Greedy - Example 2 Continued

Start:D Finish:L Greedy will chose D,G,L. Total distance of 330km, optimum

Start:L Finish:D Greedy will chose L,C,D. Total distance of 370km, sub-optimum

The Travelling Salesperson Problem

This is a classic problem in Computer Science, and is essentially a variant on the above: The Salesperson must visit all the cities on the network exactly once, without "retracing his/her steps". Given the distances, calculate the shortest such route.

The Coins Problem

In a certain country, coins are available in denominations of $c_1, c_2, c_3, \ldots, c_d$. Find the least number of coins that are

necesssary to make up some fixed amount n, assuming we have a large number (∞ if you want) of each coin.

Greedy Solution to coins problem

At each step, choose largest coin that does not excede remaining amount.

Example 1 $c_i = 1, 2, 5, 10, 20, 50$ and $n = 28$:

step number	1	2	3	4
coin chosen	20	5	2	1
running total	20	25	27	28

so, a total of 4 coins: this is the best answer in this case.

Example 2 $c_i = 4, 5$ and $n = 20$:

step number	1	2	3	4
coin chosen	5	5	5	5
running total	5	10	15	20

again the greedy strategy gets the best answer.

Greedy Solution to coins problem - Continued

What about the cases

1. $c_i = 4, 5$ and $n = 19$
2. $c_i = 4, 5$ and $n = 18$
3. $c_i = 4, 5$ and $n = 17$

The (0/1) Knapsack Problem

In this problem, we have a bag/container of (weight) capacity K kilograms. We have n distinct items with weights $w(1), w(2), w(3), \ldots w(n)$ (kg) and corresponding values $v(1), v(2), v(3), \ldots v(n)$ (euro). The aim is to put as many items as possible in to our bag, in order to obtain maximum value.
Note:

- There is only one of each item (which is why it is called the 0/1 knapsack problem)

- If $w(1) + w(2) + w(3) + \cdots + w(n) < K$ then the problem is trivial (i.e. easy to solve): just take everything!

Knapsack Example

Suppose our bag capacity is $K = 4$ kg, and we have $n = 4$ items A, B, C, D, with weights $w(i)$ of 2,3,4,1 and associated values $v(i)$ 5,6,7,2.

	A	B	C	D
weight (kg)	2	3	4	1
value (euro)	5	6	7	2

Clearly all the items don't fit in the bag (since 2+3+4+1 > 4), so which items should we choose?

Brute Force Solution to 0/1 Knapsack Problem

Consider/Enumerate all possibilities:

- Choose just one item: 4 possibilities
- Choose any two items: 6 possibilities
- Choose any three items: 4 possibilities
- Choose four items: 1 possibility

(In actual fact, we don't need to look at all these options: In this example, we can't pick 4 items: We can't even fit any 3 items in the bag.)

For any item, we can either take it or not, i.e. 2 choices. 2 choices each for n items means 2^n possibilities. If for example we had 50 items, 2^{50} is about 1,000,000,000,000,000. Even if we have to consider a fraction of these possibilities, this algorithm would be unworkable for anything but a supercomputer. The Brute Force Solution is $\mathbf{O}(2^n)$.

Greedy Solution to 0/1 Knapsack Problem

Greedy on weight We at each stage pick the lightest item: So we pick D, then A, then terminate with final value of 7 euro in the bag.

Greedy on value We at each stage pick the most valuable item: So we pick C, then terminate with final value of 7 euro in the bag.

Greedy on euro/kg We at each stage pick the most value per kilogram item: So we pick A, then D, then terminate with final value of 7 euro in the bag.

Algorithm Strategies: Recursion

(Sometimes this is viewed as a *methodology* rather than a strategy - i.e. it can be used within other strategies.) Problems which allow a recursive solution have two features:

1. They can be (easily) solved for some small "size" (n) of the problem (typically n is 0 or 1). This is called the base case.
2. The general problem of size n can be broken up (re-written) in terms of smaller problem(s) of size p where $p < n$:
   - Typically p is $n - 1$, or $n/2$, or $n - r$ for some r.
   - The way in which n gets reduced to p must ensure we reach the base case.

   This is called the recursive step.

Recursion - Example 1

Consider the calculation of $n!$ (the factorial of a number):

Base Case Do we know how to solve it for a small value of n? Yes, we know $0! = 1$, or $1! = 1$

Recursive Step Do we know how to write $n!$ in terms of the factorial of a smaller number? Yes - $n!$ is just n multiplied by $(n-1)!$. So we can re-write the problem in terms of one of smaller size p, where in this case p is $n-1$.