TCSS 343 - Week 4

Jake McKenzie

August 19, 2018

Dynamic Programming

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."
. . .
Richard Bellman's **Principle of Optimality**

"What we choose means more than what was handed to us by chance."
. . .
Ada Palmer

" If 'dynamic programming' didn't have such a cool name, it would be known as 'populating a table'".
. . .
Mark Dominus

1. Today we're going to explore dynamic programming. Below are three implementations of the fibonacci algorithm that I wrote in python. I want you to draw the **"tree"** for each then reflect on how the "bottom up" apprach is different from the other two? (Hint: They are all trees but also different types of trees. This is a key insight in my opinion in idea in understanding dynamic programming)

```python
1    # recusive fibonacci
2    def F(n):
3        if n == 0: return 0
4        elif n == 1: return 1
5        else: return F(n-1) + F(n-2)
```

```python
7    # "top down" memoized recursive fibonacci
8    memo = {}
9    def Fib(n):
10       if n in memo: return memo[n]
11       if n == 0: f = 0
12       elif n == 1: f = 1
13       else: f = Fib(n-1) + Fib(n-2)
14       memo[n] = f
15       return f
```

```python
17   # "bottom up" iterative fibonacci
18   def fib(n):
19       fn = [0,1]
20       for i in range(n >> 1):
21           fn[0] += fn[1]
22           fn[1] += fn[0]
23       return fn[n % 2]
```

i	w_i	h_i	n_i
1	1	1	3
2	2	4	2
3	3	6	2
4	4	5	1
5	5	7	1
6	6	8	1

2. Suppose Santa has 6 kinds of toys, each kind of toy has its own weight w_i in tons, happiness rating h_i in ... joy, and quantity n_i. Santa would like to maximize the total hapiness of the children but the total weight of his bag cannot exceed 17 tons. Their weight, hapiness rating and quantity are defined above. Please help Santa by filling in the DP table below, where dp[i][j] indicates the maximum value you can get with weight less or equal to j using toys 1 to i. What is the final solution to this problem and briefly explain how you came to this solution. To help you get started, 23 was generated by solving the equation $i_1 + 2i_2 + 3i_3 \leqslant 15$ which gives you the most value. That value was found by $3 \cdot 1 + 2 \cdot 4 + 2 \cdot 6 = 23$. 12 was found by solving the equation $i_1 + 2i_2 + 3i_3 + 4i_4 + 5i_5 + 6i_6 \leqslant 6$ which gives you the most value. That value was found by $2 \cdot 6 = 12$

i\w	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0																	
2	0																	
3	0															23		
4	0																	
5	0																	
6	0						12											

```
1    # Rules: X,Y > 0
2    def game_of_Stanley_Gill(X,Y):
3        x = X
4        u = X
5        y = Y
6        v = Y
7        while x != y:
8            if x > y:
9                x = x - y
10               u = v + u
11           elif y > x:
12               y = y - x
13               v = u + v
14       print((x + y) >> 1) #GCD(X,Y)
15       print((u + v) >> 1) #LCM(X,Y)
```

3. This is a game shown to Edsgar Dijkstra(pictured right) when he was still an undergraduate, attributed to Stanley Gill(pictured left), an early computer scientist. Now it is true that $2XY = xv + yu$, which can be seen when the variables are initialized. But it is always true given that $X, Y > 0$? Show why this is true. (Hint: You can use the comments to help you along. Remember that the $>> 1$ operation is the same as $/2$.)

4. Hopefully in problem 1 I was able to impart on you the reality that most dynamic programming problems can be thought of as graph problems, specifically directed acyclic graphs, DAG for short. Even if you don't solve them with DAGs I think it's a useful headspace for a large class of problems. We can typically solve DP problems without constructing a graph. If we aren't going to use a graph there are other useful tools that you may not have run across. Let's try to understand what a bitmask is. I've been peppering them into the packets all quarter but let's finally dive deep into them. Mask in bitmask means hiding something. Bitmask is nothing but a binary number that represents something.

Please connect the corresponding "bitmask" with their set operations or arithmetic operations. I encourage you to play around with the operation and really try understand what they're doing. All operations are on two intergers X and Y. I know this may seem disjointed from what you've covered in class but I have used all of these in solving DP problems. Often times we aren't just worried about our time complexity, but we're also worried about space complexity. Using integers instead of arrays to represent data is sometimes vital in solving hard problems. For example: we may represent the set $\{5, 4, 3, 2, 1\}$ as 11111 in binary which is 31 in decimal and the set $\{4, 2, 1\}$ as 01011 in binary which is 11 in decimal.

a) $X \& 1$ I)Union of two sets

b) $X \wedge Y$ II)Arithmetic negation

c) $(X >> 31)\&1$ III)Test for set membership

d) $X|Y$ IV)2 to the power of X(also Singleton Set)

e) $X \& Y$ V)Barrel shift left

f) $X \& = (X - 1)$ VI)Intersection of two sets

g) $(X << Y)|(X >> (32 - Y))$ VII)Signed bit

h) $\sim X + 1$ IX)Clears lowest "ON" bit in X

i) $1 << X$ X)Symmetric Difference of two sets

j) $X \& (1 << Y)! = 0$ XI)Even/Odd check

4. Robert Sapolsky is a famous neuroendocrinologist (big word that means he studies hormones & stress in humans and primates) and he's one of my favourite thinkers. He likes to use this sequence of numbers, $4, 14, 23, 34, \ldots$ in his neuroscience class to illustrate an important point in categorical thinking. For purposes of our algorithms course I will call these numbers "Sapolsky Numbers". Please compute the 10th Sapolsky Number. Can you come up with a recurrence formula for the Sapolsky numbers? (If you get stuck ask me questions)

Why was this hard?