String Matching

Input: $X = x_1 x_2 x_3 \ldots x_n$

$Y = y_1 y_2 y_3 \ldots y_m$ $\quad n \geq m$

Output: j ที่ $X(j) = Y$

$X(j) = x_j x_{j+1} x_{j+2} \cdots x_{j+m-1}$

$O(m+n)$ deterministic algorithms

KMP : (Knuth-Morris-Pratt)

BM : (Boyer-Moore)

Randomized Algorithm

- ของ Rabin + Karp
- Monte Carlo

แนวคิด (Signature matching)

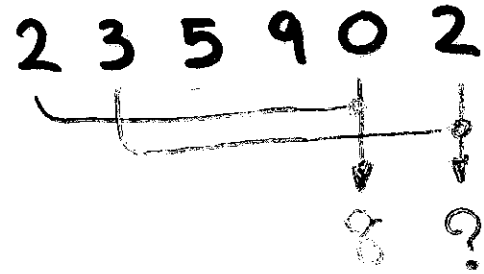
$Y = 31415$

$X = 23590231415267399$

8 9 3 11 0 1 7 8 4 5 10 11 7

$S(Y) = Y \bmod 13$

$S(31415) = (7)$, $S(23590) = 8$



$S(X(j))$

23590	8	(mod 13)
− 20000	− 2×3	
3590	2	
× 10	× 10	
35900	7	
+ 2	+ 2	
35902	9	

หา $S(X(j))$ จาก $S(X(j-1))$

ได้ใน $\Theta(1)$

หา $S(X(1))$ ได้ใน $\Theta(m)$

ลูบเปรียบเทียบ

```
for( j=1 to n-m )
    if( S(Y) == S(X(j)) return j
return 0
```

$O(n+m)$

- False hit ทำให้ ผิด
- มีน้อย หรือมาก ขึ้นกับ ฟังก์ชัน $S(Y)$

$$S(Y) = Y \bmod p$$

- เลือก p เป็น prime ขนาดใหญ่

$\Pr[\text{false hit}]$ ก็จะน้อย

เช่น เลือก p ขนาดประมาณ $n^2 m \log(n^2 m)$

ทำให้ $\Pr[\text{false hit}] = O(\frac{1}{n})$

(รายละเอียด หน้า 305 - 306)

```
StringMatching_MC( X , Y )
    p ← bigRandomPrime( n, m )
    for( j=1 to n-m )
      if( S(Y)== S(X(j)) return j
    return 0
```

```
StringMatching_LV( X, Y )
  do {
       j = StringMatching_MC(X,Y)
  } while( j ≠ 0 && X(j) ≠ Y)
  return j
```

Universal Hashing

- แกล้ง alg. ที่ใช้ hashing ได้แน่ๆ ถ้ารู้ hash function
- แกล้งไม่ได้ ถ้าไม่เปิดเผย หรือ สุ่ม hash fn จาก set ของ hash fn ที่ดีๆ

Carter & Wegman (1979)

$$H = \{ h(x) \mid h(x) = ((ax+b) \% p) \% m$$

m คือ ขนาดตาราง

p คือ prime ใน $[U, 2U)$

a คือ จำนวนเต็ม $0 < a < p$

b คือ " " $0 \le b < p \}$

x มีค่าในช่วง $[0, U)$

ถ้าสุ่ม h จาก H จะได้ว่า

$$Pr[h(x) = h(y) \mid x \neq y] \le \frac{1}{m}$$

→ expected number of collisions $= \frac{n}{m}$

Randomized Algorithms

- ได้ algorithm ที่ "ง่าย" กว่า
- ได้ algorithm ที่ "เร็ว" กว่า
- ได้ algorithm

  CSMA/CD

ดู randomized algorithm เพื่อ

- foiling an adversary
- random sampling
- abundance of witness
- finger printing (signature)
- random reordering
- load balancing
- symmetry breaking

