# Greedy Method

**Context** In general, when you try to solve a problem, you are trying to find a solution from among a large space of possibilities. You usually do this by making a series of decisions, what move to make at each step (for example: send 2 cannibals across first, or 1 cannibal and 1 missionary, etc.).

If you have no information or no way to tell which choice is best, then you may have to do an exhaustive search over all the possibilities using backtracking. Basically this means you pick some choice and go down that path until you either reach a solution or hit a dead end, in which case you go back (undo) and try something else. While this will work in principle, in practice it's often unusable for realistic problems because there are just too many possibilities to explore. Even relatively simple-seeming problems may have billions upon billions of possible solutions, and it would take infeasibly long time to check them all.

**Greedy Algorithm** In some cases, the problem has some underlying structure that lets you be more intelligent, and you can figure out the right choice at each step, without ever needing to undo or backtrack a decision. In the happiest cases, when you're especially lucky, there is sufficient structure that you can quickly reach a solution by just picking the straightforward "best" choice at each step. This is called greedy method.

The bad news is that greedy method doesn't always work! When it does work it's great, but for many problems, when you pick what looks like the best choice for part of the solution, you can later find that it actually leads you down the wrong path and forces bad choices for other parts of the solution. So before you apply greedy method, it's important to prove that it actually will find the best solution for the problem you're trying to solve.

Another point to mention here is that even for a single problem, there may be more than one potential greedy strategy, more than one way to determine what looks like the "best" choice at each step. And it could be that greedy method works using one strategy but not using another strategy. So you need to prove that it works for the particular strategy you're using.

**Other Approaches** In cases where greedy method doesn't work, you still may be able to use other approaches that are much better than doing an exhaustive search, as long as there is some structure to the problem. One such approach is dynamic programming, which will be covered in the coming weeks. Incidentally, in a way you can think of greedy method as a simple special case of Dynamic Programming.

## 1 Homework

For each of the following problems, write down a greedy strategy for solving the problem. Think about the correctness of your strategy. If you believe that your strategy is not optimal, construct a counterexample, revise your strategy, and repeat.

On the other hand, if you are convinced of the correctness of your strategy, write a convincing argument for its correctness.

**Problem 1: Coin Change**

Suppose you are working at a bank, and a client asks you for 89¢ in quarters, dimes, nickels, and pennies. The clients wallet is small, so you should exchange as few coins as possible. What is the fewest number of coins you can exchange? If another client comes in asking for $n$ cents in coins, how would you pick which coins to exchange with the client?

**Problem 2: Minimum Cost Sum**

You are given a sequence $a_1, a_2, \ldots, a_n$ of nonnegative integers, where $n \geq 1$. You are allowed to take any two numbers and add them to produce their sum. However, each such addition has a cost which is equal to the sum. The goal is to the find the sum of all the numbers in the sequence with minimum total cost. Describe an algorithm for finding the sum of the numbers in the sequence with minimum total cost. Argue the correctness of your algorithm.

# 2 Solved Problems

**Problem 3: Averaging Down**

There are 10 identical vessels, one of them with 100 pints of water and the others empty. You are allowed to perform the following operation: take two of the vessels and split the total amount of water in them equally between them. The object is to achieve a minimum amount of water in the original vessel (the one containing all the water in the initial setup) by a sequence of such operations. What is the best way to do this? You can do as many operations as you want — the goal is just to minimize the amount of water in the original vessel.

**Solution: Averaging Down**

A greedy algorithm to solve this problem is simple, and is described below. A greedy algorithm can only be applied if there is a proof that it works, though, and the proof in this case is more complicated than the algorithm itself, which is not uncommon. One valid proof follows the algorithm below.

**Algorithm** The algorithm is as follows: on each step, split the original vessel with one of the empty vessels, which has the effect of halving the amount of water in the original vessel. When there are no empty vessels left, you are done. At this point the original vessel, which originally had 100 pints of water, has been halved 9 times, and now contains $100 * (1/2)^9 \approx 0.195$ pints of water. This is the minimum possible.

