FGK Adaptive Algorithm
(Advances in Huffman's Code)

by Herbe Chun

Nov 30, 2023

UC Santa Cruz, Algorithms, Dr Radhika Gover

Huffman Tree

1. Huffman's code was introduced as a tree in 1952 where two items with lowest frequency form the leaves with parent node as the sum of leaf frequencies

2. Although Huffman's code is not new, it still represents the beginning of lossless compression.[1]

Table 1. Item vs Freq

Item	Freq
c	5
d	3
b	2
a	1

Fig 1. Huffman Tree

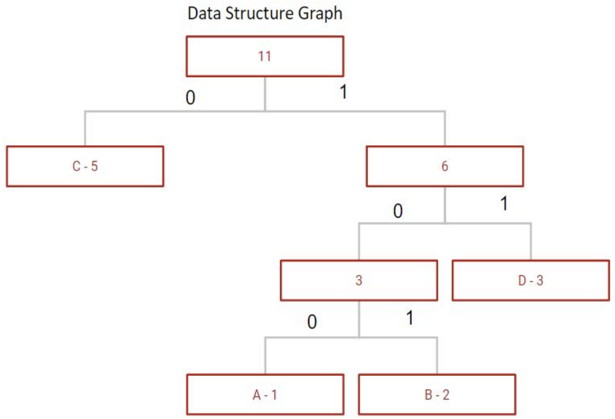


Table 2. Huffman #Bits

Item	Freq	Code	#Bits
c	5	0	5
d	3	11	6
b	2	111	6
a	1	110	3
-	-	Total	20

Example

- If Table 1 was encoded with 0's and 1's in binary, it would take 2^2 codes or 2^2 x (5 + 3 + 2 + 1) = 44 bits of encoding
- In contrast, Huffman code would require (5 x 1 + 3 x 2 + 2 x 3 + 1 x 3) or 20 bits to encode the data

Huffman Code

Huffman Code[2]

- Is a greedy algorithm that builds a solution piece-by-piece by selecting the next piece that offer most impact
- **Pro's** - Huffman is easy to implement, requires low time complexity and is able to generates lossless compression
- **Con's** - Huffman may produces an acceptable solution but maybe not an optimal solution

Although Huffman's code offers significant advantages, its weakness in generating an optimal solutions have given rise to application specific codes

Table 3. Application Specific Code Examples

Application	Description	Time	Footnote
IoT	Modification to Huffman to increase bandwidth, reduce transmit time, increase capacity etc for LoRa's long range, low power IoT network	2019	3
Bio research	An improved Huffman's method to archive text, images, and DNA characters for increased storage efficiency	2018	4
Image compression	An improved Huffman method for archiving video and audio data	2015	5

Advances over Huffman Code

- Although application codes are emerging, it would be more interesting to see what global codes have developed

  Global Codes
- **Priority Queue**: Allows fast access to the min/max elements on a heap binary tree, this tree was introduced in this course.
- **Adaptive Huffman**: The FGK algorithm is an adaptive Huffman algorithm that continuously updates the tree as data is presented.
- **Arithmetic**: Symbols are represented from 0 to 1. This simplifies the process because changes in the symbol does not affect the coder. Although technically not a Huffman, is useful for comparison purposes.

Rank from chatGPT	**Huffman**	**Adaptive Huffman**	**Priority Queue**	**Arithmetic**
Improvement	Baseline	Better adaptability, tree evolves from data stream	Can be used with Huffman for faster access	● Improved compression ● Improved adaptability, tree evolves from data stream
Compression efficiency	8	8	8	8
Complexity	6	7	7	6
Adaptability	***5***	***9***	6	7
Footnote	-	7, 8	6	8

Adaptive Huffman Code (FGK Algorithm)[9, 10]

- Since the FGK Algorithm is an extension of Huffman and because it scored highest for compression efficiency, complexity and adaptability, the following details this algorithm

- The key difference between Huffman and FGK is that the tree is continuously reorganized for efficiency as new data is presented

