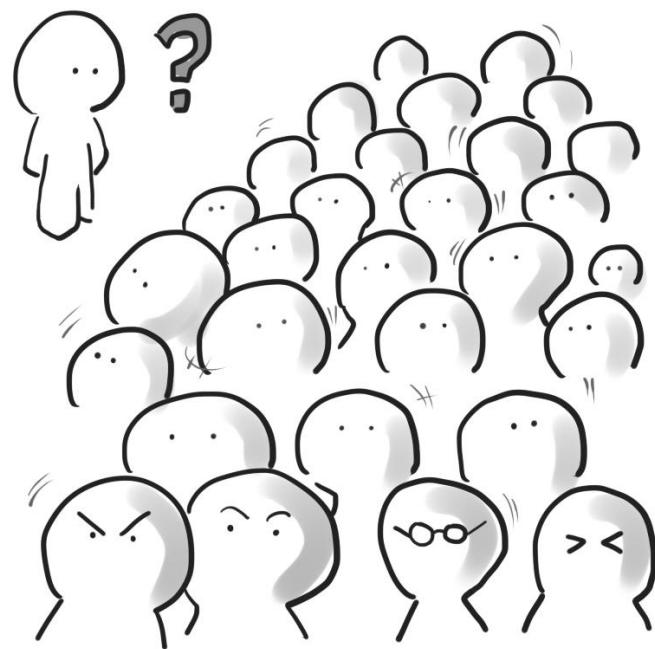


State space search

By Payongkit XI



❖ คำตอบໄດ້ມາจากการลำดับของการตัดสินใจ

- ❖ Sorting : สลับคູໃດກ່ອນ ?
- ❖ Activity Selection : ทำກິຈกรรมໃດກ່ອນ ?
- ❖ MST : ເລືອກເສັ້ນເຊື່ອມໄດ້ມາເປັນຂອງ MST ?
- ❖ Knapsack : ພຍົບຂອງຊື່ນໄດ້ໃສ່ຄຸງເປົ່າ ?

❖ Greedy :

- ❖ ວິທີເລືອກແບບ greedy ທີ່ໃຫ້ນໍາໄປສູ່ຄຳຕອບທີ່ດີສຸດ (ເຈອກຮົນແບບນີ້ໄມ່ມາກ)

❖ Dynamic Programming

- ❖ ບໍ່ຄຳຕອບຍ່ອຍທຸກແບບ ເພື່ອໃຫ້ຕັດສິນໃຈສ້າງຄຳຕອບຂອງປົ້ນຫາໃໝ່
- ❖ ເຮົວເນື້ອຈຳນວນປົ້ນຫາຍ່ອຍທັງໝົດມີໄມ່ມາກ

❖ ແຕ່ມີປົ້ນຫານຳມາກມາຍ

- ❖ ໄມຮັວງທີ່ເລືອກແບບ greedy ທີ່ດີສຸດ
- ❖ ຈຳນວນປົ້ນຫາຍ່ອຍນີ້ມາກ

ລຸຍ້າທຸກຮົບແບບ
ແຕ່ຈະລຸຍແບບຈລາດ ၅

Sum of Subset Problem

- ❖ **input** : k และ $D = \{d_1, d_2, d_3, \dots, d_n\}$
 k และ d_i เป็นจำนวนจริง
- ❖ **output** : หา $S \subseteq D$ ที่ผลรวมของ S มีค่าเป็น k

$$D = \{1, 4, 1, 9, 7\}, \quad k = 11$$

$$S = \{1, 1, 9\}, \quad \{4, 7\}$$

Sum of Subset Problem : รูปแบบผลเฉลย

❖ Sum of subset

- ❖ output : หา $S \subseteq D$ ที่ผลรวมของ S มีค่าเป็น k

❖ คำตอบเป็นเซต

- ❖ ใช้ bit vector (อาร์เรย์ 1 มิติ) $X = \langle x_1, x_2, x_3, \dots, x_n \rangle$
- ❖ $x_m = 1 \rightarrow d_m \in S$
- ❖ $x_m = 0 \rightarrow d_m \notin S$

❖ ตัวอย่าง

- ❖ input : $D = \{2, 10, 3, 5, 7\}$, $k = 14$
- ❖ output : $X = \langle 1, 0, 0, 1, 1 \rangle$

Sum of Subset Problem : ลุยกับทุกเซตย่อย

- ❖ **input** : $D = \{2, 10, 3, 5, 7\}$, $k = 14$
- ❖ แจกแจงเซตย่อยของ D ทั้งหมดมาตรวจสอบ
- ❖ คิดถึง **binary counter**
- ❖ ข้อ : ต้องแจกแจงเซตย่อย 2^n เซต

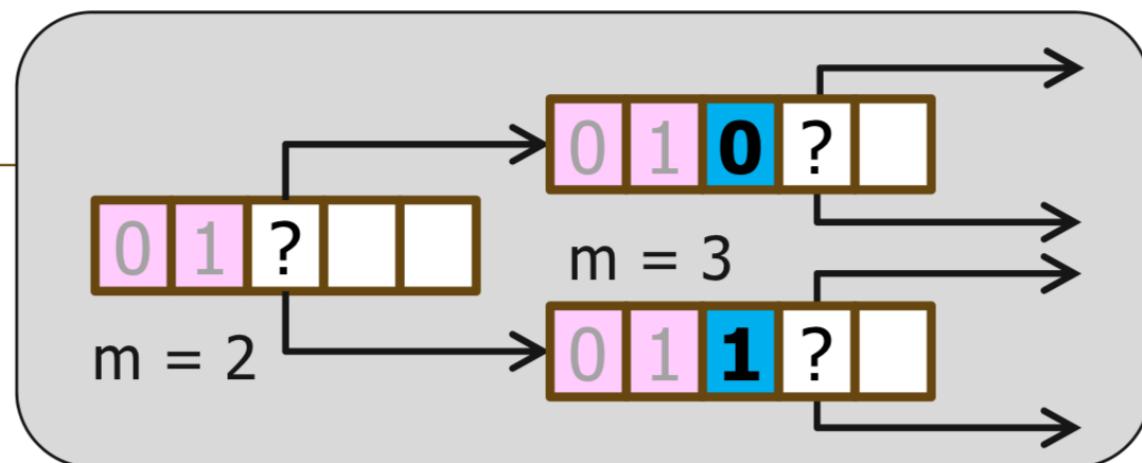
$<0,0,0,0,0>$
$<0,0,0,0,1>$
$<0,0,0,1,0>$
$<0,0,0,1,1>$
$<0,0,1,0,0>$
$<0,0,1,0,1>$
$<0,0,1,1,0>$
$<0,0,1,1,1>$
$<0,1,0,0,0>$
.
.
.
$<1,1,1,0,1>$
$<1,1,1,1,0>$
$<1,1,1,1,1>$

Sum of Subset Problem : Binary counter pseudo

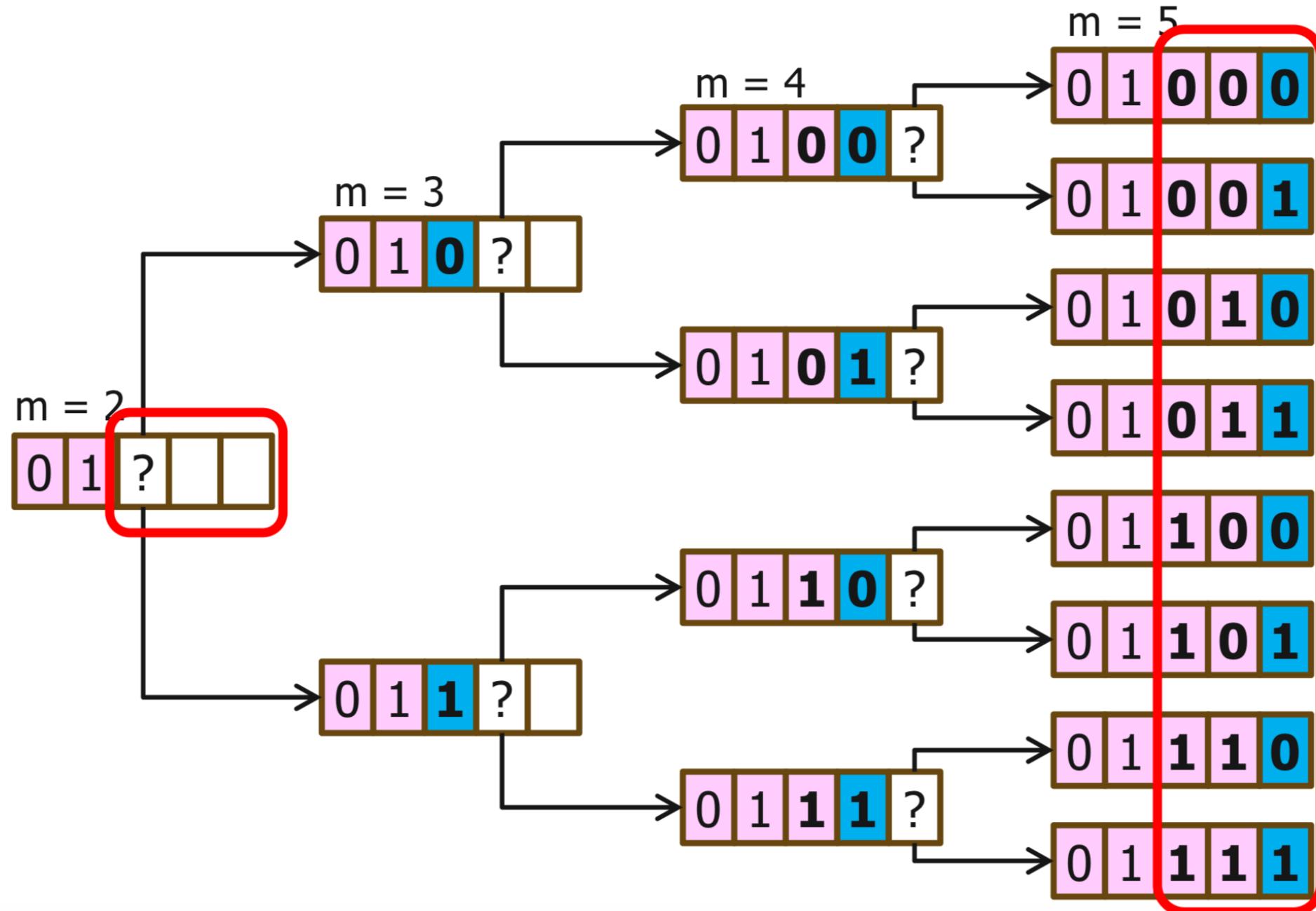
```
bcounter( n ) {  
    x = new array[1..n] with all 0's  
    count( x, 0 )  
}  
count( x[1..n], m ) {  
    if ( m == n ) print( x )  
    else {  
        x[m+1] = 0; count( x, m+1 )  
        x[m+1] = 1; count( x, m+1 )  
    }  
}
```

