บทที่ 5

การย้อนรอย (back tracking)

การขยายและจำกัดเขต(Branch and Bound)

การย้อนรอย เป็นเทคนิคที่เกี่ยวข้องกับการเรียกซ้ำและลำดับของการเดาซึ่งนำไปสู่ผลลัพธ์ หากการเดาดังกล่าวนำไปสู่ทางตัน คุณจะต้องถอยกลับไปยังจุดก่อนหน้า และทำการเดาอีกครั้ง และทำดังกล่าวต่อไปเรื่อย ๆ จนกระทั่งได้ผลลัพธ์

การย้อนรอยมักใช้ในการแก้ปัญหาเกี่ยวกับการเลือกสิ่งของจากเซ็ตที่กำหนดให้ ให้สอดคล้องกับเงื่อนไขบางสิ่งบางอย่าง การแก้ปัญหาจะมีการแตกกิ่งก้านสาขาในลักษณะต้นไม้ ซึ่งเราจะเรียกต้นไม้ดังกล่าวว่า state space tree และทำการสืบค้นในแนวดิ่ง (depth first search)

การย้อนรอยจะเป็นการสืบค้นอย่างถ้วนทั่ว โดยเราจะได้ผลลัพธ์เริ่มต้นที่ไม่ดีนัก เมื่อใช้เวลาในการสืบค้นเพิ่มขึ้น จะทำให้ได้ผลลัพธ์ที่ดีขึ้น และเมื่อสืบค้นจนถ้วนทั่วจะได้ผลลัพธ์ที่ดีที่สุด ขณะที่ทำการสืบค้นก็จะทำการลดขนาด solution space ให้เล็กลง โดยทำการตัดการแตกกิ่งก้านสาขาเมื่อพบโหนดซึ่งไม่นำไปสู่ผลลัพธ์ หรือนำไปสู่ผลลัพธ์ที่ไม่ดีขึ้น การตัดการแตกกิ่งก้านสาขานี้ เรียกว่า pruning

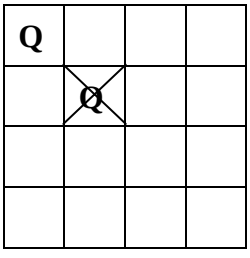
สำหรับเทคนิควิธีการขยายและจำกัดเขต คล้ายกับวิธีการย้อนรอยตรงที่มีการสร้าง state space tree เพื่อแก้ปัญหา แต่ต่างกันที่วิธีการย้อนรอยจะท่องในต้นไม้ (tree traversal) ในแนวลึกก่อน(depth first search) ในขณะที่เทคนิควิธีการขยายและจำกัดเขต จะท่องในต้นไม้ในแนวนอน (breadth first search) ใช้ขั้นตอนวิธีแบบ queue แทนที่จะใช้ recursive ใน back tracking

ตัวอย่างการใช้การย้อนรอยในการแก้ปัญหา n-Queen ปัญหา 0/1 แน็บแซ็ก (0/1 knapsack problem) และปัญหาเซลแมน (traveling salesman problem)

5.1 ปัญหา n-Queen

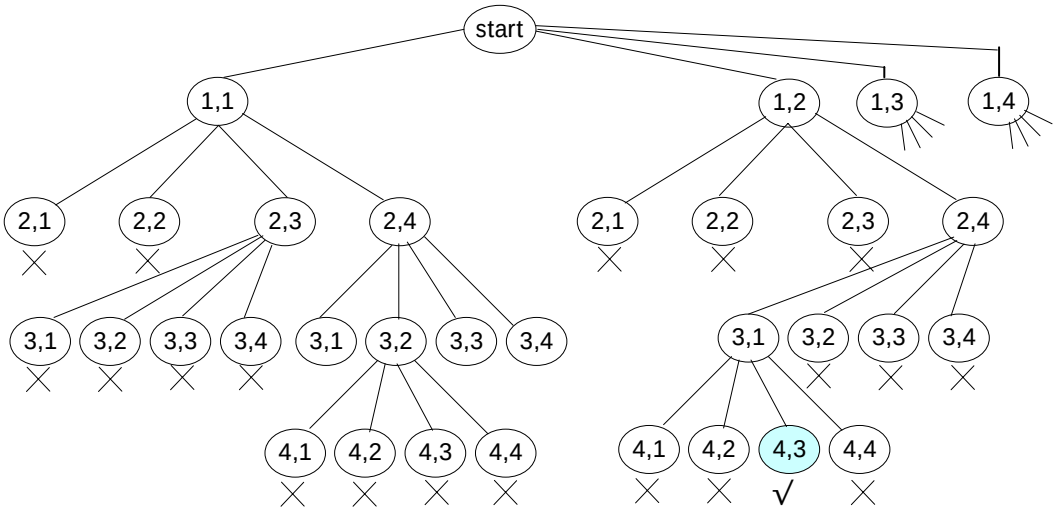
ปัญหา n-Queen เป็นปัญหาการวาง Queen จำนวน n ตัวอย่างไรในตารางหมากรุกขนาด n x n จึงจะทำให้ไม่ทำร้ายซึ่งกันและกัน โดย Queen มีอำนาจทำร้ายได้ 8 ทิศรอบตัว นั่นคือแนวดิ่ง แนวนอน และแนวทแยงมุม