This is greedy because on each step you are choosing the action that makes the maximum possible progress towards the goal (reducing the water in the orginal vessel the most) on that step.

**Proof** For the proof, we'll keep track of the minimum amount of water in any vessel that has water. Call this quantity $m$. For example, initially $m = 100$, since there is only one vessel with water and it has 100 pints. If you split the original vessel with an empty one, then they each have 50 pints, so then $m = 50$. And if you then split one of those with another empty vessel, then $m = 25$, etc.

Note that no matter what sequence of actions is taken, even if we aren't using the greedy algorithm above, the only way $m$ can ever decrease is if some vessel (not necessarily the one with $m$ pints in it) is split with an empty vessel. This is because if you split two non-empty vessels, they each have at least $m$ pints beforehand (since $m$ is the amount in the smallest non-empty vessel), and so they'll each have at least $m$ pints after the split as well.

Now, consider any sequence of steps (not necessarily following the greedy strategy above), and let $i$ be the number of steps in that sequence which split some non-empty vessel with an empty vessel. The sequence may also include steps that split two non-empty vessels, but those aren't included in $i$. We will prove the following claim:

**Claim** At the end of such a sequence of steps, it must be the case that $m \geq 100/2^i$, in other words, $m$ divided by 2, $i$ times.

**Proof of Claim** The proof is by induction on $i$[1]. Consider the case where $i = 1$, which means we've split exactly one non-empty vessel with an empty one. This is obviously just the first move, where the original vessel with 100 pints is split with an empty vessel, after which $m = 50$, which is indeed $100/2^i = 100/2^1 = 50$, as claimed. And clearly there's no way to reduce $m$ futher without involving one of the empty vessels, so the claim is true for any sequence with $i = 1$. Then, assume we know the claim is true for $i - 1$. That means that after splitting with $i - 1$ vessels, no matter what sequence of actions we did, we know $m \geq 100/2^{i-1}$. When we split some non-empty vessel with the $i$-th empty vessel, the most we can reduce $m$ by is to divide it by 2, which happens if we split the vessel containing $m$ with the empty vessel. After that split we'll have $m \geq (100/2^{i-1})/2$, which is $100/2^i$. So we'll have $m \geq 100/2^i$, which is what we claimed. Note that we could at this point start splitting non-empty vessels with each other, but this can never reduce $m$, as noted above. So our claim will always remain valid.

Having proven the claim, we're almost done with the overall proof of the greedy algorithm. The claim shows us that after we've split with 9 empty vessels, we must have $m \geq 100/2^9$. And it's not possible to ever have a sequence of actions that involves splitting with more than 9 empty vessels, since once an empty vessel is "used", it can never become empty again (no matter what you split it with it will always retain some water). Therefore the smallest possible value of $m$, the least amount of water that can ever be in any non-empty vessel, is $100/2^9$, and this is indeed what the greedy algorithm achieves.

## Problem 4: Covering a spectrum

You want to create a scientific laboratory capable of monitoring any frequency in the electromagnetic spectrum between $L$ and $H$. You have a list of possible monitoring technologies, $T_i$, $i = 1, ..n$, each with an interval $[l_i, h_i]$ of frequencies that it can be used to monitor. You want to pick as few as possible technologies that together cover the interval $[L, H]$. Give an efficient algorithm for finding such a set of technologies.

## Solution: Covering a spectrum

**Algorithm.** **Notation:** $l$-value refers to any $l_i$ and $h$-value to any $h_i$ for $1 \leq i \leq n$.

1. Sort the technologies so that the $l$-values are in nondecreasing order. Break any ties so that the $h$-values are nondecreasing.
2. We will process the technologies in the sorted order and select the next technology to maximize the coverage, that is, the selected technologies cover the interval $[L, m]$ where $m$ is as large as possible. Initially, we set $m = L$ and repeat the following steps while $m < H$.
   (a) A technology $i$ is valid for the position $m$ if $l_i \leq m \leq h_i$. Amongst all the technologies valid for $m$, select the technology with the largest $h$-value and let $k$ be the index of the selected technology.
   (b) Update $m$ to be $h_k$.
3. Output the selected technologies.

