Algorithms Design and Analysis [ETCS-301]

Dr. A K Yadav
Amity School of Engineering and Technology
(affiliated to GGSIPU, Delhi)
akyadav1@amity.edu
akyadav@akyadav.in
www.akyadav.in
+91 9911375598

October 16, 2019

Huffman Codes I

- Huffman Codes is used for data compression.
- Data is a sequence of characters.
- Each character is given with their frequency
- Each character is encoded into a codeword using some scheme.
- Suppose there are n characters in the set C.
- Each character $c \in C$ have frequency $freq_c$
- $bit(c)$ is the number of bits required to code c character whose frequency is $freq_c$.
- Our aim is to design a encoding scheme so that we can minimize the total length of codeword.

$$\text{minimize} \sum_{c \in C} freq_c \times bit(c)$$

Huffman Codes II

- We can use some standard fixed length formatting such as ASCII
- In this case the the total space requirement will be $8 \times \sum_{c \in C} freq_c$
- But we can save more space using non standard fixed length formatting scheme for n characters.
- The length of the code will be $\lceil \lg n \rceil$
- In this case the $\lceil \lg n \rceil$ number of bits are required to represent each character and total bits will be $\lceil \lg n \rceil \times \sum_{c \in C} freq_c$
- Huffman codes can be used even to save more space than both of the above and this uses variable length encoding scheme for each character.
- Huffman codes are prefix codes.

Huffman Codes III

- **Prefix Codes:** Codes in which no codeword is a prefix of some other codeword are called **Prefix Codes**.
- The benefits of Prefix codes are simplified decoding, unambiguous encoding.
- But the disadvantages is that we can not start decoding in between the encoded codeword into the original character.
- Huffman code uses full binary tree for encoding, in which every non leaf node has two children
- All characters are used as leaf, so total leaf will be n
- There are $n - 1$ internal nodes as in full binary tree.
- Each left child is labelled as 0 and each right child is labelled as 1.
- Label value from root to leaf will be the encoding for that leaf character.

Huffman Codes IV

- All characters are kept in min priority queue according to their frequency that is least frequency character is at front of the queue.
- Every time we sum the least two frequency of the first two element of the queue and make one
- So after $n-1$ operation we will be having only one element in the min priority queue and that will be the root of the tree.
- Code length of the character c will be equal to the depth of the c $d_T(c)$ in tree T. So

$$\text{minimize } B(T) = \sum_{c \in C} freq_c \times d_T(c)$$

Huffman Coding algorithm

HUFFMAN(C)//C is the set of n characters

1. $n = |C|$
2. $Q = C$ //Q is Min-priority Queue
3. for $i = 1$ to $n - 1$
4.   allocate a new node z
5. $z \rightarrow left = EXTRACT - MIN(Q)$
6. $z \rightarrow right = EXTRACT - MIN(Q)$
7. $freq_z = freq_{z \rightarrow left} + freq_{z \rightarrow right}$
8. $INSERT(Q, z)$
9. return $EXTRACT - MIN(Q)$

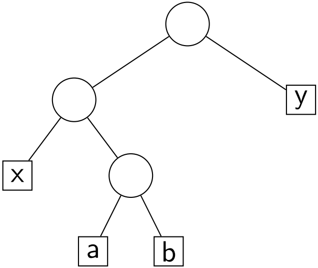
Line 2 will be require $O(n \lg n)$ time for building the min heap of n items. For loop executes $n - 1$ times and each time it rebuild the min heap in $O(\lg n)$ times. Total time complexity will be $O(n \lg n)$

Correctness of the Huffman algorithms I

If character x and y having the least frequency then they will be at highest depth in the some optimal tree T for Huffman Code.

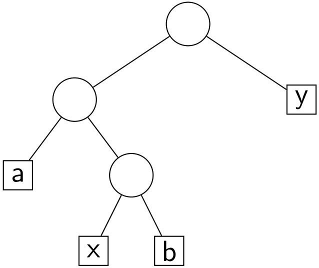
- Let character x and y having the least frequency $freq_x$ and $freq_y$ respectively. Also suppose $freq_x \leq freq_y$
- Suppose x and y are not at highest depth but a and b are at the highest depth in the tree T.
- Let $freq_a \leq freq_b$
- Since $freq_x, freq_y$ are lowest frequency and $freq_a, freq_b$ are any arbitrary frequency. Also $freq_x \leq freq_y$ and $freq_a \leq freq_b$. So $freq_x \leq freq_y \leq freq_a \leq freq_b$.
- if $freq_x = freq_b$ then $freq_x = freq_a = freq_y = freq_b$ and we can change the position of the x and y with a and b and hence proved.

- So assume $freq_x \neq freq_b$ and take tree T as shown in below figure:

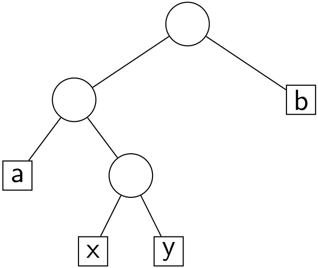


Correctness of the Huffman algorithms III

- Interchange the position of x with a in tree T and build tree $T^{'}$ as shown in below figure:



- Interchange the position of y with b in tree $T^{'}$ and build tree $T^{''}$ as shown in below figure:



Correctness of the Huffman algorithms V

- Find $B(T) - B(T^{'})$

$$B(T) - B(T^{'})$$

$$= \sum_{c \in C} freq_c \times d_T(c) - \sum_{c \in C} freq_c \times d_{T'}(c)$$

$$= freq_x \times d_T(x) + freq_a \times d_T(a) - freq_x \times d_{T'}(x) - freq_a \times d_{T'}(a)$$

$$= freq_x \times d_T(x) + freq_a \times d_T(a) - freq_x \times d_T(a) - freq_a \times d_T(x)$$

$$= (freq_a - freq_x)(d_T(a) - d_T(x))$$

Now $(freq_a - freq_x) \geq 0$ because $freq_x$ is the least frequency and $(d_T(a) - d_T(x)) \geq 0$ because a is at the highest depth. So

$$(freq_a - freq_x)(d_T(a) - d_T(x)) \geq 0$$

$$\Rightarrow B(T) - B(T^{'}) \geq 0$$

$$\Rightarrow B(T) \geq B(T^{'})$$

But $B(T)$ can not be greater then $B(T^{'})$ because $B(T)$ is the optimal value. So the only possibility is $B(T) = B(T^{'})$
Same way we can prove $B(T^{'}) = B(T^{''})$
So $B(T) = B(T^{'}) = B(T^{''})$
So $B(T^{''})$ is an optimal Huffman code where least frequency characters x and y is at the highest depth.

Thank you

Please send your feedback or any queries to akyadav1@amity.edu, akyadav@akyadav.in or contact me on +91 9911375598