ตัวอย่าง 5.1 จงแก้ปัญหา 4-Queen



วิธีทำ จะเริ่มพิจารณาวาง Queen ตัวแรกในตำแหน่ง (1,1) ก่อน และพิจารณาวาง Queen ตัวที่สอง จะพบว่าไม่สามารถวางในตำแหน่ง (2,1) และ (2,2) ได้เนื่องจากอยู่ในแนวดิ่งและแนวทแยงของ Queen ตัวแรก จึงวาง Queen ตัวที่สองลงในตำแหน่ง (2,3) และพิจารณาวาง Queen ตัวที่สาม จะพบว่าไม่สามารถวางในตำแหน่ง (3,1) ได้เนื่องจากอยู่ในแนวดิ่งของ Queen ตัวแรก และไม่สามารถวางในตำแหน่ง (3,2), (3,3) และ (3,4) ได้เนื่องจากอยู่ในแนวทแยงซ้าย แนวดิ่ง และแนวทแยงขวาของ Queen ตัวที่สอง เนื่องจากไม่สามารถวาง Queen ตัวที่สามในแถวที่ 3 ได้ทำให้ไม่สามารถวาง Queen ครบทั้งสี่ตัวได้ จึงย้อนรอยกลับไปพิจารณาวาง Queen ตัวที่สอง ณ ตำแหน่ง (2, 4) และทำการพิจารณาไปเรื่อย ๆ จนได้ผลลัพธ์ ดังแสดงในรูป 5.1

หมายเหตุ ตำแหน่ง (i, j) หมายถึง แถวที่ i คอลัมภ์ที่ j



รูป 5.1 แสดง state space tree ของปัญหา 4-Queen

จากรูป5.1 จะได้ตำแหน่งในการวาง Queen เป็น (1, 2), (2, 4) (3,1) และ (4,3) ดังรูป

	Q		
			Q
Q			
		Q	

Algorithm1: ขนาดของ solution space = 1 โหนดที่ level 0, ที่ level 1 มี solution space = n โหนด, ที่ level 2 มี solution space = n^2 โหนด, ที่ level 3 มี solution space = n^3 โหนดที่ level 3 ไปเรื่อยๆ จนถึง level ที่ n จะได้จำนวนโหนด เป็น

$$1 + n + n^2 + n^3 + n^4 + n^5 + \cdots + n^n = \frac{n^{n+1} - 1}{n - 1}$$

ถ้า n=8 จะได้ $\frac{8^{8+1}-1}{8-1} = 19{,}173{,}961$ โหนด

Algorithm2 : เราสามารถคำนวณจำนวนโหนด ด้วยข้อจำกัดที่ไม่สามารถวางบน column เดียวกันได้ ดังนั้นเมื่อเลือกวาง queen ตัวแรกลงไปในแถวแรก จะสามารถเลือกตำแหน่งได้ 8 column เมื่อจะวาง queen ตัวที่2 ก็จะมีโอกาสเลือก 7 column เมื่อจะวาง queen ตัวที่3 ก็จะมีโอกาสเลือก 6 column ไปเรื่อยๆ ดังนั้น

solution space = 1+8+(8*7)+(8*7*6)+(8*7*6*5)+…+8! = 40,481 โหนด

= 1+n+n(n-1)+n(n-1)(n-2)+…..+n!

เปรียบเทียบขนาด solution space ของอัลกอริทึมที่ 1, 2 และวิธีการย้อนรอยที่จะตัดกิ่งก้านในกรณีถึงทางตัน (pruning) ทำให้เราไม่ต้องเช็คไปจนครบทุกโหนด จะสามารถตัดทอนจำนวนโหนด ลงได้มาก ตามตารางที่ 5.1

ตาราง5.1 เปรียบเทียบขนาด solution space ของอัลกอริทึมที่ 1, 2 และวิธีการย้อนรอย

n	Algorithm1➔ $\frac{n^{n+1}-1}{n-1}$	Algorithm2 ➔1+n+n(n-1)+..+n!	Back tracking
4	341	65	33
8	19,173,961	40,481	8,889
12	9,726,655,034,461	1,302,061,332	2,044,801
14	11,966,776,581,370,200	236,975,164,790	71,656,817

โปรแกรม 5.1 Queens.java

```
public class Queens {
  public static boolean isConsistent(int[] q, int n) {
    for (int i = 0; i < n; i++) {
      if (q[i] == q[n])           return false;   // same column
      if ((q[i] - q[n]) == (n - i)) return false;   // same major diagonal
      if ((q[n] - q[i]) == (n - i)) return false;   // same minor diagonal
    }
    return true;
  }
  public static void printQueens(int[] q) {
    int N = q.length;
    for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
        if (q[i] == j) System.out.print("Q ");
```

```
            else          System.out.print("* ");
         }
         System.out.println();
      }
      System.out.println();
   }
  //  Try all permutations using backtracking
   public static void enumerate(int N) {
      int[] a = new int[N];
      enumerate(a, 0);
   }
   public static void enumerate(int[] q, int n) {
      int N = q.length;
      if (n == N) printQueens(q);
      else {
         for (int i = 0; i < N; i++) {
            q[n] = i;
            if (isConsistent(q, n)) enumerate(q, n+1);
         }
      }
   }
   public static void main(String[] args) {
            int N = Integer.parseInt(args[0]);
      enumerate(N);
   }
}
```

ผลลัพธ์ 4 - Queens

```
* Q * *
* * * Q
Q * * *
* * Q *

* * Q *
Q * * *
* * * Q
* Q * *
total loop =33
```