[1] You have probably seen proof by induction in your high school math classes. But as a refresher, or in case you haven't seen it before, the idea is that you want to prove some statement for all values of $i$. You start by proving it for $i = 1$ (or some other "base case"). Then you prove that for any $i$, if you assume it's true for $i - 1$ then it must also be true for $i$. Putting these together, you can conclude that the statement is true for all $i$ (greater than or equal to 1).

**Correctness.** Let $S = t_1, t_2, \ldots, t_p$ be the sequence of (indices of) technologies output by the greedy algorithm in the order they were selected during the execution. Since the algorithm processes the technologies in nondecreasing order of $l$-values, we have $l_{t_1} \leq l_{t_2} \leq \cdots \leq l_{t_p}$. For $1 \leq i \leq p$, define $m_i = h_{t_i}$ and $m_0 = L$. Since the technologies in $S$ cover the interval $[L, H]$ and the sequence $S$ is listed in nondecreasing order of $l$-values, we conclude that for $1 \leq i \leq p$, the sequence $t_1, \ldots, t_i$ covers the interval $[L, m_i]$ and that $m_p \geq H$. Furthermore, we have that for $1 \leq i \leq p-1$, $l_{t_i} \leq m_i$.

We will show that $S$ is an optimal solution. For the sake of contradiction, assume there is a sequence $S'$ of technologies such that the technologies in $S'$ cover $[L, H]$ and $|S'| < |S|$, where $|.|$ refers to the length of the sequence. We will prove by contradiction that such an $S'$ cannot exist. Let $S' = t'_1, t'_2, \ldots, t'_q$ where $q < p$. Without loss of generality, assume that the $l$-values of the technologies in the sequence $S'$ are nondecreasing. For $1 \leq i \leq q$, define $m'_i = \max(L, \max_{1 \leq j \leq i} h_{t'_j})$. and $m'_0 = L$. Since the technolognies in $S'$ cover the interval $[L, H]$ and the $l$-values of the sequence $S'$ are nondecreasing, we conclude that for $1 \leq i \leq q$, the sequence $t'_1, \ldots, t'_i$ covers the interval $[L, m'_i]$ and that $m'_p \geq H$. Furthermore, we have $l_{t'_i} \leq m'_i$ for $1 \leq i \leq q-1$

Let $i \geq 1$ be the smallest index in which the sequences $S$ and $S'$ differ. If the first $q$ indices are the same, we define $i$ to be $q+1$. We obtain a contradiction by reverse induction on $i$.

If $i = q+1$, the technologies $t_1, \ldots, t_q$ cover the interval $[L, H]$, which leads to a contradiction since the greedly algorithm goes on to select another tehchnology $t_{q+1}$.

Let $1 \leq i \leq q$. Let $S''$ be the sequence of technologies obatined from $S'$ by exchanging the technology $t'_i$ with $t_i$ and deleting any succeeding technologies in $S'$ whose $l$-value is lesser or equal to $l_{t_i}$. We argue that $S''$ covers $[L, H]$ and that $S''$ is ordered according to the $l$-value. Since the first $i-1$ technologies in $S$ and $S'$ are the same, we have

- $m_{i-1} = m'_{i-1}$,
- $l_{t_i}, l_{t'_i} \leq m_{i-1}$, and
- $h_{t_i} \geq h_{t'_i}$.

The last inequality follows from the condition $t_i$ is the interval with the largest $h$-value among all technologies $j$ such that $l_j \leq m_{i-1}$. As a consequence, exchanging $t'_i$ with $t_i$ does not affect the coverage. By a similar reasoning, any technology $t'_j$ for $j > i$ and $l_{t'_j} \leq l_{t_i} \leq m_i$ can be removed from $S'$ without affecting the coverage. Hence, $S''$ covers $[L, H]$ since $S'$ covers $[L, H]$ and $S''$ is ordered by nondecreasing $l$-values.