เติม $x[1..m]$ แล้ว
จงเติมที่เหลือ

เติมครบแล้ว



Sum of Subset Problem : Binary counter pseudo



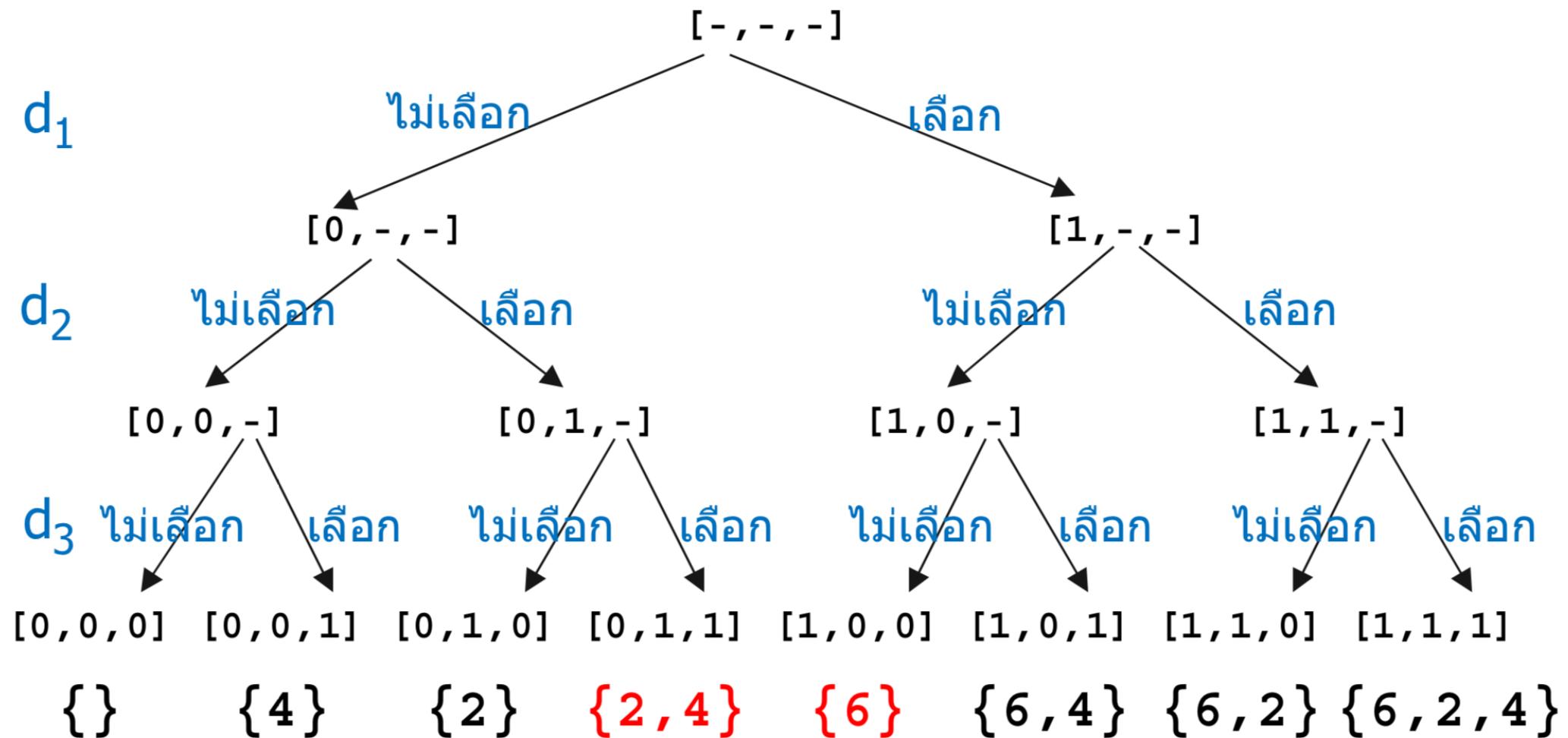
Sum of Subset Problem : តុលាបែពុកមេដីយោ

```
subsetSum( d[1..n] , k ) {  
    x = new array[1..n] with all 0's  
    subsetSum( d, k, x, 0 )  
}  
subsetSum( d[1..n] , k, x[1..n] , m ) {  
    if (m == n) {  
        if (sum(d, x) == k) print( x )  
    } else {  
        x[m+1] = 0; subsetSum( d, k, x, m+1 )  
        x[m+1] = 1; subsetSum( d, k, x, m+1 )  
    }  
}
```

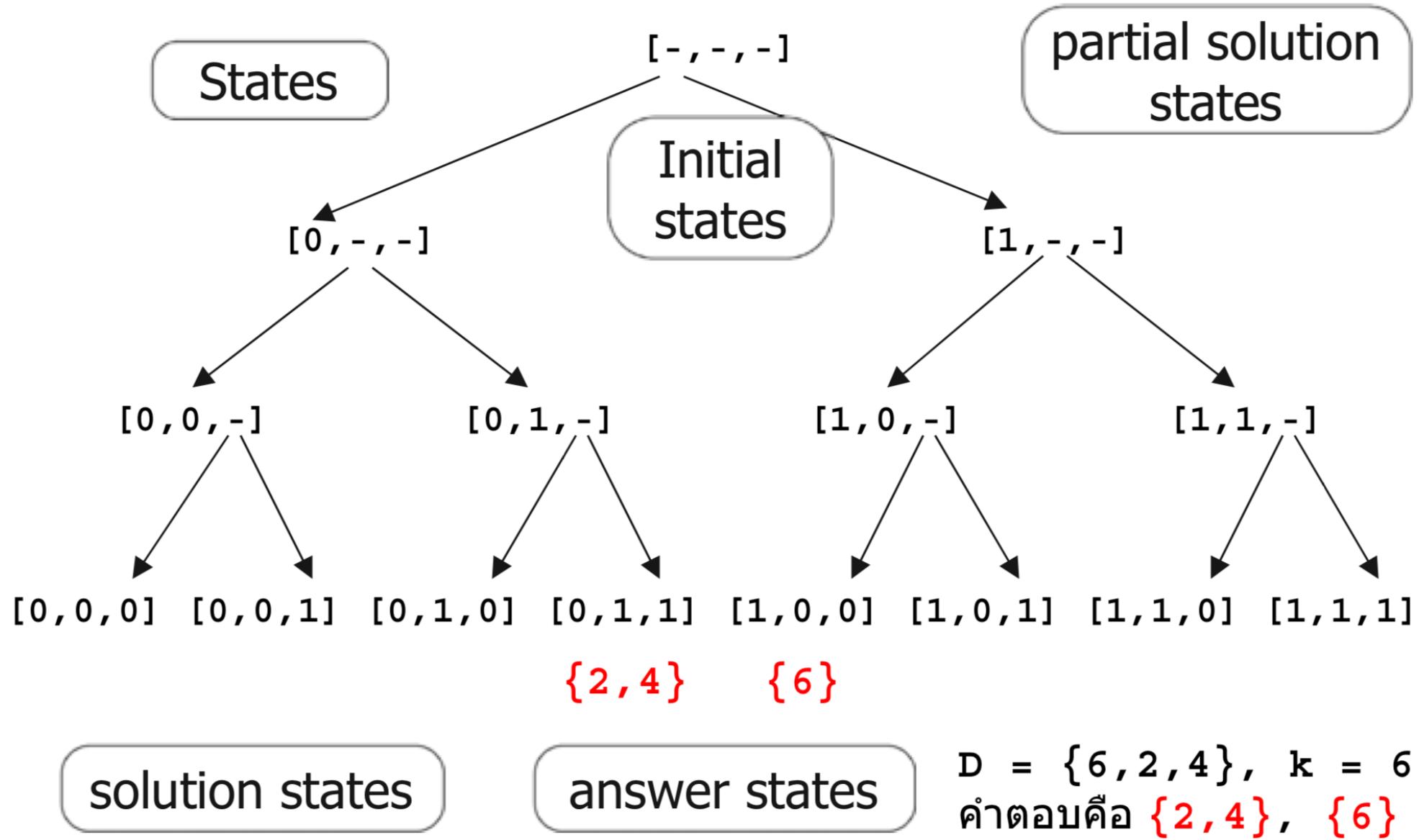
$\Theta(n2^n)$

Sum of Subset Problem : ลุยไปทุกเซตย่อย

$D = \{6, 2, 4\}$, $k = 6$

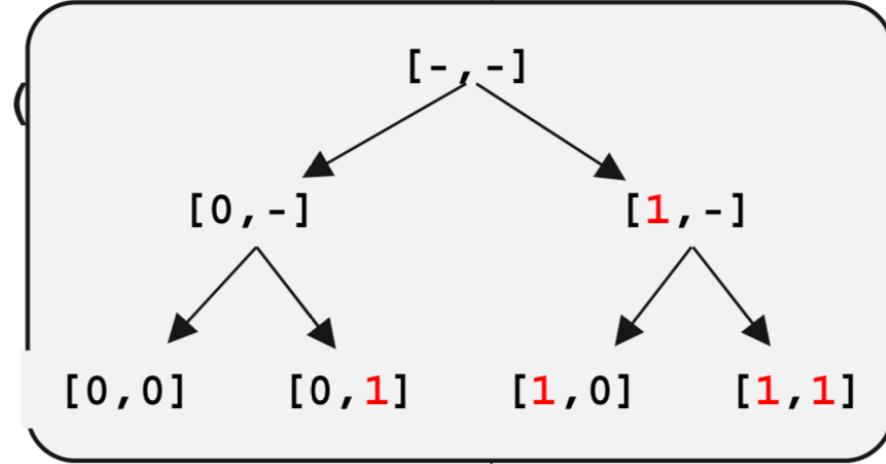


State Space:

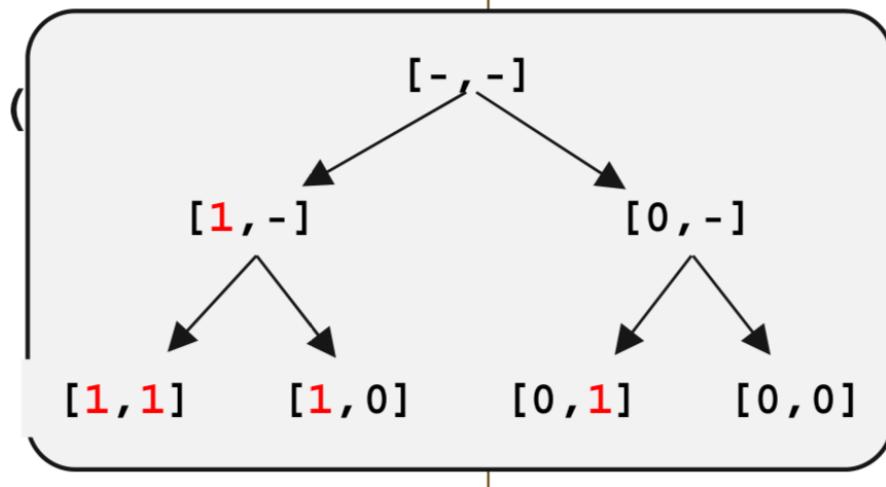


Sum of Subset Problem : state space มีหลายรูปแบบ

```
subsetSum( d[1..n], k, x[1..n], m ) {  
    if (m == n) {  
        if (sum(d, x) == k) print()  
    } else {  
        x[m+1] = 0; subsetSum( d,  
        x[m+1] = 1; subsetSum( d,  
    }  
}
```

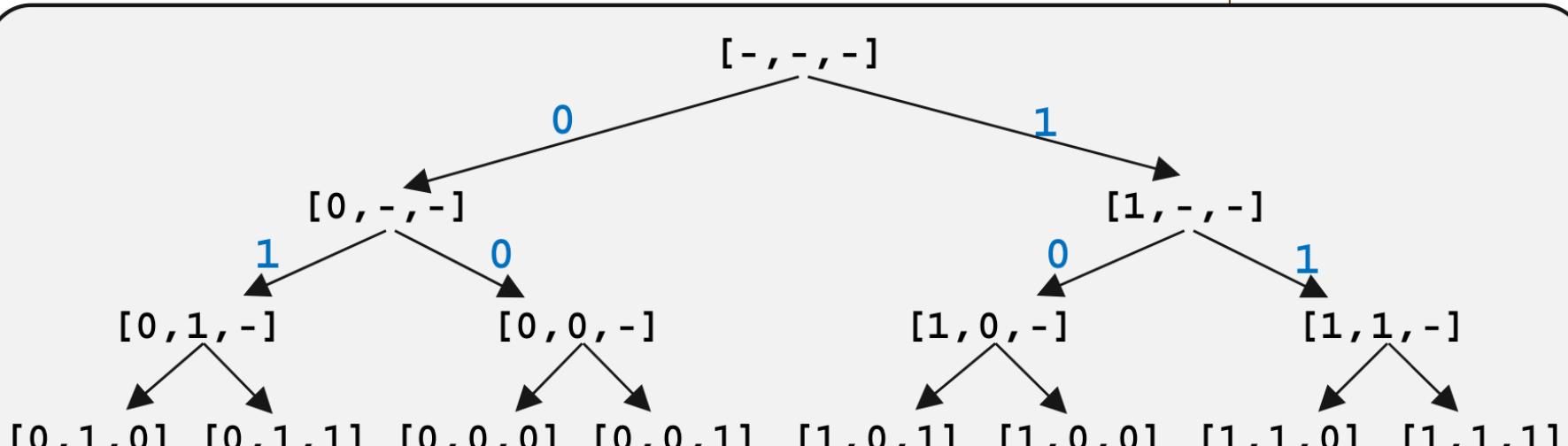


```
subsetSum( d[1..n], k, x[1..n], m ) {  
    if (m == n) {  
        if (sum(d, x) == k) print()  
    } else {  
        x[m+1] = 1; subsetSum( d,  
        x[m+1] = 0; subsetSum( d,  
    }  
}
```