5.2 ปัญหาถุงเป้ 0-1 (0-1 knapsack problem)

เราเคยวิเคราะห์ปัญหาถุงเป้จากบทที่แล้ว มาในบทนี้เราจะมีวิธีการย้อนรอยกับปัญหาถุงเป้อีกครั้ง มีโจรรายหนึ่งเข้าไปทำการโจรกรรมในพิพิธภัณฑ์แห่งหนึ่ง ซึ่งมีของมีค่ามากมาย โจรรายนี้รู้มูลค่าของสิ่งของ และน้ำหนักของสิ่งของได้เป็นอย่างดี ปัญหามีอยู่ว่าโจรต้องตัดสินใจว่า ควรเลือกหยิบหรือไม่หยิบสิ่งของใดลงในถุงเป้ เพื่อให้มีมูลค่าของสิ่งของมากที่สุด และไม่หนักเกินที่ถุงเป้จะรับได้

ตัวอย่าง 5.2 จงพิจารณาปัญหาถุงเป้ 0-1 โดยมีสิ่งของ 4 ชนิดที่จะพิจารณาบรรจุลงถุงเป้ ให้ w_i แทน น้ำหนักของสิ่งของชนิดที่ i และ p_i แทนมูลค่าของสิ่งของชนิดที่ i

w_i = [2,5,10,7]

p_i = [4000, 6000, 3500, 5000]

โดยถุงเป้มีความจุเป็น 16 กก. และต้องการบรรจุสิ่งของลงในถุงเป้ให้มีมูลค่าสูงสุด จงแสดง State Space tree ที่ใช้วิธีการย้อนรอย ในการตัดผลลัพธ์ที่ไม่นำไปสู่ผลลัพธ์ที่ดีที่สุดออก โดยพิจารณาจาก bounding Function

$$W = 16$$

$$totalweight = weight + \sum_{j=i+1}^{k-1} w_j$$

$$bound = \left(profit + \sum_{j=i+1}^{k-1} p_j \right) + (W - totalweight) \times \frac{p_k}{w_k}$$

โดยที่ W = น้ำหนักสูงสุดที่ถุงเป้จะรับได้

i = ระดับของโหนดที่ยังทำให้ผลรวมน้ำหนักสิ่งของที่หยิบใส่ถุงเป้(weight) $\leq$ W

k = ระดับของโหนดที่เริ่มทำให้ผลรวมน้ำหนักสิ่งของ(weight) > W

วิธีทำ

1. ทำการเรียงลำดับสิ่งของตามมูลค่าต่อน้ำหนักดังตาราง 5.2

ตาราง 5.2 ลำดับสิ่งของตามมูลค่าต่อน้ำหนัก

i	W_i	P_i	P_i/W_i
1	2	4,000	2,000
4	5	5,000	1,000
2	10	6,000	600
3	7	3,500	500
ถ้าเป็นถุงเป้แบบหยิบบางส่วน ➔ W=16={2,5,9} มูลค่าสูงสุดที่จะใส่เป้ได้ไม่เกิน(bound) = 4,000+5,000+9*600=14,400 ฿			

2. คำนวณมูลค่าสูงสุดที่จะใส่เป้ได้
3. พิจารณาบรรจุสิ่งของที่มีมูลค่าต่อน้ำหนักมากที่สุดก่อน ที่โหนด [1][1] คำนวณ

   มูลค่า(totalprice) = 4000 ฿

   น้ำหนัก(totalweight) = 2 กก. < 16 กก.

   Bound price = 0+4,000+5,000+(16-7)*600=14,400 ฿
4. ที่โหนด [2][1] คำนวณ

   มูลค่า(totalprice) = 4000+5000 ฿

   น้ำหนัก(totalweight) = 2 + 5 = 7 กก. < 16 กก.

   Bound price = 0+4,000+5,000+(16-7)*600=14,400 ฿
5. ที่โหนด [3][1] มูลค่า (totalprice) = 4000+5000+6000 = 15000 ฿ น้ำหนัก(totalweight) = 2+5+10 = 17 กก. > 16 กก.

   มูลค่า(totalprice) = 4,000+5,000+6,000=15,000 ฿

   น้ำหนัก(totalweight) = 2 + 5 +10= 17 กก. > 16 กก.
6. น้ำหนักเกินความจุ ตัดกิ่งก้านสาขาที่อยู่ลึกลงไปทิ้ง(pruning)
7. เมื่อน้ำหนักเกินความจุจะย้อนรอยกลับไปจุดก่อนหน้า (โหนดที่ [2][1]))
8. ที่โหนด[3][2]

   มูลค่า(totalprice) = 4,000+5,000=9,000 ฿

   น้ำหนัก(totalweight) = 2 + 5 = 7 กก. < 16 กก.

   Bound price = 0+4,000+5,000+3,500=12,500 ฿
9. ที่โหนด[4][1]

   มูลค่า(totalprice) = 4,000+5,000+3,500 = 12,500 ฿

   น้ำหนัก(totalweight) = 2 + 5 +7 = 14 กก. < 16 กก.

   Bound price = 0+4,000+5,000+3,500 =12,500 ฿
10. เมื่อเลือกสิ่งของสุดท้ายให้ย้อนรอยกลับไปจุดก่อนหน้า(โหนด[3][2])
11. ที่โหนด[4][2]

    มูลค่า(totalprice) = 4,000 +5,000=9,000฿

    น้ำหนัก(totalweight) = 2 +5 =7 กก. < 16 กก.

