Greedy algorithms

•••

cuz data tastes good

A simple coin change problem

You have 20$, 10$, 5$, 2$ and 1$ bills
→ you want to pay N$ with as few bills as possible

e.g 58$ = 2*20$ + 1*10$ + 1*5$ + 1*2$ + 1*1$ → 6 bills

Strategy: let’s try to use the biggest bills as long as possible

A simple coin change problem

Algorithm:

```
n = int(input())
bills = [20, 10, 5, 2, 1]
nbill = []

for bill in bills:
     nbill.append(n // bill)
     n -= nbill[-1] * bill

print(" + ".join("%d*%d$" % (nbill[i], bills[i])
                 for i in range(len(bills))))
```

Complexity:

O(len(bills)) ⟶ const, O(1)

Why it works:

The dollar system is called a *canonical* coin system

⟶ cf Xuan Cai, https://arxiv.org/pdf/0809.0400.pdf

Non-canonical system: [9, 4, 1]

Optimal: 12 = 3*4

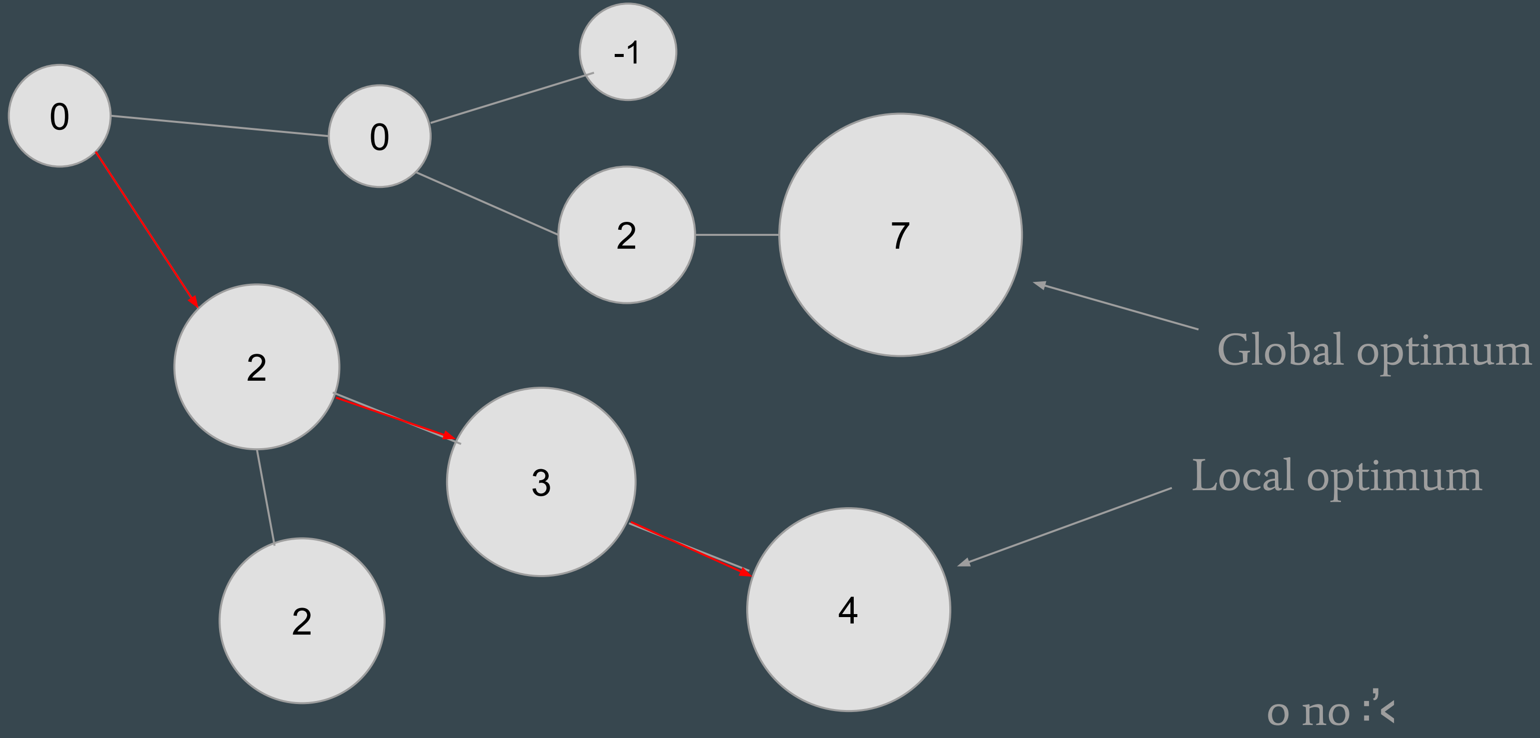
Greedy: 12 = 1*9 + 3*1

So what's a greedy algorithm?

- it's an algorithm that makes **locally-optimal choices** at each step
- the global solution might not be the optimal one
- but many well-known greedy algorithms are proven to find the optimal solution, e.g:
  - Kruskal's and Prim's algorithms (minimum spanning tree)
  - Dijkstra's and A* algorithms (shortest path)

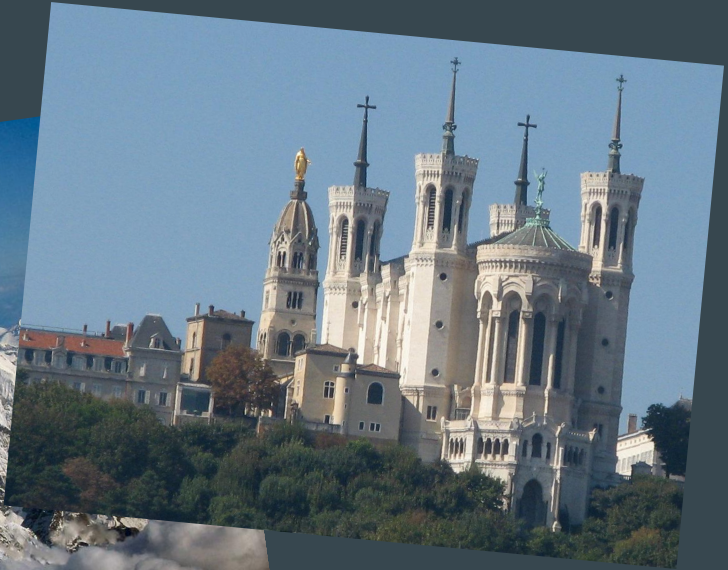
Greedy algorithms are usually **much faster** than their *dynamic programming* equivalents, but their correctness can be hard to prove...

So what's a greedy algorithm? - Understand with the search space



Example of a non-optimal greedy algorithm

You want to climb on the highest mountain.
You think “if I keep going up, I will reach the highest point”.
You end up on the Fourvière hill.
Not quite as tall as Mount Everest.



Credits

Slides: Louis Sugy for INSAlgo, modified by Louis Hasenfratz

Pictures: Wikipedia

Picture of the cookie monster: fair use of the character from *Sesame Street*