Sum of Subset Problem : state space มีหลายรูปแบบ

```
subsetSum( d[1..n] , k, x[1..n] , m ) {  
    if (m == n) {  
        if (sum(d, x) == k) print( x )  
    } else {  
        x[m+1] = random( {0,1} ) // 0 or 1  
        subsetSum( d, k, x, m+1 )  
        x[m+1] = 1 - x[m+1]  
        subsetSum( d, k, x, m+1 )  
    }  
}
```



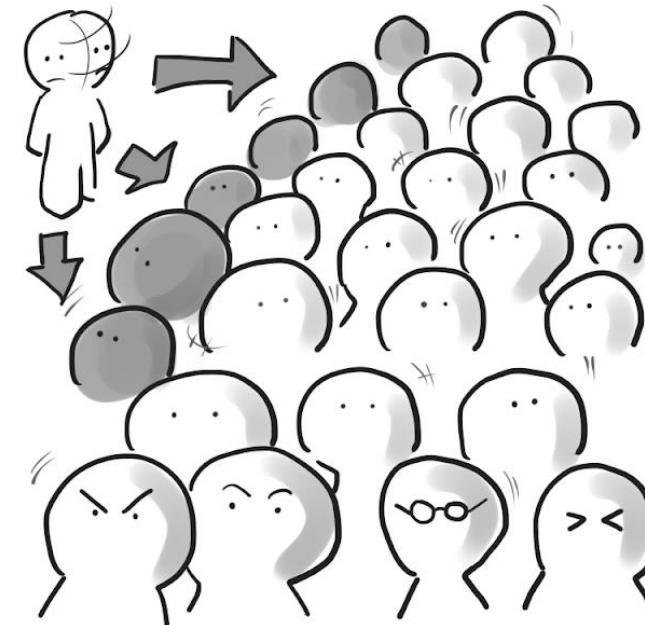
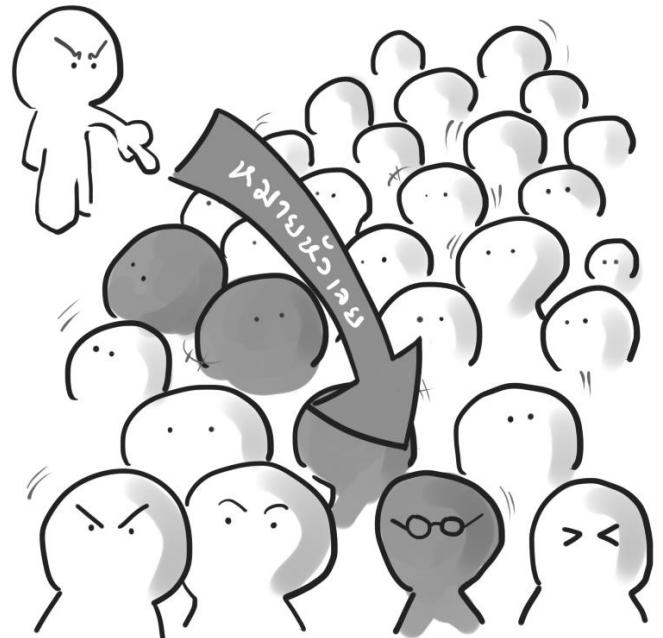
Quiz!!!!!! : 0/1 Knapsack problem : เขียน state space ของ ปัญหานี้ จากการที่มีของ 4 ชิ้น

- ❖ ของ n ชิ้นมีหมายเลข : $1, 2, 3, \dots, n$
- ❖ แต่ละชิ้นหนัก : $w_1, w_2, w_3, \dots, w_n$
- ❖ แต่ละชิ้นมีมูลค่า : $v_1, v_2, v_3, \dots, v_n$
- ❖ ถุงเป็นหนึ่งใบจุของได้น้ำหนักไม่เกิน W
- ❖ ปัญหา : จงเลือกของใส่ถุง เพื่อให้
 - ❖ ถุงไม่ขาด
 - ❖ ได้มูลค่ารวมมากสุด

1		2.5Kg. \$400
2		1.4Kg. \$50
3		2.8Kg. \$350
4		0.7Kg. \$150
		4Kg.

DFS & BFS

Depth first search and Breadth first search



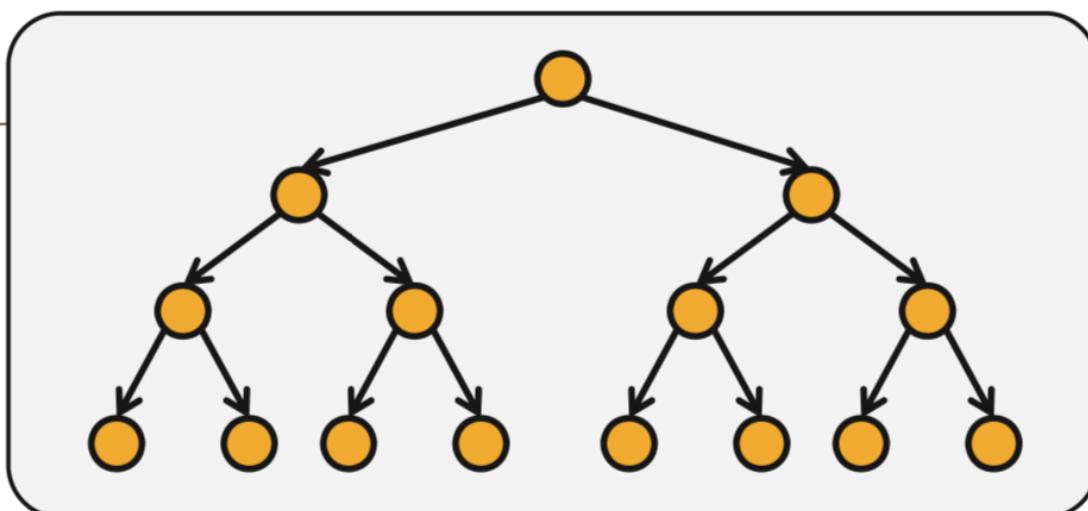
By Payongkit XI

- ❖ คล้ายกับ DFS และ BFS ของการค้นปัมในกราฟ
- ❖ แต่ในที่นี่ เราค้นในปริภูมิสถานะ
 - ❖ ไม่มีโครงสร้างมาให้ค้น
 - ❖ ปัมสถานะถูกสร้างขึ้นระหว่างการค้น
 - ❖ เส้นเชื่อมระหว่างปัม คือการผลิตปัมใหม่จากปัมเดิม
- ❖ ถ้าเขียนแบบเรียกช้า
 - ❖ stack frame ที่เก็บพารามิเตอร์ต่าง ๆ ของการเรียกช้า แทนสถานะของการค้น
- ❖ ถ้าเขียนแบบวนทำช้า
 - ❖ ต้องสร้างที่เก็บข้อมูล ไว้แทนสถานะของการค้น

Sum of Subset Problem : DFS (recursive)

```
subsetSum( d[1..n] , k ) {  
    x = new array[1..n] with all 0's  
    subsetSum( d, k, x, 0 )  
}  
subsetSum( d[1..n] , k, x[1..n] , m ) {  
    if (m == n) {  
        if (sum(d, x) == k) print( x )  
    } else {  
        x[m+1] = 0; subsetSum( d, k, x, m+1 )  
        x[m+1] = 1; subsetSum( d, k, x, m+1 )  
    }  
}
```

พารามิเตอร์คือ
สถานะของการค้น



Sum of Subset Problem : DFS (Iterative)

```
subsetSum( d[1..n] , k )
    s ← an empty stack;
    S.push( a zero-length array )
    while ( s ≠ ∅ ) {
        x[1..m] ← s.pop()
        if ( m == n ) {
            if (sum(d, x) == k) {
                print(x);
            }
        } else {
            x0 = copyOf(x, m+1); x0[m+1] = 0
            x1 = copyOf(x, m+1); x1[m+1] = 1
            S.push(x1);
            S.push(x0);
        }
    }
}
```

แทนสถานะด้วย array ที่เก็บใน stack

สร้างอาร์เรย์ขนาด $m+1$ มี
ข้อมูลเหมือนกับของ x

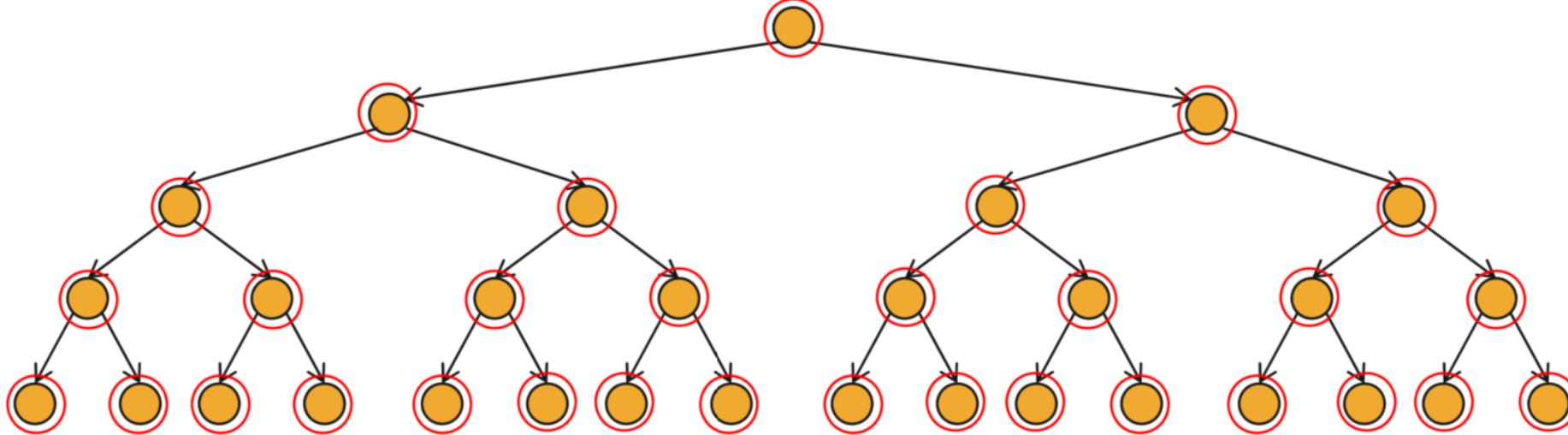
Sum of Subset Problem : BFS (Iterative)

```
subsetSum( d[1..n] , k )
    S ← an empty stack;
    S.push( a zero-length array )
    while ( S ≠ ∅ ) {
        x[1..m] ← S.pop()
        if ( m == n ) {
            if (sum(d, x) == k) {
                print(x);
            }
        } else {
            x0 = copyOf(x, m+1); x0[m+1] = 0
            x1 = copyOf(x, m+1); x1[m+1] = 1
            S.push(x1);
            S.push(x0);
        }
    }
}
```

