Greedy Set 03

Adapt from Slide ดร.กันต์ ศรีจันทร์ทองศิริ

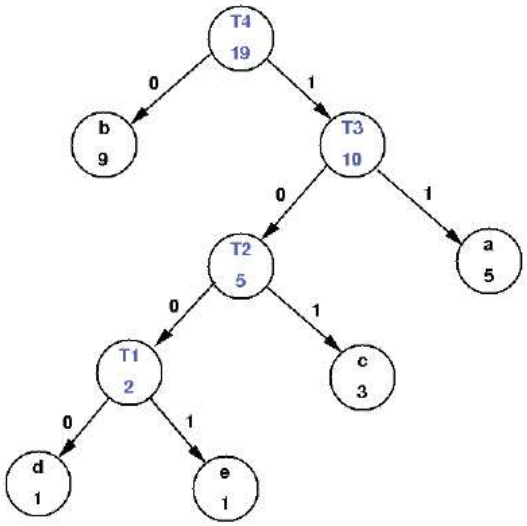
Huffman Coding

- ตามปกติ รหัส Unicode หรือ ASCII ใช้จำนวนบิตเท่ากันสำหรับทุกตัวอักษร
  - e ใช้ 8 bits. z ก็ใช้ 8 bits. (ASCII)
- แต่ e มักปรากฏบ่อยกว่า z
  - Variable-length coding: ถ้าใช้จำนวนบิตแทนตัวอักษรที่พบบ่อย (e) น้อยกว่าจำนวนบิตแทนตัวอักษรที่ไม่ค่อยพบ (z), จะใช้พื้นที่ในการเก็บข้อความทั้งหมดได้น้อยลง
  - (ไม่สามารถลดจำนวนบิตของทุกตัวอักษรได้ บางตัวต้องมี code ที่ยาวขึ้น)

- สมมติว่า e = 10, z = 11 (fixed-length codes)
  eezeee = 101011101010 => 12 บิต
- แต่อาจจะใช้ e = 0, z = 1111 (variable-length codes)
  eezeee = 001111000 => 9 บิต

- นอกจากนี้ การใช้ variable-length coding มีเงื่อนไขว่า code ของตัวอักษรหนึ่ง ๆ ต้องไม่เป็น prefix ของ code ของตัวอักษรอื่น
- a = 1, b = 11
- Code = 111 ...ไม่รู้ว่าเป็น aaa หรือ ab หรือ ba

ใช้ binary tree แทน code ของแต่ละตัวอักษร



รหัสของแต่ละตัวอักษรจาก tree

b = 0

a = 11

c = 101

d = 1000

e = 1001

- การใช้ binary tree โดยให้ตัวอักษรอยู่ที่ leaf เท่านั้น ทำให้มั่นใจได้ว่ารหัสของตัวอักษรหนึ่ง จะไม่เป็น prefix ของรหัสของตัวอักษรอื่น

- **ปัญหา:** มีข้อความยาว ๆ ข้อความหนึ่ง และรู้ว่าตัวอักษรแต่ละตัวปรากฏกี่ครั้งในข้อความนี้ (frequency) ควรจะให้รหัสตัวอักษรต่าง ๆ เป็นอะไร เพื่อที่ใช้รหัสเก็บข้อความนี้แล้ว ใช้พื้นที่น้อยที่สุด
  - หรืออีกอย่างก็คือ สร้าง tree รหัสอย่างไร

Mahidol University Wisdom of the Land
ACM-ICPC ACM-ICPC Thailand Central Group B Programming Contest 2013
SIPA IBM ICT

Huffman Coding

- ความถี่ของแต่ละตัวอักษร:
- a = 4
- b = 2
- c = 1
- d = 8

- เริ่มให้ทุกตัวอักษรเป็น tree ที่มี node เดียว
- ให้ความถี่ของตัวอักษรเป็นค่าของ node นั้น
- หยิบ 2 tree ที่มีความถี่ต่ำสุดมาเชื่อมกัน โดยสร้าง node ใหม่ขึ้นมา และให้ root ของ 2 tree นี้เป็นลูกทั้งสองของ node ใหม่
- ให้ค่าความถี่ของ node ใหม่เท่ากับผลรวมของความถี่ของลูกทั้งสอง
- ทำไปเรื่อย ๆ จนทั้งหมดรวมเป็น tree เดียว

- สังเกตว่า เป็นวิธี greedy
- นอกจากนี้ ยังได้คำตอบที่ optimal ด้วย