CS F364: Design & Analysis of Algorithm

07 Huffman Algorithm

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

Feb 03, 2021 ONLINE (Campus @ BITS-Pilani Jan-May 2021)

http://ktiwari.in/algo

Huffman codes

Huffman invented a **greedy algorithm** that constructs an optimal prefix code called a Huffman code.

- It is a variable-length prefix code, useful for lossless data compression

Consider for example, a data file of 100,000 characters only containing six characters `a, b, c, d, e, f`

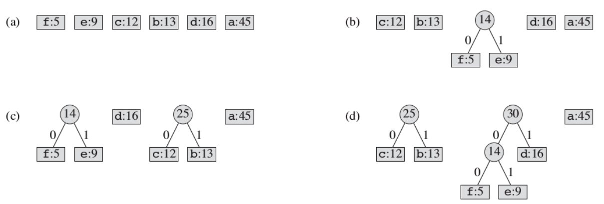
	a	b	c	d	e	f
Frequency(in k)	45	13	12	16	9	5
Fixed length code	000	001	010	011	100	101
Huffman code	0	101	100	111	1101	1100

- Fixed length code takes 300k bits
- Huffman code needs 224k bits (∼25% compression)

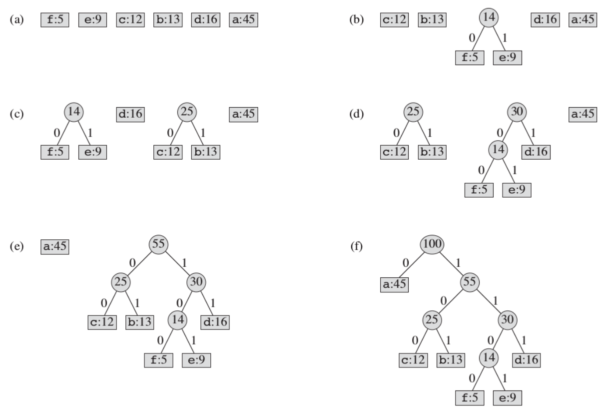
Huffman code

Algorithm 1: HUFFMAN(C)

Q = C
for *i=1 to length(C)* − 1 **do**
  Allocate New node z
  $z.left = x =$ EXTRACT-Min(Q)
  $z.right = y =$ EXTRACT-Min(Q)
  $z.freq = x.freq + y.freq$
  Insert (Q, z)
return EXTRACT-Min(Q)



Huffman code



Complexity

Algorithm 2: HUFFMAN(C)

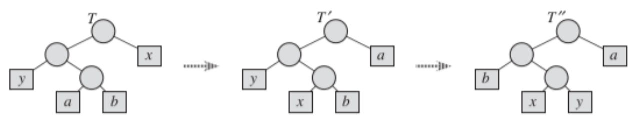
Q = C
for *i=1 to length(C)* − 1 **do**
  Allocate New node z
  $z.left = x =$ EXTRACT-Min(Q)
  $z.right = y =$ EXTRACT-Min(Q)
  $z.freq = x.freq + y.freq$
  Insert (Q, z)
return EXTRACT-Min(Q)

- If you assume EXTRACT-Min takes $O(\log n)$
- Inner block is called $n - 1$ times

So total the time is $O(n \log n)$

Correctness - greedy choice property

There exists optimal prefix code, with two lowest frequency characters having same length codewords, differing only in the last bit.



- Let x, y be two lowest freq items. And a, b are at bottom

$$\begin{aligned}
B(T) - B(T') &= \sum_{c \in C} c.f \times d_T(c) - \sum_{c \in C} c.f \times d_{T'}(c) \\
&= x.f \times d_T(x) + a.f \times d_T(a) - x.f \times d_{T'}(x) - a.f \times d_{T'}(a) \\
&= x.f \times d_T(x) + a.f \times d_T(a) - x.f \times d_T(a) - a.f \times d_T(x) \\
&= (a.f - x.f) \times (d_T(a) - d_T(x)) \geq 0
\end{aligned}$$

- Similarly $B(T') - B(T'') \geq 0$
- Since T is optimal $B(T) \leq B(T'')$ So $\boxed{B(T) = B(T'')}$

Correctness - Optimal Substructure (using induction)

Assume it produces optimal tree for size n

- Consider C of size $n+1$, Let us make C' as $C - \{x, y\} + z$ where x and y are minimum frequency item, and $z.f = x.f + y.f$
- As the size of C' is n, so one can get optimal tree T_0 using the algorithm. Expand z in T_0 to get T_1 for C. $\boxed{T_1 \text{ is optimal}}$ how?
- Prove by contradiction. Note that $B(T_1) = B(T_0) + x.f + y.f$
- Let T_2 is optimal tree instead of T_1. Does T_2 has x and y at the deepest leaf? If not make it using greedy choice property.
- In T_2 contract x and y in z using $z.f = x.f + y.f$. Let it becomes T_3

$$\begin{aligned} B(T_3) &= B(T_2) - x.f - y.f \\ &< B(T_1) - x.f - y.f \quad \text{due to our assumption that } T_2 \text{ is optimal} \\ &= B(T_0) \end{aligned}$$

- Contradiction as T_3 and T_0 both are of size n, and at this size algorithm produces optimal tree, two optimal tree can not differ

Thank You!

Thank you very much for your attention! (Reference[1])

Queries ?

[1] [1] Book - *Introduction to Algorithm*, By THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, CLIFFORD STEIN