สร้างอาร์เรย์ขนาด $m+1$ มี
ข้อมูลเหมือนกับของ x

แทนสถานะด้วย array ที่เก็บใน stack

ปริมาณหน่วยความจำ : DFS vs BFS

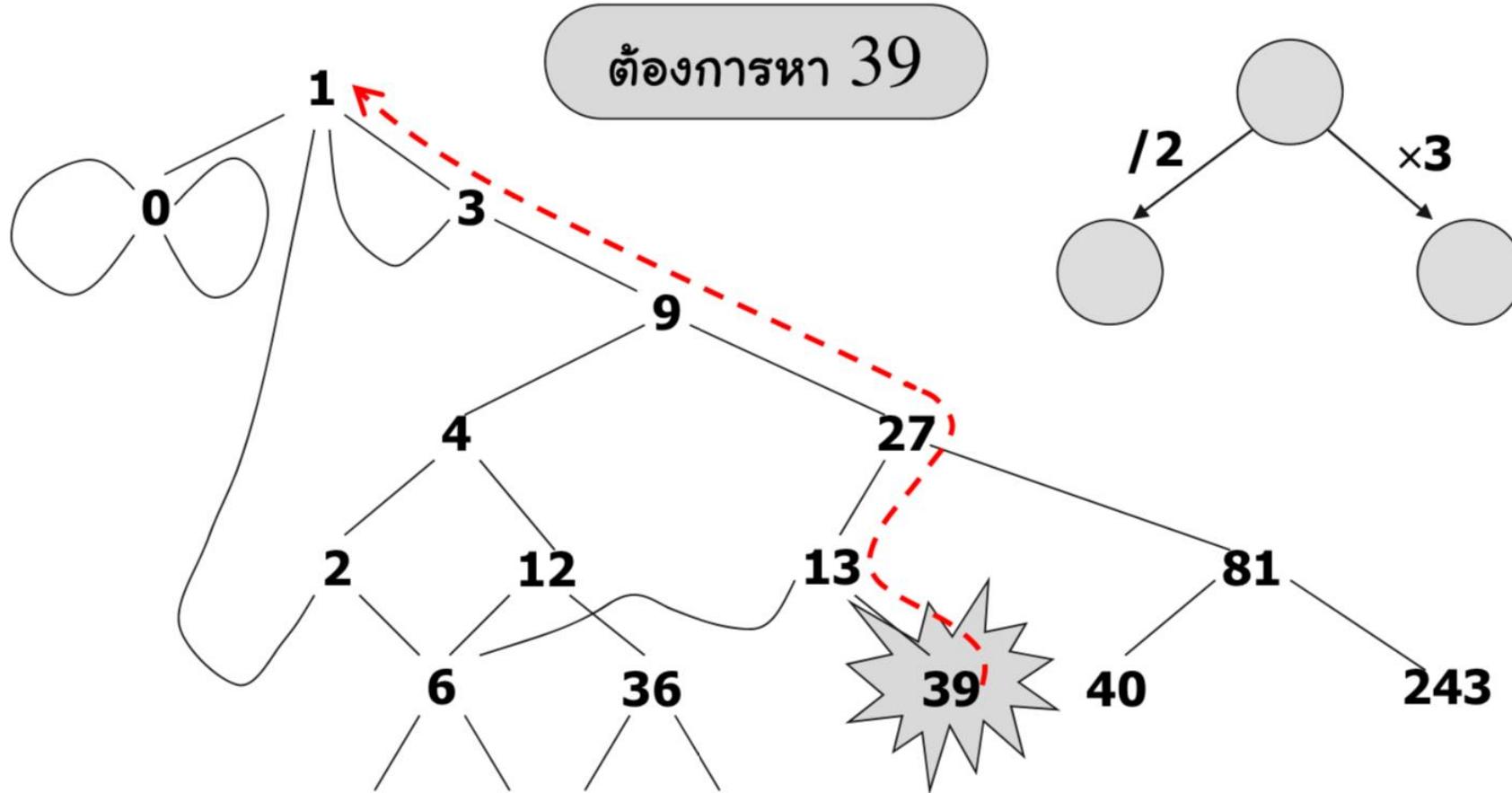


- ❖ 1 ปมแตก b กิ่ง ต้นไม้มีสูง h
- ❖ **depth-first search**
 - ❖ ใช้ stack เนื้อที่ $O(bh)$
- ❖ **breadth-first search**
 - ❖ ใช้ queue เนื้อที่ $O(b^h)$

ปัญหาคูณสามหารสอง

- ❖ ให้จำนวนเต็ม v
- ❖ เริ่มด้วย 1 จะต้อง $\times 3$ และหรือ $/2$ (ปิดเศษทิ้ง)
อย่างไร จึงมีค่าเท่ากับ v
- ❖ เช่น
 - ❖ $v = 10 = 1 \times 3 \times 3 \times 3 \times 3 / 2 / 2 / 2$
 - ❖ $v = 31 = 1 \times 3 \times 3 \times 3 \times 3 / 2 / 2 / 2 / 2 \times 3 \times 3 / 2$
- ❖ ขอถยัหุกรูปแบบตามแนวกว้าง

ปัญหาคูณสามหารสอง : BFS



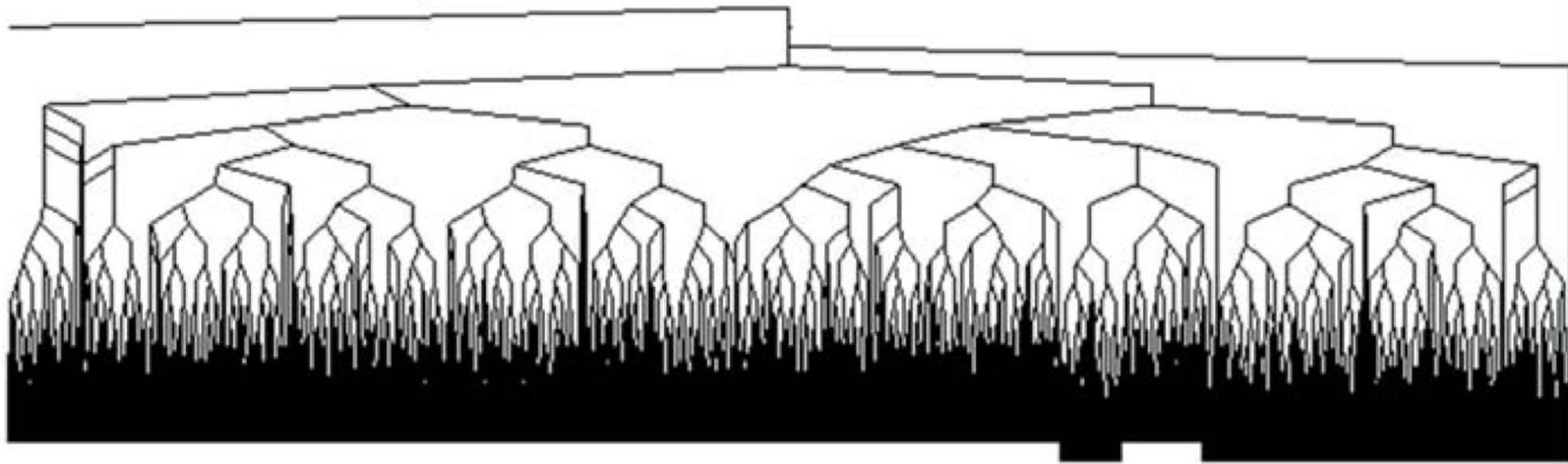
$$39 = 1 \times 3 \times 3 \times 3 / 2 \times 3$$

—————
BFS : สั้นสุด

ប័ណ្ណកូនសាមគរសង : Pseudo code

```
m3d2( target ) {  
    set = an empty set  
    Q = an empty queue  
    state = new State(1);  
    Q.enqueue( state ); set.add( state )  
    while ( Q ≠ ∅ ) {  
        state = Q.dequeue();  
        if (state.value == target) return state;  
        d2 = new State(state.value/2, state);  
        if (d2 ∈ set) {Q.enqueue( d2 ); set.add( d2 )}  
        m3 = new State(state.value*3, state);  
        if (m3 ∈ set) {Q.enqueue( m3 ); set.add( m3 )}  
    }  
    return ∅  
}
```

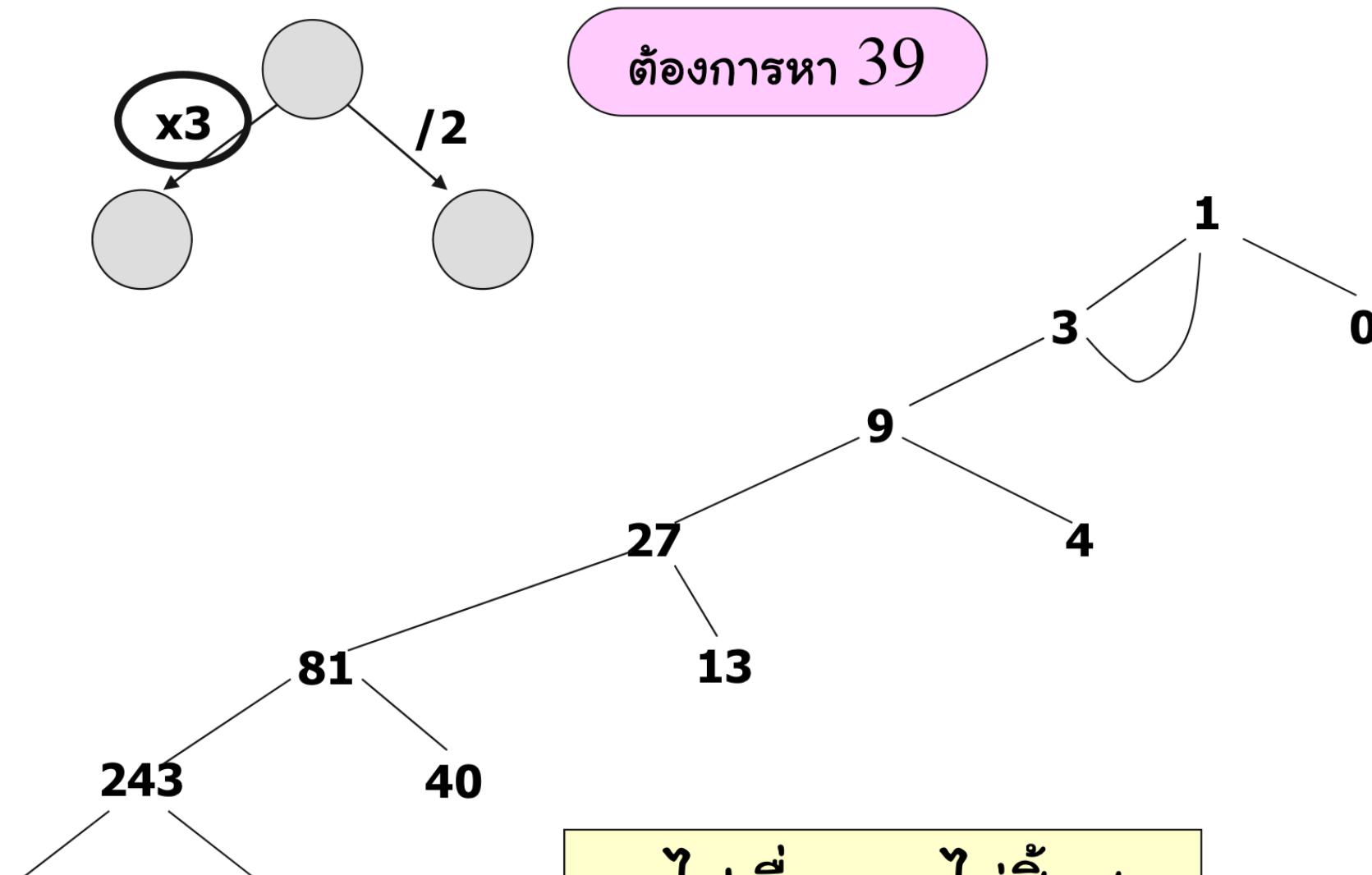
ปัญหาคูณสามหารสอง 41 : BFS



การค้นในแนวกว้างได้ต้นไม้ 12628 ปม สูง 23

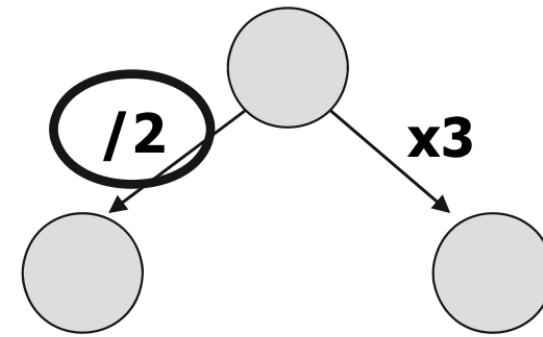
$$\begin{aligned} 41 = & \ 1 \times 3 \times 3 \times 3 / 2 \times 3 \times 3 / 2 / 2 \times 3 \times 3 \\ & / 2 / 2 \times 3 \times 3 / 2 / 2 / 2 \times 3 \times 3 / 2 / 2 / 2 / 2 \end{aligned}$$

