Dynamic Programming

Victor Chui

March 17, 2022

1 Prerequisite

1.1 Bitwise operator

Computers are operated in binary system. Therefore it is the fastest to use bitwise operators to handle cases possible as they are more efficient.

Most common operators are:

```
int a = 1;
int b = 2;

a << 1; //it shifts one bit to the left, which means a is
    multiplied by 2.
b >> 1; //it shifts one bit to the right, which means b is divided
    by 2.
```

In general cases, `<<` multiplies the number or variable by 2^n. Similarly, `>>` divides the number or variable by 2^n.

Notice that the compiler is smart enough to perform bitwise operation without user specification. Below is a code snippet:

```
int a = 6;
int b = 16;
int c = a*b;
std::cout << c << std::endl;
```

In this problem, b could be represented by 2^n. The compiler will take this integer multiplication into bitwise shifting. Therefore, we can conclude that $6 * 16$ is equal to 6 `<<` 4.

1.2 Bitwise AND, OR, XOR, Invert

Let's look at the two below code snippet to better explain bitwise operators. First:

```
int a = 1;
int b = 1;
cout << a&b << endl;
```

In the first code snippet, the terminal will print out 1 or True.
We can conclude that:

$$AND(m,n) = \begin{cases} 0, & \text{for a or b equals to 0} \\ 1, & \text{for a and b equals to 1} \end{cases}$$

Below is the second snippet:

```
int a = 0;
int b = 0;
cout << a|b << endl;
```

The terminal will print out 0 or False in this snippet.
We can conclude that:

$$AND(m,n) = \begin{cases} 0, & \text{for a and b equals to 0} \\ 1, & \text{for a or b equals to 1} \end{cases}$$

2 Knapsack Problem

2.1 Introduction

There are a set of weights $W = [2, 3, 4, 5]$ with cooresponding profit of $P = [1, 2, 5, 6]$. The question asks about the maximum profit from those weights with the maximum capacity of 8.

Below is the generic algorithm for solving this 0/1 Knapsack problem.

2.2 Breakdown

$$V(i,x) = max[V(i-1,w), V(i-1, w-w[i]) + P[i]]$$

2.3 Code Snippet in C++

```
#include <bits/stdc++.h>
using namespace std;
```

2.4 Conclusion

To use the Knapsack Algorithm for this

2.5 Reference

Visually explained by Abdul Bari:
`https://www.youtube.com/watch?v=nLmhmB6NzcM&t=944`