**Time complexity.** It takes $O(n \log n)$ time for sorting the technologies.

We can compute the best technology among all valid technologies by iterating through the $l$-values in sorted order and tracking the the maximum $h$-value for technologies where the $l$-value is less than or equal to the current value of $m$. Once a technology is selected, we continue from the next technology with $l$-value greater than $m$ and update the value of $m$. It only takes $O(n)$ time to traverse the technologies while keeping tracking of the desired quantities.

The overall time complexity is $O(n \log n)$.

# 3 Problems

**Problem 5: Maximizing the number of tasks**

You have the following list of tasks to complete:

- Walk the dog (15 minutes)
- Mow the lawn (60 minutes)
- Shovel the snow (45 minutes)
- Take out the trash (2 minutes)

- Clean the pool (45 minutes)
- Wash the windows (75 minutes)
- Wash the car (30 minutes)
- Cook dinner (20 minutes)

You would like to complete as many jobs as you can in a certain amount of time. How would you select the jobs such that you will accomplish the greatest number of jobs in any amount of time?

**Problem 6: Maximizing the profit**

You have the following list of tasks to complete:

- Walk the dog (15 minutes, $5)
- Mow the lawn (60 minutes, $50)
- Shovel the snow (45 minutes, $20)
- Take out the trash (2 minutes, $2)
- Clean the pool (45 minutes, $15)
- Wash the windows (75 minutes, $60)
- Wash the car (30 minutes, $15)
- Cook dinner (20 minutes, $5)

If you have 2 hours, what is the maximum profit you can make? How did you select which tasks to complete?

**Problem 7: Coupons**

Consider the following "coupon collector" problem. There are different varieties of cereal, and each comes with a single coupon for a discount on another box of cereal, perhaps of another variety. You can use multiple coupons when purchasing a new box, up to getting the new box free, but you never get money back. You want to buy one box of each variety, for as little money as possible. Describe an efficient algorithm, which, given as input, for each variety, its price , the value of the enclosed coupon, and the brand for which the coupon gives a discount, computes an optimal order in which to buy the cereal.

**Problem 8: Non-Attacking Kings**

Place the greatest possible number of kings on an 8x8 chessboard so that no two kings are placed on adjacent — vertically, horizontally, or diagonally — squares.

**Problem 9: Party planning (DPV)**

Alice wants to throw a party and is deciding whom to call. She has $n$ people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least five other people whom they know and five other people whom they don't know.

Give an efficient algorithm that takes as input the list of $n$ people and the list of pairs who know each other and outputs the best choice of party invitees. Give the running time in terms of $n$.

**Problem 10: Gas stops (CLRS)**

Professor Midas drives an automobile from Newark to Reno along Interstate 80. His car's gas tank, when full, holds enough gas to travel $n$ miles, and his map gives the distances between gas stations on his route. The professor wishes to make as few gas stops as possible along the way. Give an efficient method by which Professor Midas can determine at which gas stations he should stop, and prove that your strategy yields an optimal solution.

**Problem 11: Bridge Crossing at Night**

A group of four people, who have one flashlight, need to cross a rickety bridge at night. A maximum of two people can cross the bridge at one time, and any party that crosses (either one or two people) must have the flashlight with them. The flashlight must be walked back and forth; it cannot be thrown. Person A takes 1 minute to cross the bridge, person B takes 2 minutes, person C takes 5 minutes, and person D takes 10 minutes. A pair must walk together at the rate of the slower person's pace. Find the fastest way they can accomplish this task.

**Problem 12: Rumor Spreading**

There are $n$ people, each in possession of a different rumor. They want to share the news with each other by sending electronic messages. What is the minimum number of messages they need to send to guarantee that every one of them gets all the rumors? Assume that a sender includes all the rumors he or she knows at the time the message is sent and that a message may only have one addressee.

**Problem 13: Chain Cutting**

You have a chain of $n > 1$ paper clips. What is the minimum number of single clips that must be removed from the chain so that it would be possible to create a chain of any integer length between 1 and $n$ clips, inclusive, from the resulting pieces?