ปัญหาคุณสมาร์ทสอง : DFS อาจไม่เจอคำตอบ

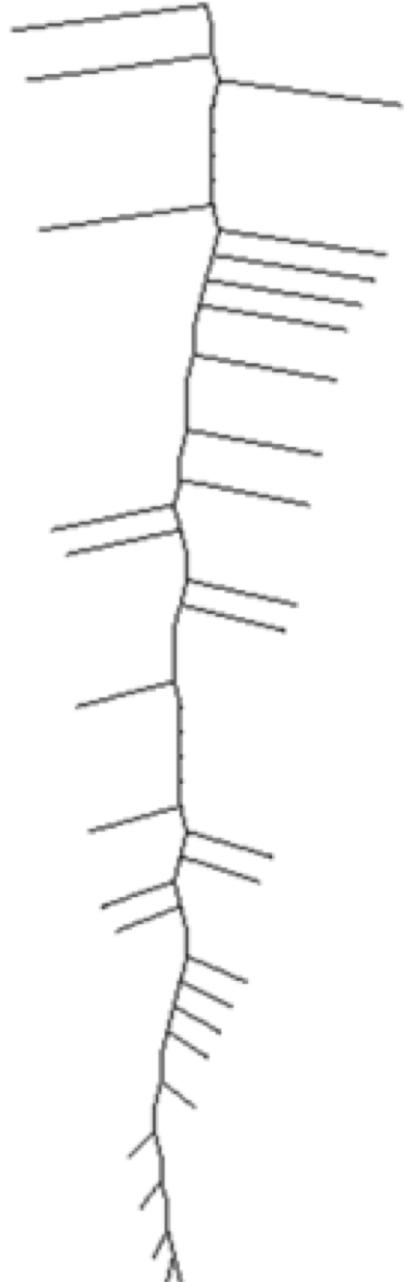


ប័ណ្ណការណីសាមសារសង់ : DFS គិតរវិនិទ្ទេករកនៅ

```
m3d2( target ) {  
    set = an empty set  
    S = an empty stack  
    state = new State(1);  
    S.push( state ); set.add( state )  
    while ( S ≠ ∅ ) {  
        node = S.pop();  
        if (state.value == target) return state;  
        m3 = new State(state.value*3, state);  
        if (m3 ∉ set) {S.push( m3 ); set.add( m3 )}  
        d2 = new State(state.value/2, state);  
        if (d2 ∉ set) {S.push( d2 ); set.add( d2 )}  
    }  
    return ∅  
}
```



ปัญหาคุณสามารถ思考 41 : DFS vs BFS



DFS : ได้ต้นไม้ 82 ปม สูง 51

$$\begin{aligned}41 &= 1 \times 3 \times 3 / 2 / 2 \times 3 \times 3 \times 3 \times 3 / 2 / 2 \\&\quad / 2 / 2 / 2 \times 3 / 2 \times 3 \times 3 / 2 \times 3 / 2 \\&\quad / 2 / 2 \times 3 / 2 / 2 \times 3 \times 3 / 2 \times 3 \times 3 \\&\quad \times 3 \times 3 / 2 / 2 / 2 / 2 \times 3 / 2 / 2 \\&\quad / 2 / 2 \times 3 / 2 \times 3 / 2 \times 3 / 2 \times 3 / 2 / 2\end{aligned}$$

BFS : ได้ต้นไม้ 12628 ปม สูง 23

$$41 = 1 \times 3 \times 3 \times 3 / 2 \times 3 \times 3 / 2 / 2 \times 3 \times 3 \\ / 2 / 2 \times 3 \times 3 / 2 / 2 / 2 \times 3 \times 3 / 2 \\ / 2 / 2 / 2$$

State space search : DFS vs BFS

❖ **depth-first search** ใช้ stack

- ❖ ถ้าต้นไม้มีขนาดไม่จำกัด, อาจหาไม่พบคำตอบ
- ❖ ต้นไม้สูง h , จะใช้เนื้อที่ใน stack แค่ $O(b \cdot h)$

❖ **breadth-first search** ใช้ queue

- ❖ ประกันว่าพบ answer state ที่ใกล้รากสุด
- ❖ ต้นไม้สูง h , จะใช้เนื้อที่ใน queue เป็น $O(b^h)$

❖ **state space tree** มีขนาดใหญ่, ซึ่มมาก ๆ

Backtracking:



The word "Backtracking:" is written in a large, bold, black sans-serif font. A large black question mark symbol is positioned above the text. Inside the question mark, there is a black checkmark symbol (\checkmark) and a black asterisk symbol ($*$). Ellipsis dots (\dots) are placed at the end of the word.

By Payongkit XI

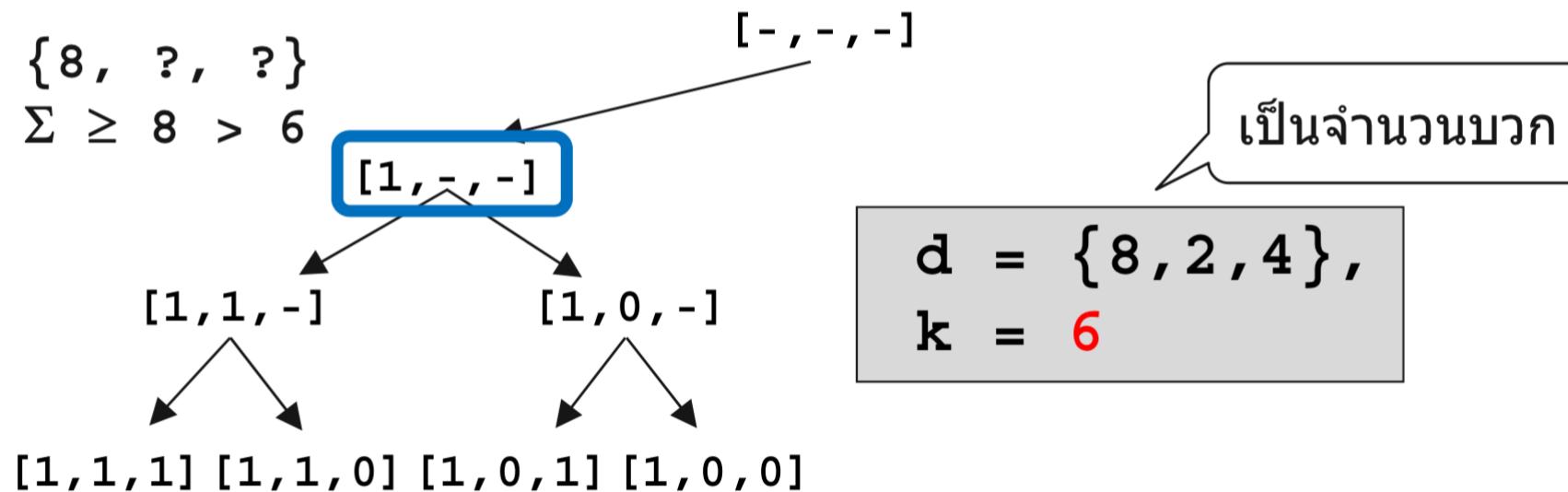
Backtracking (การย้อนรอย) algorithm

เป็นขั้นตอนวิธีพิจารณาค้นทุกทางที่เป็นไปได้เพื่อหาคำตอบของปัญหา โดยการค้นหาคำตอบไปทีละส่วน (partial solution) เมื่อแน่ใจว่าผลเฉลยที่หาได้นั้นเป็นส่วนหนึ่ง ของคำตอบ จะพยายามหาคำตอบส่วนอื่นต่อไป เพื่อให้ได้คำตอบที่สมบูรณ์เมื่อได้พบร่วมกันแล้ว ผลเฉลยส่วนที่หาได้ไม่ใช่คำตอบของปัญหา จะทำการย้อนรอยถอยหลัง (backtrack) โดยการถอนผลเฉลยในขณะนั้นออกแล้ว พยายามหาคำตอบใหม่โดยเลือกดำเนินการกับทางเลือกอื่นที่เป็นไปได้ต่อไป

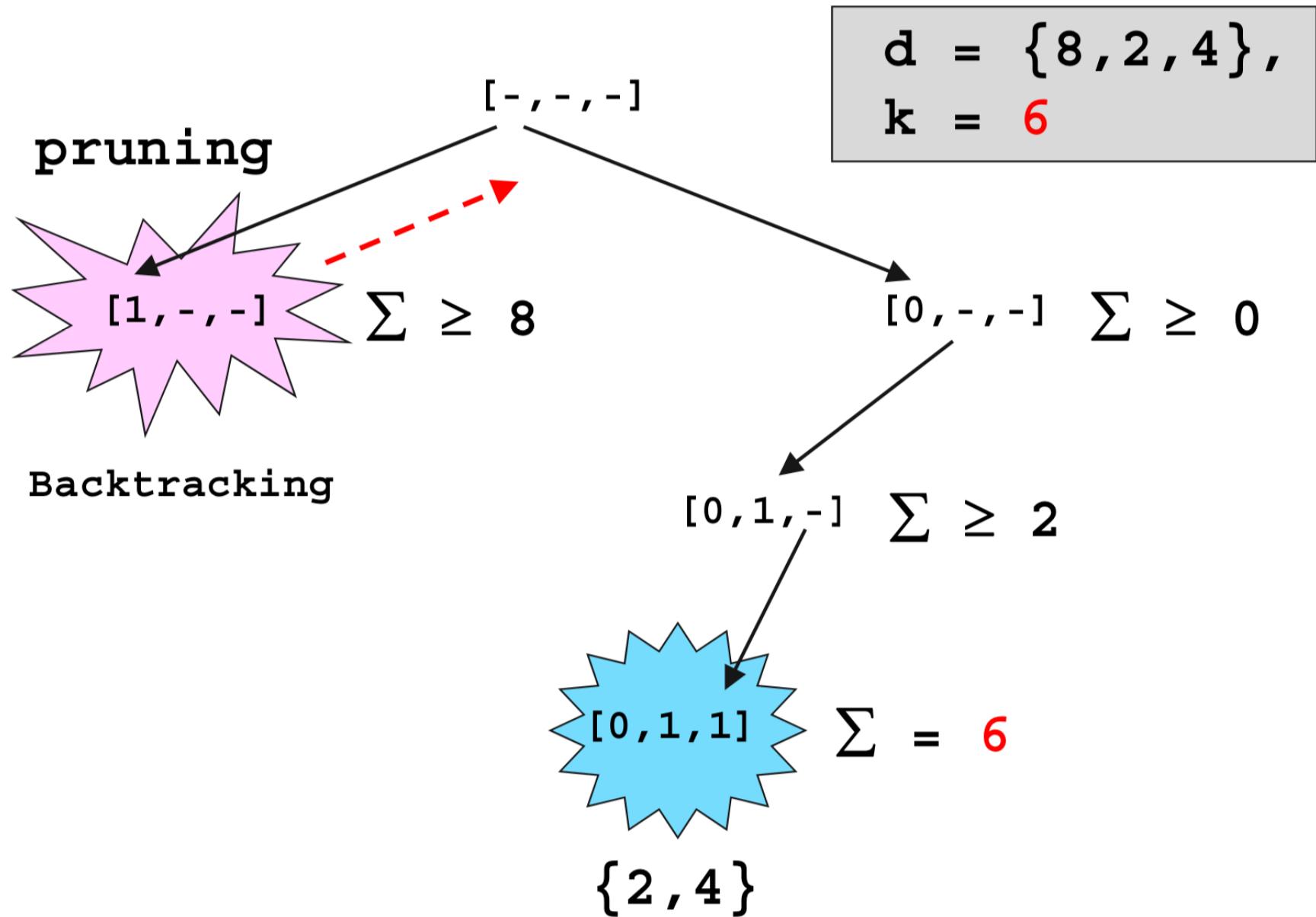
Backtracking algorithm เป็นขั้นตอนวิธี (วิธีการแก้ปัญหา) ที่เหมาะสมกับภาระที่ในตอนเริ่มต้นดูเหมือนจะมีแนวทางแก้ปัญหาที่เป็นได้มากหลายแบบ แต่เมื่อเลือกแนวทางเหล่านั้นมา ทำการทดสอบจะพบว่าแนวทางส่วนใหญ่ไม่สามารถหาคำตอบได้คงมีเฉพาะบางแนวทางเท่านั้นที่สามารถใช้หาคำตอบได้