State space search

- DFS, BFS, LC-Search ช้า
- Backtracking เร็วขึ้น
- Branch & Bound เร็วขึ้น (แต่ก็ยังช้า)

• ใช้ partial solution ให้เป็นประโยชน์ ในการ "เล็ม" ต้นไม้

Branch & Bound

• ใช้กับ optimization problem
• สมมติว่าเป็น minimization
• ต้องมีวิธีคิด lower bound cost ของ partial solution
• มี lower bound cost กำกับ node
• ใช้ "_____" ในการเลือก node ใหม่ เพื่อแตกลูก (Branch)
• ใช้ cost ของ solution ที่ดีที่สุดที่ได้พบเป็นตัว "เล็ม" ต้นไม้ (Bound)

Branch & Bound

- ทำงาน ทำแบบเร็ว ขึ้นกับ lower bound cost function

Assignment Problem

- n คน n งาน
- cost matrix : c_{ij} ให้ i ทำงาน j
- minimize $\sum_{1 \le i \le n} c_{i\,p(i)}$

	1	2	3	4
1	14	12	15	20
2	15	10	14	25
3	10	19	17	23
4	16	18	8	40

lower bound cost :

$\sum$ ตัวน้อยสุดในแต่ละ: แถว แนวนอน

$12 + 10 + 10 + 8 = 40$

$\sum$ ตัวน้อยสุดในแต่ละ: แถว แนวตั้ง

$10 + 10 + 8 + 20 = 48$

~~14~~	(12)	~~15~~	~~20~~
15	10	14	25
10	19	17	23
16	18	8	40

$\sum$ ตัวน้อยสุดในแต่ละแถวแนวนอน

(12) + 14 + 10 + 8 = 44

0/1 Knapsack (maximization problem)

Upper bound value

	1	2	3	4	5	6	
v_i	10	12	6	10	2	6	
w_i	10	14	7	12	6	24	W=23

รู้ว่า value ของ 0/1 $\leq$ value ของ fractional

ใช้ greedy alg. หา value รวม ของ แบบ fractional

Traveling Salesperson Problem

- หา distance matrix
- หา tour ที่สั้นสุด (minimization problem)

i \	1	2	3	4
1	0	10	20	15
2	8	0	12	14
3	10	6	0	10
4	4	2	6	0

d_{ij} = ระยะทาง จาก i ไป j

$$\sum_{\forall i}$$

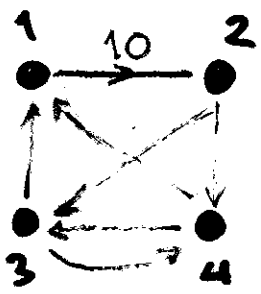
เลือกเส้นสั้นสุดเฮาะครั่งหนึ่ง

เลือกเส้นสั้นสุดเฮาะครั่งหนึ่ง

Lower bound tour length

$$\left(\frac{4}{2}+\frac{10}{2}\right)+\left(\frac{2}{2}+\frac{8}{2}\right)+\left(\frac{6}{2}+\frac{6}{2}\right)+\left(\frac{10}{2}+\frac{2}{2}\right)$$

	1	2	3	4
1	0	10	20	15
2	8	0	12	14
3	10	6	0	10
4	4	2	6	0



$$\left(\frac{4}{2}+\frac{10}{2}\right)+\left(\frac{10}{2}+\frac{2}{2}\right)+\left(\frac{6}{2}+\frac{10}{2}\right)+\left(\frac{10}{2}+\frac{4}{2}\right)$$

Local Search

- branch & bound ก็ทำ
- ส่วนใหญ่ไม่ใช่ partial solution
- ส่วนมากแต่เฉพาะ: complete solution

```
localSearch ( ) {
    s ← initialSolution ()
    while ( ! terminate(s) ) {
        s' ← select ( next(s) );
        if ( accept(s, s') ) s ← s'
    }
}
```

next(s)

TSP: 1→2→5→3→6→4→7

1→2 [5]←3←6←[4] 7

1→2→4→6→3→5→7