ALGORITHM DESIGN STRATEGIES

Brute Force Strategy

The Brute Force method is normally the most obvious method of solving a problem, but often the least efficient: It consists in listing *all* possible solutions, and then simply checking which is the best one (sometimes it is known also as *exhaustive search*). It can only be applied to problems where we can actually ennumerate all possible solutions.

Brute Force: Example 1

Suppose we have a number (n) of shops/businesses in a row on a street. We know the yearly profit/loss each makes. We want to "buy" a block of shops such as to maximize our yearly profit. Note that

1. We can ignore the "purchase price" of the shop (pretend it is being given away for free).
2. The shops we "buy" must all neighbor one another.

Given the list of n numbers, design an algorithm to decide which to purchase. Examples are:

Profits = $[6, 1, 4, 7]$

Profits = $[2, 6, 1, -4, 3]$

Profits = $[-1, -4, -7]$

Profits = $[-5, 8, 5, -3, 4, -2, 1]$

The Brute Force approach here is to “consider all possibilities”:
Lets suppose $n = 4$.

Option 1 Buy no businesses!

Option 2 Buy one business (i.e. buy 1, or 2, or 3, or 4)

Option 3 Buy 2 neighboring businesses (i.e. buy 1&2 or 2&3 or 3&4)

Option 4 Buy 3 neighboring businesses (i.e. buy 1&2&3 or buy 2&3&4)

Option 5 Buy all 4 businesses.

We simply write down the profits for all these possibilities, and find the largest number.

How many cases do we have to consider?

For the general case of n shops:

Ways of choosing 0 shops: 1

Ways of choosing 1 shop: n

Ways of choosing 2 neighboring shops: n-1

Ways of choosing 3 neighboring shops: n-2

. . . . . .

Ways of choosing r neighboring shops: n-r+1

Ways of choosing n-1 shops: 2

Ways of choosing n shops: 1

So the total number of cases is

$$1 + 2 + 3 + \cdots + (n-1) + n + 1 = \frac{n}{2}(n+1) + 1 = \mathbf{O}(n^2)$$

We can write the algorithm for this in pseudocode:

Pseudocode - Brute Force solution of “Shops” Problem

```
BEGIN
IN(n, p[1], p[2], ...  , p[n])
FOR X ← 1 TO n DO
      FOR Y ← X TO n DO
            BEGIN
            S ← 0
            FOR Z ← X TO Y DO
                  S ← S + p[z]
            OUT(S)
            END
END
```

As an aside, suppose we were to modify the problem, and remove the restriction that we have to purchase *neighboring* shops. This increases the number of possibilities we have to enumerate in Brute Force:

Ways of choosing 0 shops: 1 ($\equiv \binom{n}{0}$)

Ways of choosing 1 shop: n ($\equiv \binom{n}{1}$)

Ways of choosing 2 shops: $n(n-1)/2 \equiv \binom{n}{2}$

Ways of choosing 3 shops: $\binom{n}{3}$

. . . . . .

Ways of choosing r shops: $\binom{n}{r}$

Ways of choosing n-1 shops: n ($\equiv \binom{n}{n-1}$)

Ways of choosing n shops: 1 ($\equiv \binom{n}{n}$)

The binomial theorem states that

$$
\begin{aligned}
(a+b)^n &= \sum_{r=0}^{n} \binom{n}{r} a^{n-r} b^r \\
&= \binom{n}{0} a^n + \binom{n}{1} a^{n-1} b + \binom{n}{2} a^{n-2} b^2 + \dots \\
&+ \binom{n}{r} a^{n-r} b^r + \cdots + \binom{n}{n-1} a b^{n-1} + \binom{n}{n} b^n
\end{aligned}
$$

so if we set $a = 1$ and $b = 1$ we get

$$
\sum_{r=0}^{n} \binom{n}{r} 1^{n-r} 1^r = \sum_{r=0}^{n} \binom{n}{r} = (1+1)^n = 2^n
$$

which means the brute force method for solving this modified shops problem is $\mathbf{O}(2^n)$.

An Aside....

A simpler way to calculate the complexity: for each shop, we have two possibilities (choose it or don't). Since we have these two choices independently for n shops, we multiply 2 by itself n times to get the total number of choices, 2^n.

Brute Force - Example 2

Linear/Sequential Search (of an unordered list) is an example of a brute force technique. We simply consider every possible option to find X in the list L.

Brute Force - Example 3

Suppose we want to find the best way from Galway to Kilkenny! Suppose we consider a road network of Ireland which has just 40 cities/towns, and the distances along each road are given. A brute force approach is to consider all routes:

- Start at Galway
- Go to any of the other 38 towns (38 choices).
- Go to any of the other 37 towns (37 choices).
- Go to any of the other 36 towns (36 choices).
- etc.

This leads to 38! different routes. Find the distances for all these routes and take the smallest distance.

38! is an enormous number, approximately 10^{44}, i.e. 1,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000. A program using this approach, written in any current

programming language, run on any machine, would not finish in the lifetime of the Universe!