Backtracking (การย้อนรอย)

- ❖ ค้นทุกปมในปริภูมิสถานะ → ข้า
- ❖ พิจารณาสถานะที่แทนผลเฉลยบางส่วน (partial solution state) ถ้าไม่มีเวว ไม่ค้นต่อ
- ❖ ไม่มีเวว → มันใจว่าไม่พบสถานะคำตอบแน่ ๆ



Sum of subset : Backtracking



Sum of subset : DFS+Backtracking

```
subsetSum( d[1..n] , k, x[1..n] , m ) {  
    sum = sum(d, x, m)  
    if (sum > k) return  
    if (m == n) {  
        if (      sum == k      ) print( x )  
    } else {  
        x[m+1] = 1; subsetSum( d, k, x, m+1 )  
        x[m+1] = 0; subsetSum( d, k, x, m+1 )  
    }  
}
```

$$\text{sum}(d, x, m) = \sum_{j=1}^m x[j]d[j]$$

Sum of subset : DFS+Backtracking

```
subsetSum( d[1..n] , k, x[1..n] , m ) {  
    sum = sum(d, x, m)  
    if (sum > k) return  
    if (m == n) {  
        if ( sum == k ) print( x )  
    } else {  
        if ( sum+d[m+1] <= k )  
            x[m+1] = 1; subsetSum( d, k, x, m+1 )  
        x[m+1] = 0; subsetSum( d, k, x, m+1 )  
    }  
}
```

$$\text{sum}(d, x, m) = \sum_{j=1}^m x[j] d[j]$$

Sum of subset : BFS+Backtracking

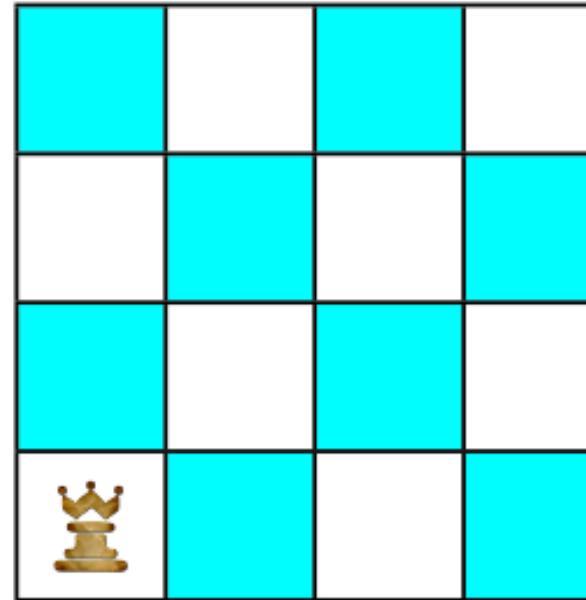
```
subsetSum( d[1..n] , k )
    Q ← an empty queue;
    Q.enqueue( a zero-length array )
    while ( Q ≠ ∅ ) {
        x[1..m] ← Q.dequeue()
        sum = sum(d, x, m)
        if (sum <= k) {
            if ( m == n ) {
                if (      sum == k      ) print(x)
            } else {
                x0 = copyOf(x, m+1); x0[m+1] = 0
                x1 = copyOf(x, m+1); x1[m+1] = 1
                Q.enqueue(x0);
                Q.enqueue(x1);
            }
        }
    }
```

Sum of subset : Backtracking

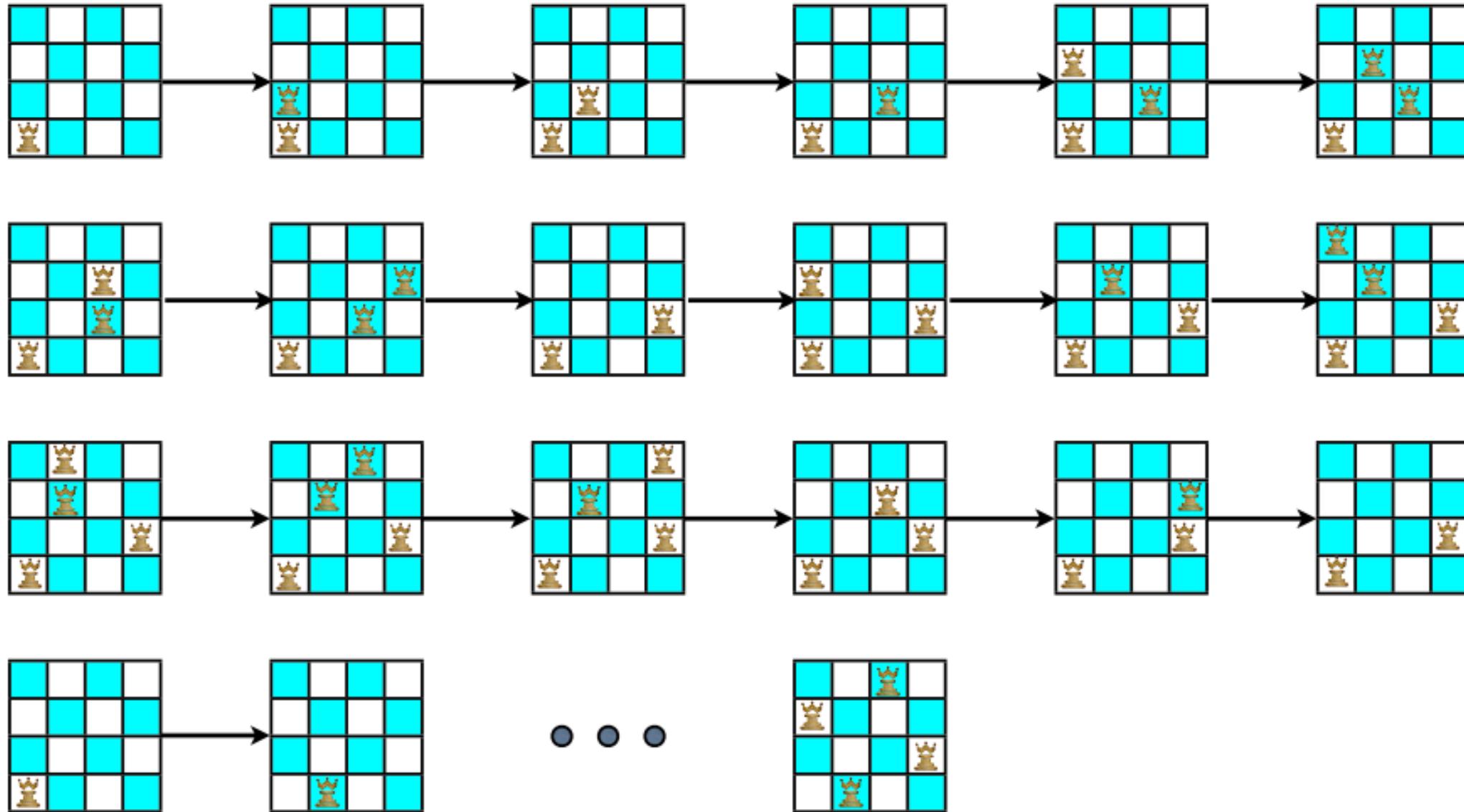
n	จำนวนคำตอบ	DFS (จำนวนปมที่ค้น)		DFS + Backtracking (จำนวนปมที่ค้น)	
		คำตอบแรก	คำตอบทั้งหมด	คำตอบแรก	คำตอบทั้งหมด
10	6	1,402	2,047	133	384
11	20	621	4,095	97	1,934
12	37	708	8,191	78	3,660
13	36	1,922	16,383	144	4,100
14	81	3,967	32,767	26	8,198
15	204	1,797	65,535	60	27,351
16	128	4,085	131,071	41	20,015
17	616	7,684	262,143	76	70,795
18	1,052	16,354	524,287	74	121,853
19	2,515	6,649	1,048,575	120	360,480
20	7,133	6,152	2,097,151	21	949,847
21	6,837	30,472	4,194,303	97	789,842
22	8,979	442,309	8,388,607	139	915,099
23	50,129	14,090	16,777,215	73	7,145,916
24	92,457	24,456	33,554,431	25	12,847,798
25	54,493	1,015,288	67,108,863	234	6,159,965

N-Queen problem

- ต้องการวาง Queen จำนวน n ตัว บนกระดานหมากรุกที่มีขนาด $n \times n$ (ซึ่งจะมีจำนวนของช่องสี่เหลี่ยมทั้งหมด n^2) โดยไม่ให้ Queen แต่ละตัวกินกันได้
- จะต้องวาง Queen ไว้ในตำแหน่งใดบ้าง



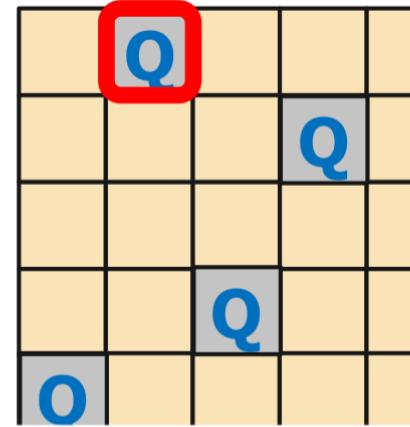
N-Queen problem : state space



N-Queen problem : ใช้อาร์ย 1 มิติ แทนการวางqueen

แถวที่ 1 วางคอลัมน์ที่ 2

col	1	2	3	4	...
1	2				
2		4			
3			9		
4				3	
5					1
.					