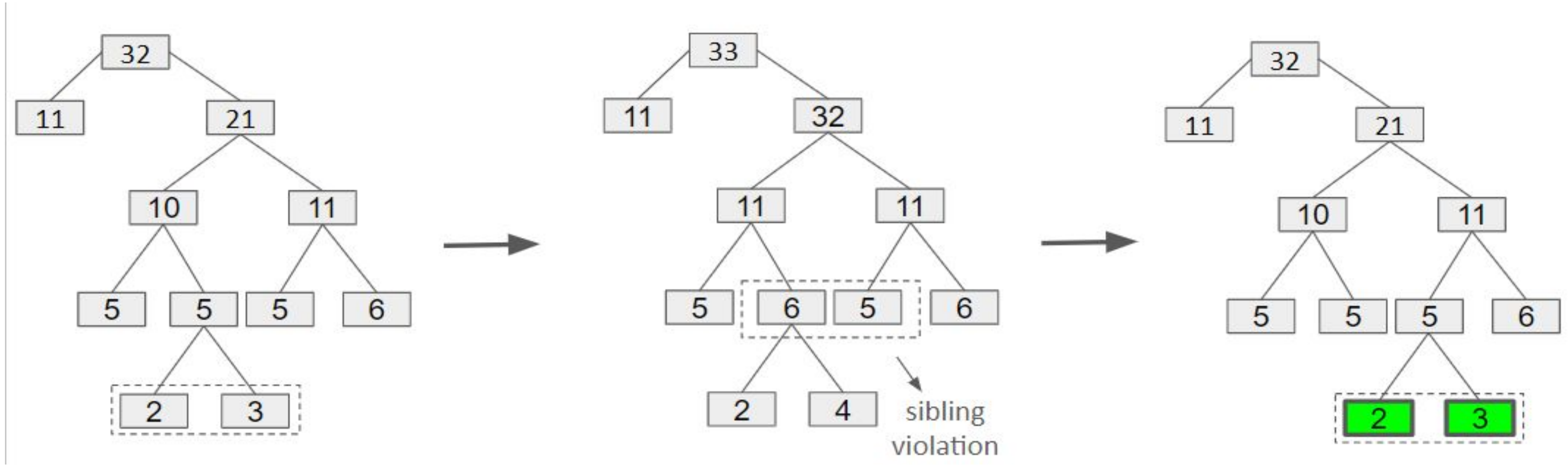
- This algorithm was introduced in 1973 by Faller and Galleger although Vitter introduced another later code in 1987

- Although Viter’s is newer, the original FGK method is highlighted since it is the first of many adaptive solutions

Example - FGK Method[9, 10]

Steps

1. Transform current symbol into a Huffman tree
2. If there is a sibling violation, update tree for the new symbol
3. Repeat step 2 until the sibling violation is no longer an issue
4. The following example is presented by Ankur Singh.



Step 1 - Original Huffman Tree	Step 2	Step 3
--	1. Leaf 3 is updated as 4 2. Tree is updated 3. Sibling violation detected, right sibling is less than left sibling	Leaf 2 and 3 are moved to the right sibling

Conclusion

1. The key difference between Huffman and adaptive algorithms is that the tree is continuously transformed as information is presented.

2. The FGK algorithm is an example of a continuous tree optimization.

3. This continuous update feature is where significant advances are being made in algorithm development.

References

1. Huffman code background, https://en.wikipedia.org/wiki/Huffman_coding
2. Huffman's code is a greedy algorithm, https://www.geeksforgeeks.org/greedy-algorithms-general-structure-and-applications/
3. LoRa WAN, https://ieeexplore.ieee.org/document/9060099
4. BioTech, https://www.future-science.com/doi/10.2144/000113218
5. Image Compression, chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.ijsta.com/papers/ijstav1n4y15/IJSTA-V1N4R10Y15.pdf
6. Priority Que-chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://courses.cs.washington.edu/courses/cse143x/15au/lectures/huffman/huffman.pdf
7. Adaptive Huffman-https://en.wikipedia.org/wiki/Adaptive_Huffman_coding
8. Adaptive Huffman-chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://repository.lsu.edu/cgi/viewcontent.cgi?article=3716&context=gradschool_theses
9. FGK algorithm, https://www.youtube.com/watch?v=D_rphEWfZRE
10. FGK algorithm, https://ics.uci.edu/~dan/pubs/DC-Sec4.html