Greedy Algorithms and String Matching

Christian Garcia

May 3, 2023

Questions

0.1 Greedy Algorithms

0.1.1 Create the infamous greedy algorithm counting change in either Python or C. The change amounts that must be included in the algorithm are the half-dollar ($0.50), quarter ($0.25), dime ($0.10), nickel ($0.05), and penny ($0.01).

0.1.2 Describe why the counting change algorithm is considered a greedy algorithm.

0.1.3 **Create another infamous greedy algorithm named the knapsack problem in C. The main point of the greedy knapsack problem is to fill your bag with items without exceeding the weight limit and obtaining optimal value. For this problem, the knapsack has a capacity of 20. The weight and value of an item cannot exceed 10.**

0.1.4 **Describe what makes the knapsack problem a greedy algorithm.**

0.1.5 **Based on your analysis of the two greedy algorithms, what classifies an algorithm as a greedy algorithm?**

0.2 String Matching

0.2.1 This code implements a function to compute the levenshtein distance in Python:

```python
def tail(l: List[T]) -> List[T]:
    return l[1:]

def head(l: List[T]) -> T:
    return l[0]

def lev(a: List[T], b: List[T]) -> int:
    if len(b) == 0:
        return len(a)
    if len(a) == 0:
        return len(b)
    if head(a) == head(b):
        return lev(tail(a), tail(b))
    return 1 + min(
        lev(tail(a), b),
        lev(a, tail(b)),
        lev(tail(a), tail(b)))
```

0.2.2 Pick two different strings and use this algorithm to find the levenshtein distance, *by hand* between the two strings. Make sure to show each step.

0.2.3 What dose the levenshtein distance algorithm tell us about strings, based on the previous exercise?