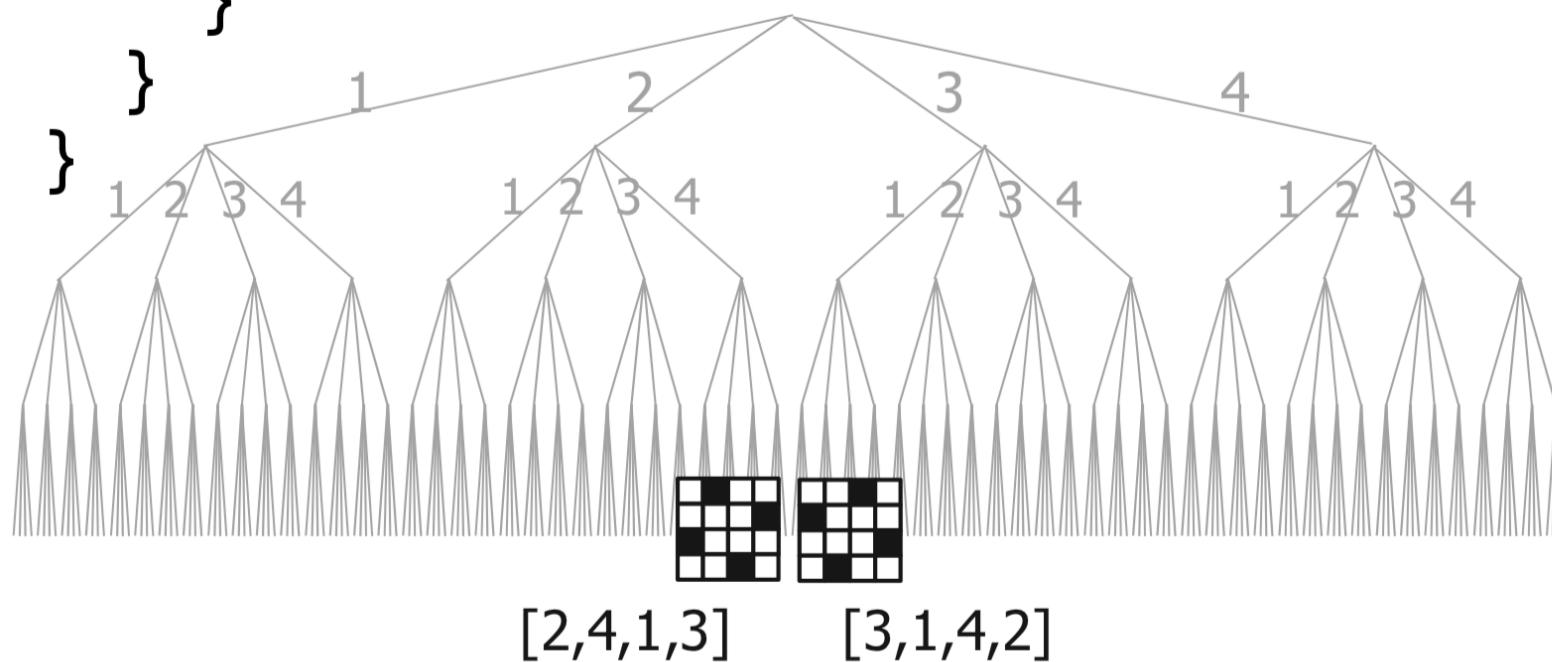


col [r] แทนหมายเลขคอลัมน์ที่วางควีนของแถวที่ r

[2, 4, 9, 3, 1, ...]

N-Queen problem : DFS

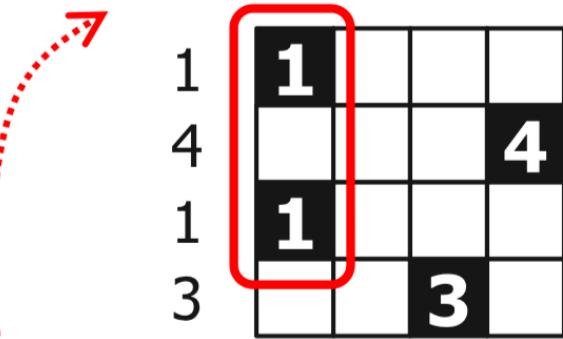
```
queen(col[1..n], m) {  
    if (m == n) {  
        if (isValid(col)) print(col)  
    } else {  
        for (i = 1; i <= n; i++) {  
            col[m+1] = i; queen(col, m+1);  
        }  
    }  
}
```



```
[1, 1, 1, 1]  
[1, 1, 1, 2]  
[1, 1, 1, 3]  
[1, 1, 1, 4]  
[1, 1, 2, 1]  
[1, 1, 2, 2]  
[1, 1, 2, 3]  
[1, 1, 2, 4]  
  
...  
  
[4, 4, 3, 1]  
[4, 4, 3, 2]  
[4, 4, 3, 3]  
[4, 4, 3, 4]  
[4, 4, 4, 1]  
[4, 4, 4, 2]  
[4, 4, 4, 3]  
[4, 4, 4, 4]
```

N-Queen problem : การวาง queen ที่ผิดกฎหมาย

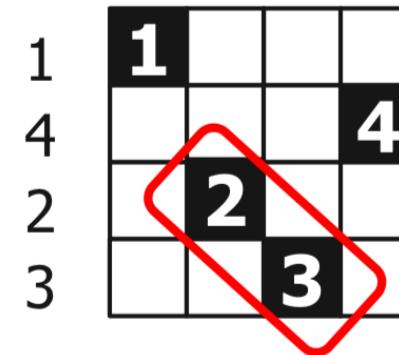
[1,4,1,3]



คอลัมน์เดียวกัน

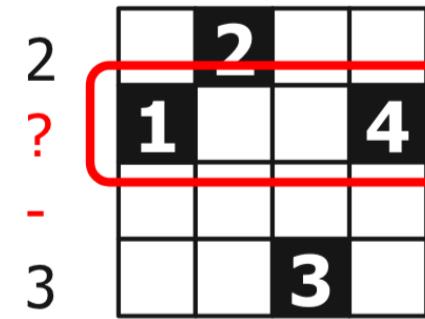
มีตัวเลขซ้ำกัน
หรือไม่ ?

[1,4,2,3]



แนวทแยงเดียวกัน

[2,?,-,3]



ไม่เกิด

[1, 1, 1, 1]

[1, 1, 1, 2]

[1, 1, 1, 3]

...

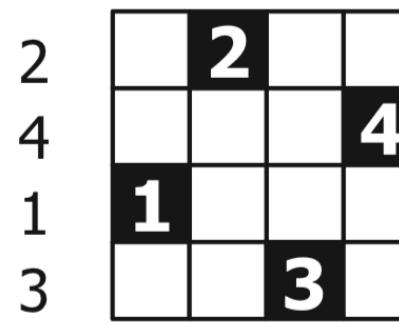
[4, 4, 4, 1]

[4, 4, 4, 2]

[4, 4, 4, 3]

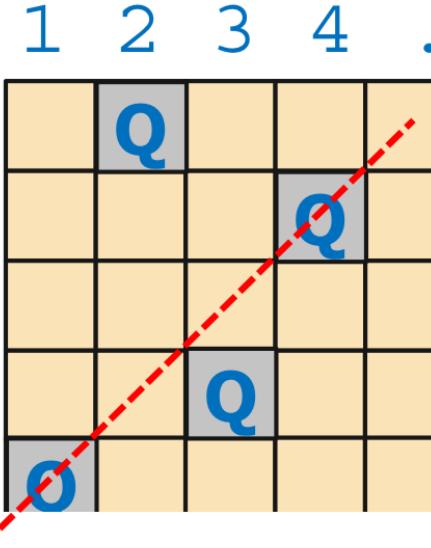
[4, 4, 4, 4]

[2,4,1,3]



N-Queen problem : การตรวจสอบในแนวเส้นทแยงมุม

col	1	2	3	4	...
1	2				
2		4			
3			9		
4				3	
5					1



. **col[r]**

แทนหมายเลขคอลัมน์
ที่วางควีนของقاءที่ **r**

$$\begin{aligned} |1 - 2| &\neq |2 - 4| \\ |1 - 3| &\neq |2 - 9| \\ |1 - 4| &\neq |2 - 3| \\ |1 - 5| &\neq |2 - 1| \end{aligned}$$

...

$$\begin{aligned} |2 - 3| &\neq |4 - 9| \\ |2 - 4| &\neq |4 - 3| \\ |2 - 5| &= |4 - 1| \end{aligned}$$



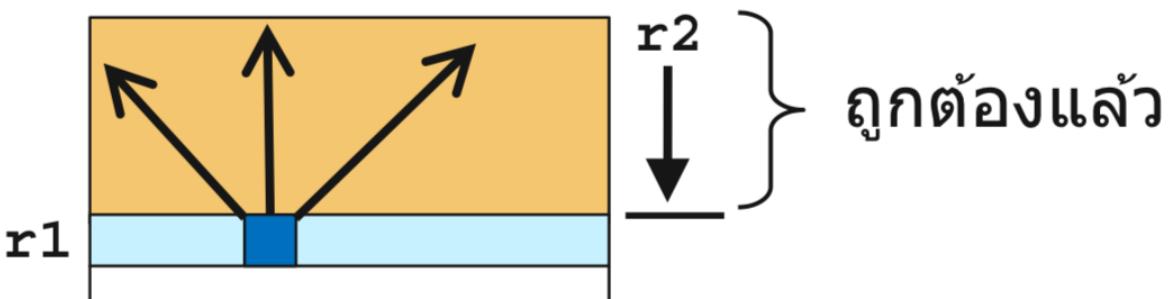
...

ควีนของقاء r_1 กับ r_2 จะไม่เห็นกันแนวทแยง ถ้า

$$|r_1 - r_2| \neq |\text{col}[r_1] - \text{col}[r_2]|$$

N-Queen problem : การตรวจสอบ N queen ที่ถูกต้อง

```
isValid(col[1..n]) {  
    for (r1 = 2; r1 <= n; r1++) {  
        for (r2 = 1; r2 < r1; r2++) {  
            if ( col[r1] == col[r2] ||  
                |col[r1] - col[r2]| == r1 - r2 )  
                return false  
        }  
        return true;  
    }  
}
```



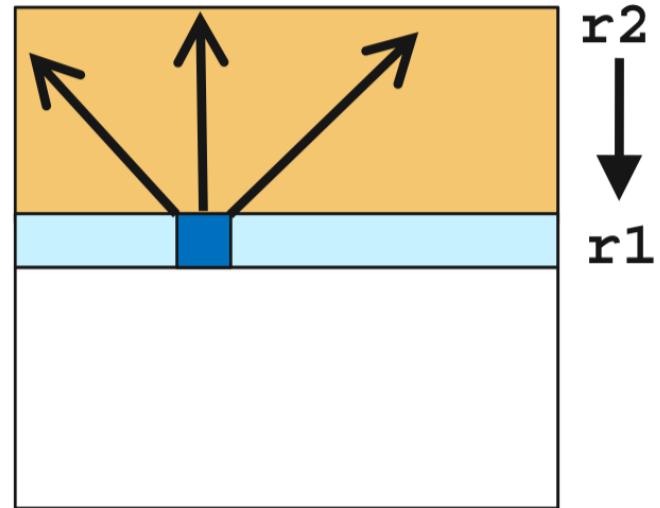
คุณของแกร r_1 กับ r_2 จะไม่เห็นกันแนวทแยง ถ้า