    Bound price = 0+4,000+5,000 =9,000 ฿
12. เมื่อเลือกสิ่งของสุดท้าย พบว่ามูลค่าน้อยกว่าที่โหนด [4][1] ยังไม่ใช่คำตอบที่ดีที่สุด ย้อนรอยกลับไปที่โหนด [1][1]
13. ที่โหนด[2][2]

    มูลค่า(totalprice) = 4,000 ฿

น้ำหนัก(totalweight) = 2 กก. < 16 กก.

Bound price = 0+4,000+6,000+(16-12)*500 =12,000 ฿

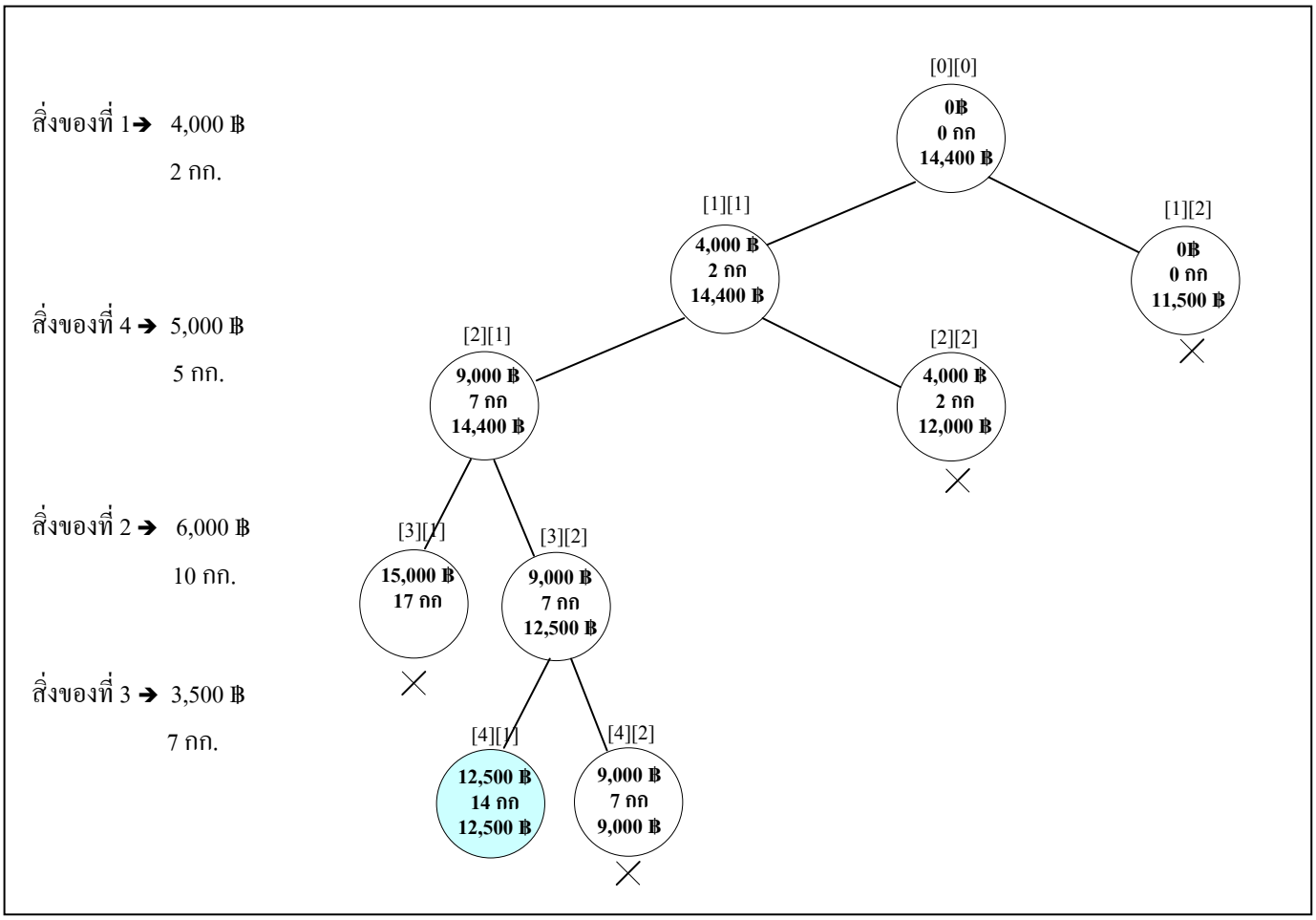
14. Bound price น้อยกว่ามูลค่าที่โหนด[4][1] ตัดกิ่งก้านสาขาที่อยู่ลึกลงไปทิ้ง(pruning)
15. ย้อนรอยกลับไปที่โหนด [0][0]
16. ที่โหนด[1][2]

มูลค่า(totalprice) = 0 ฿

น้ำหนัก(totalweight) =0 +10=12 กก. < 16 กก.

