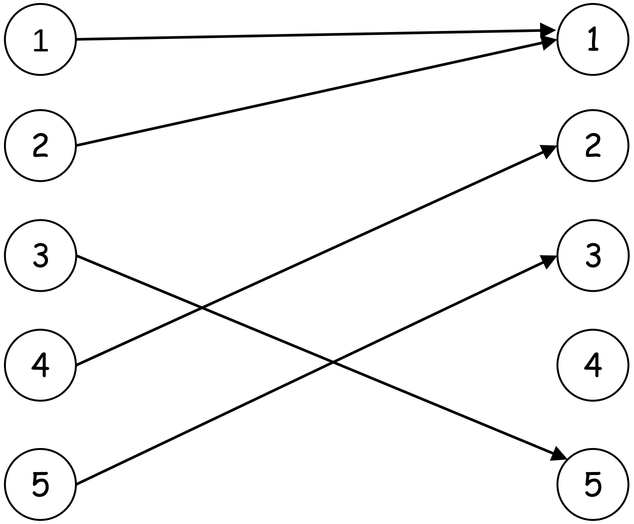
Design by Induction

Q: Can you find an element i that provably in S?

Q: Can you find an element i that provably not in S?
A: Yes. 4

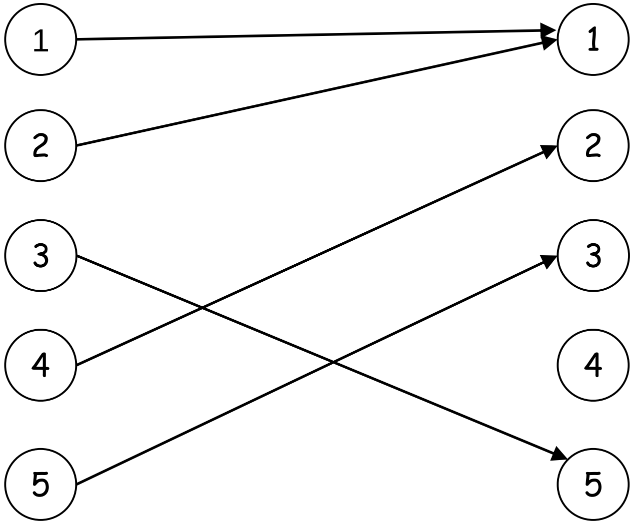
Claim: If $f(i) \neq j$ for all $i \in A$, then
$j \notin S$

Pf. (by contradiction)
If $j \in S$, then S is not a bijection on S.
Contradiction



Inductive Algorithm

```
Initialize A
while (there is an j in A s.t., f(i)<>j for all i in A) {
    Remove j from A
}
Output A.
```



Pf of Correctness

Prove by induction

Base Case: If $|A| = 1$ then $S = A$ because

- f is one-to-one because $|S| = |A| = 1$
- f maps S to S because f maps A to A.

Induction Hyothesis: Given $|B| < n$ and f that maps B to B, ALG finds maximum set S s.t., f maps S to S and f is one-to-one on S.

Induction Step: Given |A|=n. If there is $j \in S$ s.t., $f(i) \neq j$ for all $i \in A$. Then, $j \notin S$.

So, the optimum is the optimum S of $B = A - \{j\}$. ← By Claim

Otherwise, f is one-to-one on A.

Ind Hyp Check: Since $f(i) \neq j$, f maps B to B.

Maximum Consecutive Subsequence

Problem: Given a sequence $x_1, \ldots, x_n$ of integers (not necessarily positive),

Goal: Find a subsequence of consecutive elements s.t., the sum of its numbers is maximum.

1 -3 [7 -2 -3 8] -10 1 -7

Applications: Figuring out the highest interest rate period in stock market

Brute Force Approach

Try all consecutive subsequences of the input sequence.

There are $\binom{n}{2} = \Theta(n^2)$ such sequences.

We can compute the sum of numbers in each such sequence in $O(n)$ steps.

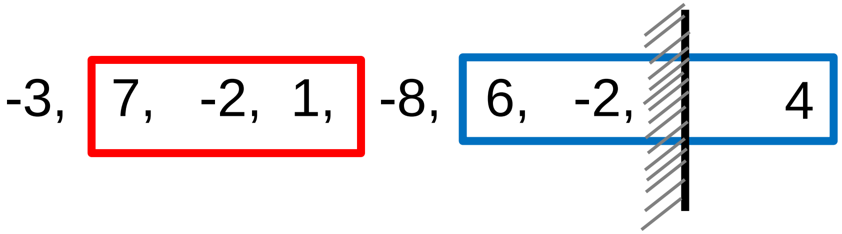
So, the ALG runs in $O(n^3)$.