$$| r_1 - r_2 | \neq | \text{col}[r_1] - \text{col}[r_2] |$$

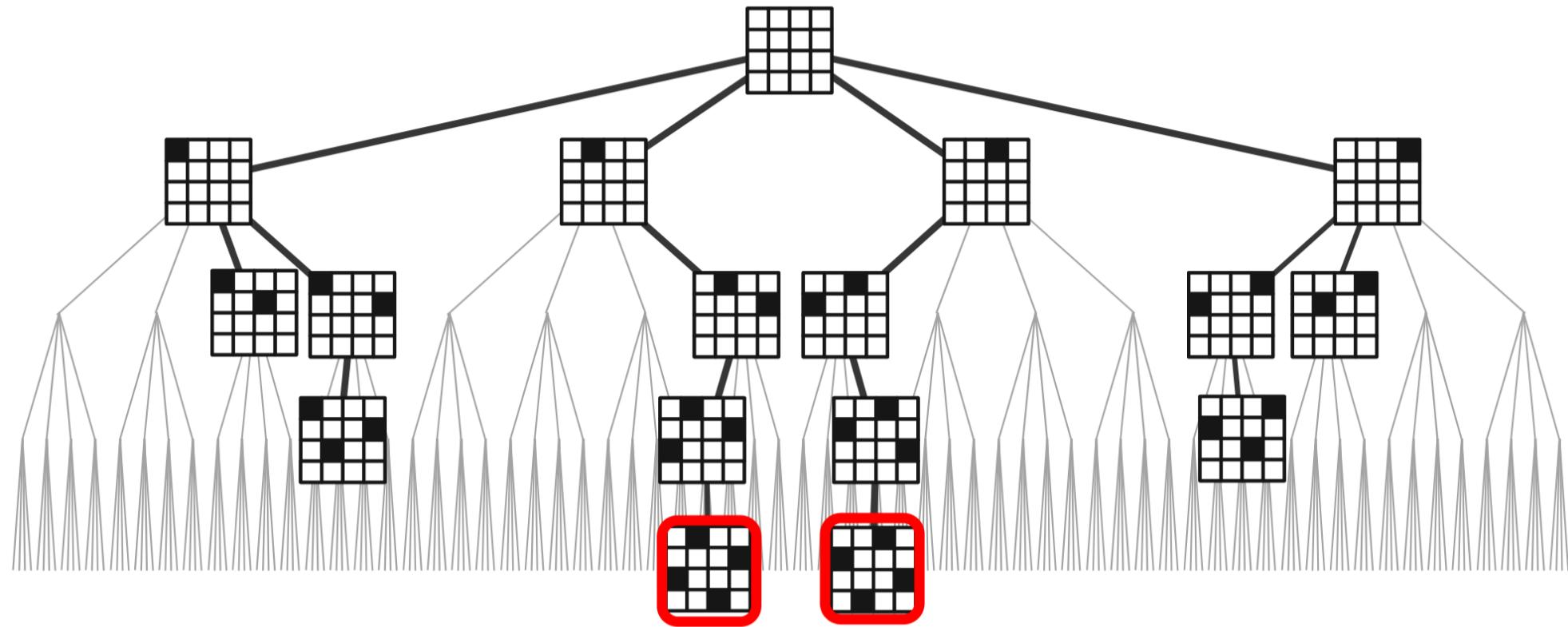
N-Queen problem : pseudo code DFS+Backtracking

```
queen(col[1..n], m) {  
    if (m == n) {  
        if (isValid(col)) print(col)  
    } else {  
        for (i = 1; i <= n; i++) {  
            col[m+1] = i  
            if (isValid(col, m+1))  
                queen(col, m+1)  
        }  
    }  
}
```

```
isValid(col[1..n], k) {  
    r1 = k  
    for (r2 = 1; r2 < r1; r2++) {  
        if ( col[r1] == col[r2] ||  
            |col[r1] - col[r2]| == r1 - r2 )  
            return false  
    }  
    return true;  
}
```



N-Queen problem : DFS+Backtracking



N-Queen problem :

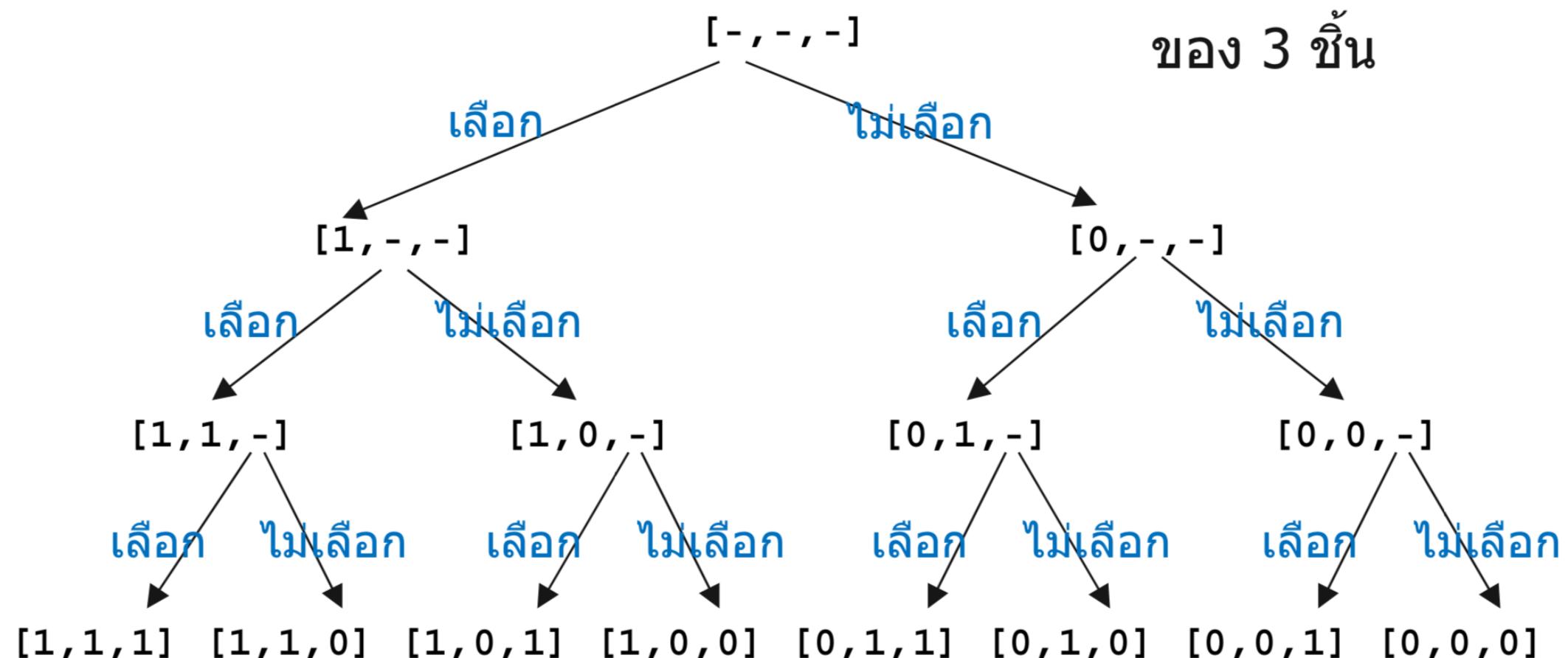
n	จำนวน คำตอบ	จำนวนการ เรียกซ้ำจนพบ คำตอบแรก
4	2	9
5	10	6
6	4	32
7	40	10
8	92	114
9	352	42
10	724	103
11	2,680	53
12	14,200	262
13	73,712	112
14	365,596	1,900
15	2,279,184	1,360
16	14,772,512	10,053
17	95,815,104	5,375

0/1 Knapsack problem :

- ❖ ของ n ชิ้นมีหมายเลข : $1, 2, 3, \dots, n$
- ❖ แต่ละชิ้นหนัก : $w_1, w_2, w_3, \dots, w_n$
- ❖ แต่ละชิ้นมีมูลค่า : $v_1, v_2, v_3, \dots, v_n$
- ❖ ถุงเป้หนึ่งใบจุของได้หนักไม่เกิน W
- ❖ ปัญหา : จงเลือกของใส่ถุง เพื่อให้
 - ❖ ถุงไม่ขาด
 - ❖ ได้มูลค่ารวมมากสุด
- ❖ หา $\langle x_1, x_2, x_3, \dots, x_n \rangle$,
 $x_k = 0$ หรือ 1

$$\begin{aligned} & \text{maximize} \sum_{k=1}^n x_k v_k \\ & \text{subject to} \sum_{k=1}^n x_k w_k \leq W \\ & x_k \in \{0,1\} \end{aligned}$$

0/1 Knapsack problem : state space



0/1 Knapsack problem : pseudo code

```
knapsack( w[1..n] , v[1..n] , W ) {
    x = new array[1..n] with all 0's
    vmax = -∞; xmax = x
    knapsack( w, v, W, x, 0 )
    return xmax
}
knapsack(w[1..n], v[1..n], W, x[1..n], m) {

    if (m == n) {
        if (sum(w,x) <= W AND sum(v,x) > vmax) {
            vmax = sum(v, x); xmax = x
        }
    } else {
        x[m+1] = 1; knapsack( w, v, W, x, m+1 )
        x[m+1] = 0; knapsack( w, v, W, x, m+1 )
    }
}
```

$$\text{sum}(w, x) = \sum_{j=1}^n w[j]x[j]$$

$$\text{sum}(v, x) = \sum_{j=1}^n v[j]x[j]$$

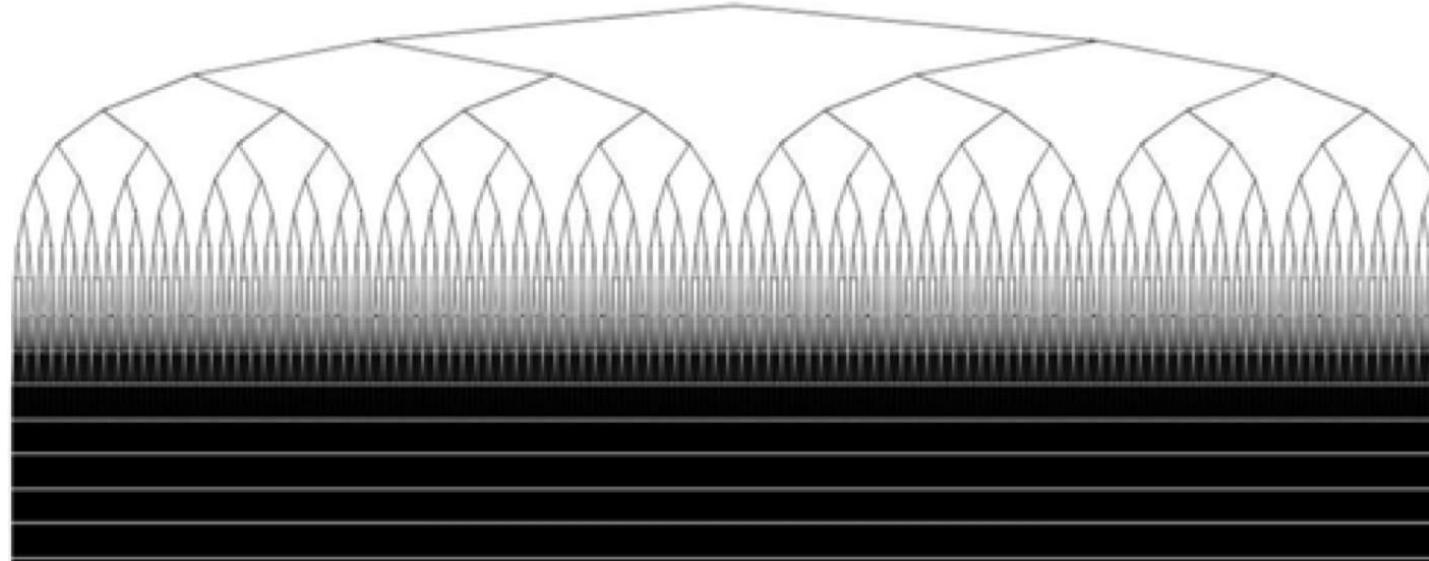
0/1 Knapsack problem : Backtracking

```
knapsack( w[1..n], v[1..n], W ) {
    x = new array[1..n] with all 0's
    vmax = -∞; xmax = x
    knapsack( w, v, W, x, 0 )
    return xmax
}

knapsack(w[1..n], v[1..n], W, x[1..n], m) {
    if (sum(w, x, m) > W) return
    if (m == n) {
        if (sum(w,x) <= W AND sum(v,x) > vmax) {
            vmax = sum(v, x); xmax = x
        }
    } else {
        x[m+1] = 1; knapsack( w, v, W, x, m+1 )
        x[m+1] = 0; knapsack( w, v, W, x, m+1 )
    }
}
```

$$\text{sum}(w, x, m) = \sum_{j=1}^m w[j]x[j]$$

0/1 Knapsack problem :



เต็มตัน

เพิ่มการ
ย้อนรอย