Bound price = 0+5,000+6,000+(16-15)*500 =11,500 ฿

17. Bound price น้อยกว่ามูลค่าที่โหนด[4][1] ตัดกิ่งก้านสาขาที่อยู่ลึกลงไปทิ้ง(pruning)
18. คำตอบที่ดีที่สุด คือที่โหนด [4][1] เลือกหยิบสิ่งของที่ 1, 4, 3 ตามรูป 5.2



รูป 5.2 แสดง state space tree ของปัญหาถุงเป้ 0-1

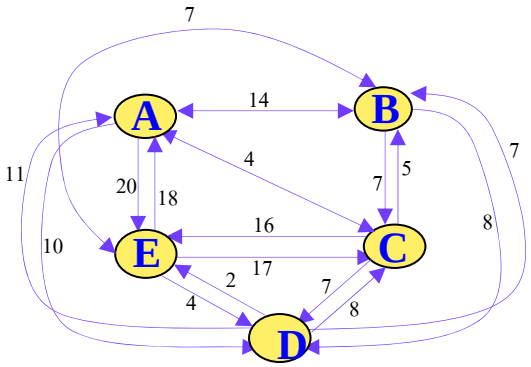
คราวนี้มาพิจารณาขนาดของ Solution space ในระดับที่ 0 มี 1 โหนด, 2, 4, …, n^2

Solution space = $1+2+2^2+\ldots+2^n = 2^{n+1}+1$

5.3 ปัญหาการเดินทางของพนักงานขาย (traveling salesman problem)

ปัญหาพนักงานขายเป็นปัญหาคลาสสิคอีกปัญหาหนึ่ง กล่าวคือ พนักงานขายจะต้องเดินทางออกจากเมือง A เพื่อไปติดต่อกับลูกค้าในเมืองต่าง ๆ และวนกลับมายังเมือง A โดยต้องการให้เส้นทางที่พนักงานขายเดินทางมีระยะทางสั้นที่สุด

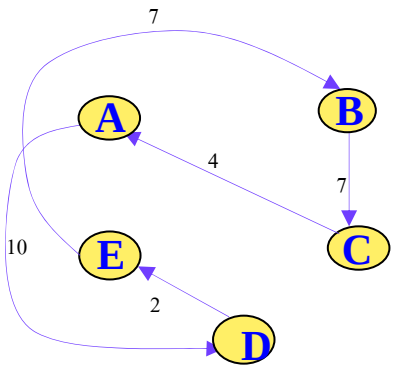
ตัวอย่าง 5.3 กำหนดให้ กราฟแทนเมืองต่าง ๆ และระยะทางที่เชื่อมระหว่างแต่ละเมืองแสดงในรูปที่ 5.3 และใช้เมตริกซ์ตามรูป 5.4 แทนระยะทางระหว่างเมือง โดยมีสำนักงานอยู่ที่เมือง A และให้พนักงานขายเริ่มออกจากสำนักงานแล้วติดต่อลูกค้าตามเมือง B,C,D,E และวนกลับมาที่สำนักงานที่เมือง A จงหาเส้นทางที่มีระยะทางสั้นที่สุดเพื่อให้พนักงานขายเดินทางเริ่มจาก A ไปเยือนทุกเมืองและวนกลับมาที่เมือง A



รูป 5.3 กราฟแทนเมืองและถนนที่เชื่อมเมืองพร้อมทั้งระยะทาง

	A	*B*	*C*	*D*	*E*
A	0	14	4	10	20
B	14	0	7	8	7
C	4	5	0	7	16
D	11	7	9	0	2
E	18	7	17	4	0

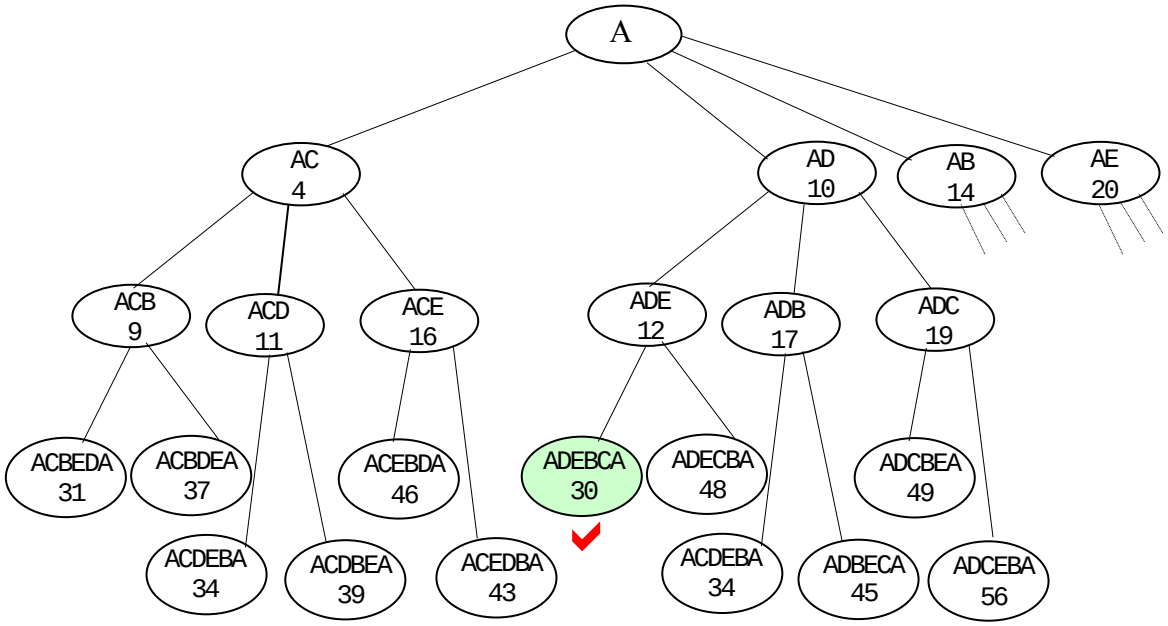
รูป 5.4 เมตริกซ์แทนเมืองและระยะทางระหว่างเมือง