With a clever loop we can do this in $O\left(n^2\right)$.
But, can we solve in linear time?

First Attempt (Induction)

Suppose we can find the maximum-sum subsequence of $x_1, \ldots, x_{n-1}$. Say it is $x_i, \ldots, x_j$

- If $x_n < 0$ then it does not belong to the largest subsequence. So, we can output $x_i, \ldots, x_j$
- Suppose $x_n > 0$.
  - If $j = n - 1$ then $x_i, \ldots, x_n$ is the maximum-sum subsequence.
  - If $j < n - 1$ there are two possibilities
    1) $x_i, \ldots, x_j$ is still the maximum-sum subsequence
    2) A sequence $x_k, \ldots, x_n$ is the maximum-sum subseqence



Second Attempt (Strengthing Ind Hyp)

Stronger Ind Hypothesis: Given $x_1, \ldots, x_{n-1}$ we can compute the maximum-sum subsequence, and the maximum-sum suffix subsequence.

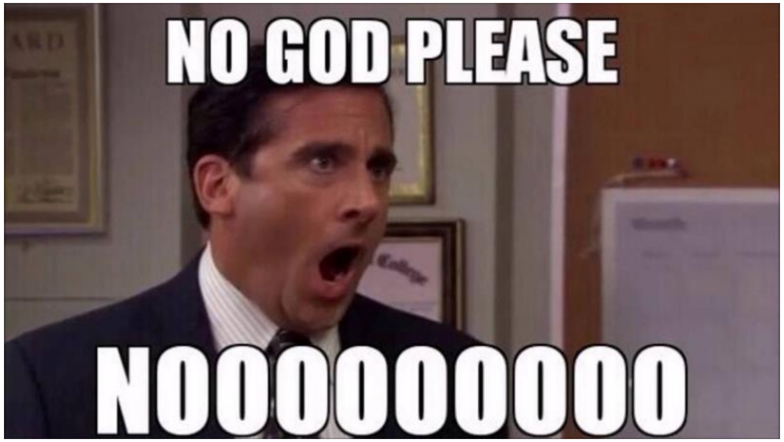
-3, [7, -2, 1,] -8, [6, -2]

x_i x_j x_k x_{n-1}

Say $\boldsymbol{x_i, \ldots, x_j}$ is the maximum-sum and $x_k, \ldots, x_{n-1}$ is the maximum-sum suffix subsequences.

- If $x_k + \cdots + x_{n-1} + x_n > x_i + \cdots + x_j$ then $x_k, \ldots, x_n$ will be the new maximum-sum suffix subsequence

Are we done?



Updating Max Suffix Subsequence

-3, 7, -2, 1, -8, 6, -2, | 4

Say $x_k, \dots, x_{n-1}$ is the maximum-sum suffix subsequences of $x_1, \dots, x_{n-1}$.

- If $x_k + \cdots + x_n \geq 0$ then,
  $x_k, \dots, x_n$ is the new maximum-sum suffix subsequence

- Otherwise,
  The new maximum-sum suffix is the empty string.

Maximum Sum Subsequence ALG

```
Initialize S=0 (Sum of numbers in Maximum Subseq)
Initialize U=0 (Sum of numbers in Maximum Suffix)
for  (i=1 to n) {
    if (x[i] + U > S)
        S = x[i] + U

    if (x[i] + U > 0)
        U = x[i] + U
    else
        U = 0
}
Output S.
```

Pf of Correctness

Suppose $x_i, \dots, x_j$ is the optimum subsequence of $x_1, \dots, x_n$
We show that we correctly find this sequence assuming we are given maximum-sum subseq and maximum-sum suffix subseq of $x_1, \dots, x_{n-1}$.

- Case 1 ($j = n$): In this case $x_i, \dots, x_{n-1}$ must be the maximum-sum suffix subseq of $x_1, \dots, x_{n-1}$.
  - Because if there is a larger suffix subseq $x_k, \dots., x_{n-1}$ the OPT is larger which is a contradiction

- Case 2 ($j < n$): In this case $x_i, \dots, x_j$ is already the maximum-sum subseq of $x_1, \dots, x_{n-1}$ and we are done by induction hypothesis

In both cases we have all the information to compute OPT.

Is that it? No, we need to show we compute the max suffix subseq correctly

Summary

- Try to reduce an instance of size n to smaller instances
  - Never solve a problem twice

- Before designing an algorithm study properties of optimum solution

- If ordinary induction fails, you may need to strengthen the induction hypothesis