รูป 5.5 เส้นทางที่สั้นที่สุด ADEBCA ระยะทาง = 30

วิธีทำ

1. เราจะเลือกเดินทางระยะทางที่สั้นที่สุดก่อนระหว่าง AB, AC, AD, AE นั่นคือเส้นทาง AC = 4
2. จากเมือง C จะมีเส้นทาง ACB, ACD, ACE เลือกระยะทางที่สั้นที่สุด คือ ACB = 4+5 = 9
3. จากเมือง B จะมีเส้นทาง ACBD, ACBE เลือกระยะทางที่สั้นที่สุด คือ ACBE =9+7=16 และเมืองที่เหลือ คือเส้นทาง ACBEDA = 16+4+11=31
4. พิจารณาเส้นทางอื่นๆ โดยย้อนรอย กลับมาที่เส้นทาง ACB แล้วเลือกเส้นทาง ACBD = 9+8=17 และเมืองที่เหลือ คือเส้นทาง ACBDEA =17+2+18=37 ตัดเส้นทางนี้ทิ้งเพราะมากกว่า ACBEDA
5. หลังจากนั้นพิจารณาเส้นทางอื่นๆ โดยย้อนรอย กลับมาที่เส้นทาง AC แล้วเลือกระหว่าง ACD กับ ACE เลือกระยะทางที่สั้นที่สุด คือ ACD = 4+7 = 11
6. จากเมือง D จะมีเส้นทาง ACDB, ACDE เลือกระยะทางที่สั้นที่สุด คือ ACDE =11+2 =13 และเมืองที่เหลือ คือเส้นทาง ACDEBA = 13+7+14=34 ตัดเส้นทางนี้ทิ้งเพราะมากกว่า ACBEDA
7. ย้อนรอยกลับไปข้อ 6 เหลือเส้นทาง ACDB = 11+7=18 และเมืองที่เหลือ คือเส้นทาง ACDBEA = 18+7+18 = 43 ตัดเส้นทางนี้ทิ้งเพราะมากกว่า ACBEDA
8. หลังจากนั้นพิจารณาเส้นทางอื่นๆ และตัดการแตกกิ่งก้านสาขาในกรณีที่ระยะทางที่ได้ใหม่มากกว่าหรือเท่ากับระยะทางเดิม ทำเช่นนี้ต่อไปเรื่อย ๆ จะได้ state space tree ดังรูป5.5



รูปที่ 5.6 แสดง state space tree ของปัญหาการเดินทางของพนักงานขาย

จากรูปที่ 5.6 จะได้เส้นทางที่มีระยะทางสั้นที่สุดเป็น ADEBCA และมีระยะทางรวมเป็น 30
จำนวนโหนด = 1 + 4 + 4x3 + 4x3x2 = 41

การขยายและจำกัดเขต(Branch and Bound)

เราจะนำปัญหาถุงเป้แบบ มานำเสนออีกครั้งในเทคนิควิธีการขยายและจำกัดเขต

5.4 ปัญหาถุงเป้แบบ 0-1

ตัวอย่าง 5.4 จงพิจารณาปัญหาถุงเป้ 0-1 โดยมีสิ่งของ 4 ชนิดที่จะพิจารณาบรรจุลงถุงเป้ ให้ w_i แทนน้ำหนักของสิ่งของชนิดที่ i และ p_i แทนมูลค่าของสิ่งของชนิดที่ i

w_i = [2,5,10,7]

p_i = [4000, 6000, 3500, 5000]

โดยถุงเป้มีความจุเป็น 16 กก. และต้องการบรรจุสิ่งของลงในถุงเป้ให้มีมูลค่าสูงสุด จงแสดง State Space tree ที่ใช้วิธีการขยายและจำกัดเขต(Branch and Bound)
ในการตัดผลลัพธ์ที่ไม่นำไปสู่ผลลัพธ์ที่ดีที่สุดออก พร้อมทั้งแสดงผลลัพธ์ที่ดีที่สุด

วิธีทำ ทำการเรียงลำดับสิ่งของตามมูลค่าต่อน้ำหนักดังตารางข้างล่างนี้ เหมือนกับบทที่แล้ว โดยพิจารณาบรรจุสิ่งของที่มีมูลค่าต่อน้ำหนักมากที่สุดก่อน แล้วพิจารณาโหนดต่างๆในลักษณะ breadth first search

1. ทำการเรียงลำดับสิ่งของตามมูลค่าต่อน้ำหนักดังตารางข้างล่างนี้

i	W_i	P_i	P_i/W_i
1	2	4,000	2,000
4	5	5,000	1,000
2	10	6,000	600
3	7	3,500	500
ถ้าเป็นถุงเป้แบบหยิบบางส่วน ➔ W=16={2,5,9} มูลค่าสูงสุดที่จะใส่เป้ได้ไม่เกิน(bound) = 4,000+5,000+9*600=14,400 ฿			

2. คำนวณมูลค่าสูงสุดที่จะใส่เป้ได้
3. พิจารณาบรรจุสิ่งของที่มีมูลค่าต่อน้ำหนักมากที่สุดก่อน ที่โหนด [1][1] คำนวณ

   มูลค่า(totalprice) = 4000 ฿

   น้ำหนัก(totalweight) = 2 กก. < 16 กก.

   Bound price = 0+4,000+5,000+(16-7)*600=14,400 ฿

4. ที่โหนด[1][2]

   มูลค่า(totalprice) = 0 ฿

   น้ำหนัก(totalweight) =0 +10=12 กก. < 16 กก.

   Bound price = 0+5,000+6,000+(16-15)*500 =11,500 ฿

5. ที่โหนด [2][1] คำนวณ

มูลค่า(totalprice) = 4000+5000 ฿

น้ำหนัก(totalweight) = 2 + 5 = 7 กก. < 16 กก.

Bound price = 0+4,000+5,000+(16-7)*600=14,400 ฿

6. ที่โหนด[2][2]

มูลค่า(totalprice) = 4,000 ฿

น้ำหนัก(totalweight) = 2 กก. < 16 กก.

Bound price = 0+4,000+6,000+(16-12)*500 =12,000 ฿

7. ที่โหนด[2][3]

มูลค่า(totalprice) = 5,000 ฿

น้ำหนัก(totalweight) = 5 กก. < 16 กก.

Bound price = 5,000+6,000+(16-15)*500 =11,500 ฿

8. ที่โหนด[2][4]

มูลค่า(totalprice) = 0 ฿

น้ำหนัก(totalweight) = 0 กก. < 16 กก.

Bound price = 6,000+(16-10)*500 =9,000 ฿

Bound price ที่โหนดนี้เท่ากับมูลค่าของโหนด [2][1] ตัดกิ่งก้านสาขาที่อยู่ลึกลงไปทิ้ง

9. ที่โหนด [3][1]

มูลค่า (totalprice) = 4000+5000+6000 = 15000 ฿

น้ำหนัก(totalweight) = 2+5+10 = 17 กก. > 16 กก.

10. น้ำหนักเกินความจุ ตัดกิ่งก้านสาขาที่อยู่ลึกลงไปทิ้ง(pruning)

11. ที่โหนด[3][2]

มูลค่า(totalprice) = 4,000+5,000=9,000 ฿

น้ำหนัก(totalweight) = 2 + 5 = 7 กก. < 16 กก.

Bound price = 0+4,000+5,000+3,500=12,500 ฿

12. ที่โหนด[3][3]

มูลค่า(totalprice) = 4,000+6,000=10,000 ฿

น้ำหนัก(totalweight) = 2 + 10 = 12 กก. < 16 กก.

Bound price = 0+4,000+6,000+(16-12)*500=12,000 ฿

13. ที่โหนด[3][4]

มูลค่า(totalprice) = 4,000 ฿

น้ำหนัก(totalweight) = 2 กก. < 16 กก.

Bound price =4,000+3,500 =7,500 ฿

Bound price น้อยกว่ามูลค่าของโหนด[3][3] ตัดกิ่งก้านสาขาที่อยู่ลึกลงไปทิ้ง

14. ที่โหนด[3][5]

    มูลค่า(totalprice) = 5,000 + 6,000 = 11,000 ฿

    น้ำหนัก(totalweight) = 15 กก. < 16 กก.

    Bound price =5,000+6,000+(16-15)*500 =11,500 ฿

15. ที่โหนด[3][6]

    มูลค่า(totalprice) = 5,000 ฿

    น้ำหนัก(totalweight) = 5 กก. < 16 กก.

    Bound price =5,000+3,500 =8,500 ฿

    Bound price น้อยกว่ามูลค่าของโหนด[3][5] ตัดกิ่งก้านสาขาที่อยู่ลึกลงไปทิ้ง

16. ที่โหนด[4][1]

    มูลค่า(totalprice) = 4,000+5,000+3,500 = 12,500 ฿

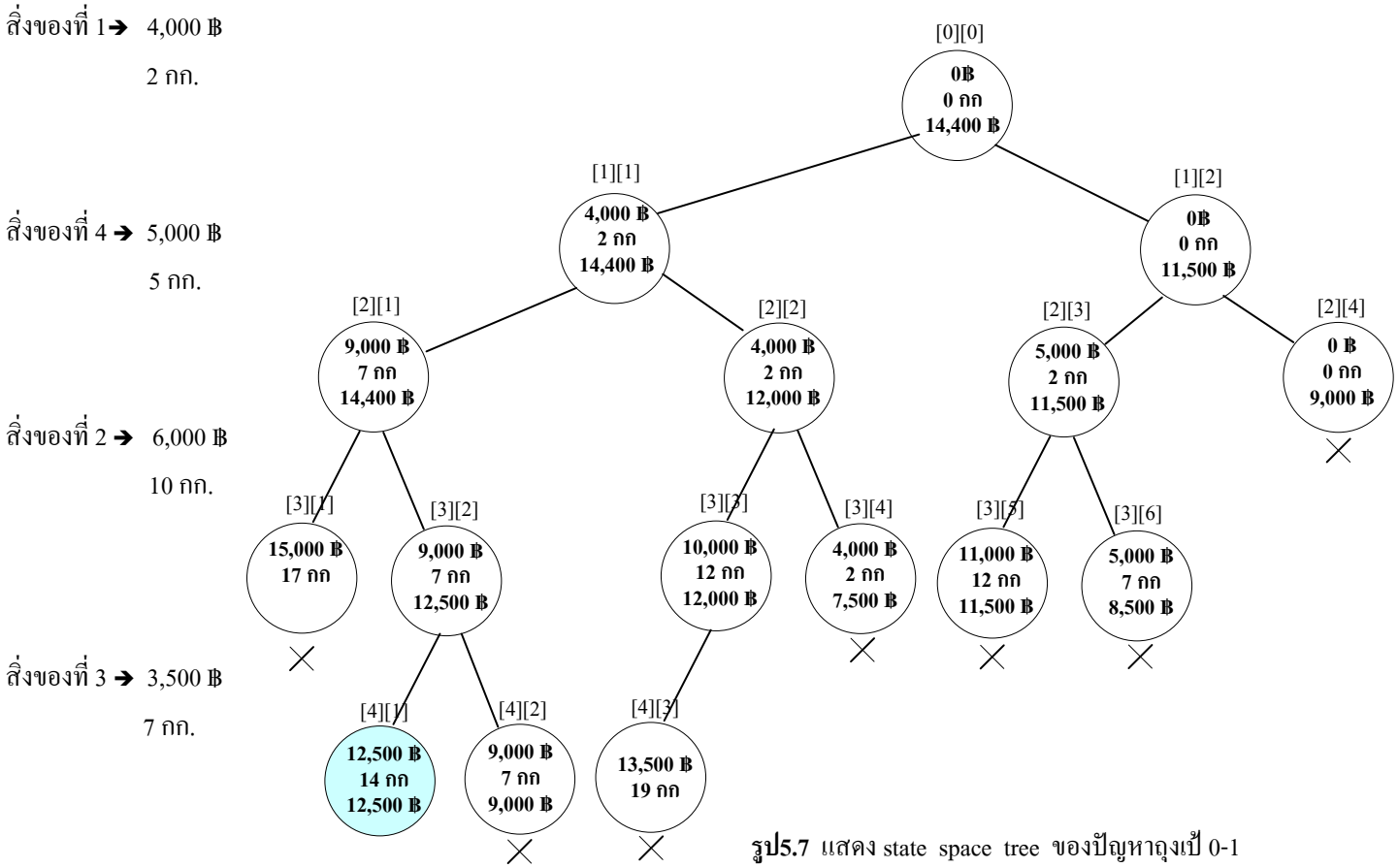
    น้ำหนัก(totalweight) = 2 + 5 +7 = 14 กก. < 16 กก.

    Bound price = 0+4,000+5,000+3,500 =12,500 ฿

17. ได้มูลค่าสูงกว่า bound price ของโหนดอื่นๆ ที่ผ่านการพิจารณามาแล้ว

18. คำตอบที่ดีที่สุด คือที่โหนด [4][1] เลือกหยิบสิ่งของที่ 1, 4, 3

Solution space = $1+2+2^2+\ldots+2^n = 2^{n+1}+1$



รูป5.7 แสดง state space tree ของปัญหาถุงเป้ 0-1

แบบฝึกหัด

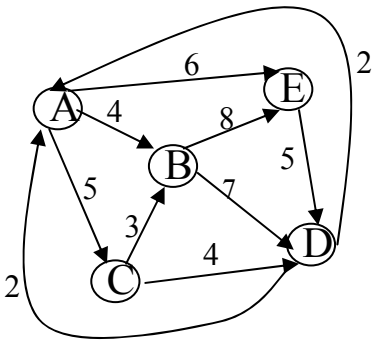
1. จงพิจารณาปัญหา 0/1 แน็บแซ็ก โดยมีสิ่งของ 4 ชนิดที่จะพิจารณาบรรจุลงถุงเป้ ให้ w_i แทนน้ำหนักของสิ่งของชนิดที่ i และ p_i แทนมูลค่าของสิ่งของชนิดที่ i

w_i = [10, 4, 5, 8]

p_i = [60, 16, 25, 36]

โดยถุงเป้มีความจุเป็น 15 และต้องการบรรจุสิ่งของลงในถุงเป้ให้มีมูลค่าสูงสุด จงแสดง State Space tree ที่ใช้วิธี backtracking & branch and bound ในการตัดผลลัพธ์ที่ไม่นำไปสู่ผลลัพธ์ที่ดีที่สุดออก พร้อมทั้งแสดงผลลัพธ์ที่ดีที่สุด

2.



จากกราฟที่กำหนดให้ พนักงานขายต้องเดินทางจากเมือง A จะต้องไปติดต่อกับลูกค้าในเมือง B,C, D, E และวนกลับมาที่เมือง A จงหาเส้นทางที่มีระยะทางสั้นที่สุดที่เดินทางเริ่มจาก A ไปเยือนทุกเมืองและวนกลับมาที่เมือง A

3. ปัญหา subset sum ให้เลือกหยิบสิ่งของ ให้ได้น้ำหนักรวมเท่ากับ 21 กก. กำหนดสิ่งของให้ 5 ชิ้น น้ำหนักชิ้นที่ 1, 2, .. เป็น 5, 6, 10, 11, 16 กก.ตามลำดับ จงเขียน state space tree และหาทุกคำตอบที่เป็นไปได้ (5+6+10, 10+11, 5+16)

4. จงออกแบบขั้นตอนวิธีการย้อนรอย (back tracking) สำหรับรับค่าเลขจำนวนเต็มบวก C และหาว่า ผลบวกของเลขจำนวนเต็มบวกทั้งหมดที่ให้ค่าเท่ากับ C

ตัวอย่างถ้า C = 6 จะได้ 1+2+3, 1+5, 2+4, 6

ถ้า C=10 จะได้ 1+2+3+4, 1+2+7, 1+3+6, +1+4+5, 1+9, 2+3+5, 2+8, 3+7, 4+6, 10