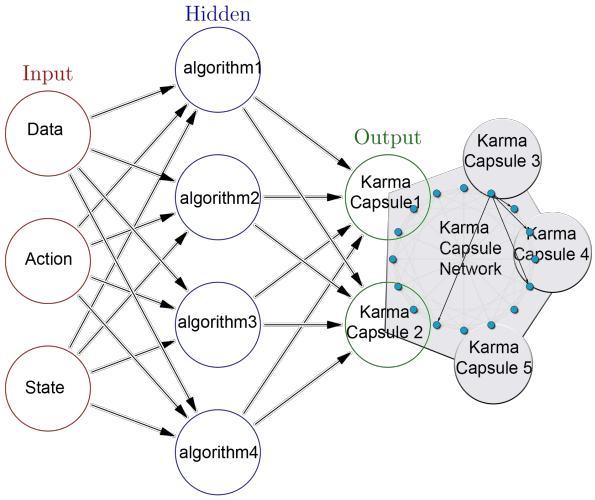
Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

In the current Internet of Things ecosystems, most actors are not interested in the one time Naked-Data from the edge devices, but the “Outcomes" in the form of “Intelligence”, “Values”, and “Service" they can provide on a continuous basis. Karma Capsule Network is a distributed Artificial Intelligence Neural Network model of outcomes from the “**Living Things**”. A living thing shall be a human, a plant, an animal, a robot, an edge device, a machine, a mobile vehicle or computer system. These outcomes are an encapsulation of Data, Actions and ts called “**KarmaCapsules**”. Using digiBlitz, organizations, communities, government and consortiums can build value-oriented networks to Monetize their cyber-physical assets. These network of Capsules shall be used in real-time by the user applications and computing platforms to produce useful services & products. Each one of the node in the network is called a “Capsule” that encapsulating the secured “data” with the immutable “Code”(Artificial intelligence) for various future “state” scenarios.



KarmaCapsule:

Capsule is an encapsulation of Data, AI Code and State.

Applicability

I. digital Assets Monetization

Monetization is the process of converting non-revenue generating assets into tradable assets through matching them with the opportunity. digiBlitz Platform is designed to help the companies to build resilient systems, components and processes to achieve successful monetization of their Assets.

There are three different types of digital Assets : Data, Event, Process.

Author: Maya Suresh Kannan Balabisegan

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

Whether it is Smart Factory or Smart City or Smart Transportation or Smart Energy, there is a limited vision and understanding of how the hard assets will be combined with digital assets to work for the socio, economic and political aspect of the world.

1. Convert Asset into Derivative
2. Store the Asset Data into Data Oracles or Traditional database
3. Store Asset Tokens in Blockchain
4. Store the rata of change of assets, ie derivatives in the State Machine Tree
5. Create Micro Service by encapsulating Data, Event and State
6. Store the Service Compostability values in the Hoffmanns Tree.

Monetizable Assets: Derivative of Data, Derivative of State, Marginal costing and Marginal Profit.

digital Twin:

II. digital Twin Creation using differential & Integral Calculus.

Money Hot-Spot Tree

Definition of Money:

Value -> Asset

Rate of Change of Asset -> Money

Data-> Commodity

Author: Maya Suresh Kannan Balabisegan

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

Data + Function -> Object

Object + State -> Service

Service + State

While Blockchain or DAG can serve as a ledger;

Integrate “Micro Data” to “Objects”; Integrate Objects to Services.

Apply derivative of objects and

The “life” of a human is determined through three aspects:

1. Anatomy
2. Physiology
3. Neurology

While the human body might seems to be intact, we undergo transformations every day with thousands of cells dying and new cells getting created. The neutral network is vital part of the human that gives life to us.

Same way, organizations have the anatomical, physical and neurological parts. The life of an organization undergoes transformation every day. And the “Life” of the organization is determined by the “State” of the organization at a given point of time. An organization is nothing but a huge state machine that transforms from one form to the other.

State of a “Thing” is the Life moment of the “Thing” at specific period of time in the past. If we can store the State in the distributed format where the no one can damper or duplicate and can be recovered at a specific state of the thing, then we can rebuild the series of moments chained together to recreate the sequence of actions to build a larger actions or put the sequence in to Loop between a lower limit and upper limit.

Author: Maya Suresh Kannan Balabisegan

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

What if we can remember the State of a Machine at specific period of time? A machine can be made to produce the right output when it can be fed back with the same “State”.

Human DNA Pattern at specific period before the death can be remembered, we can make the cells to rebuild and brig the human ti life.

A Smart City can be put into auto pilot if it can remember the States of moments of every entity that it had.

Use Case1: Transformational Models for Companies.

Remember, an Enterprise is “One Big State Machine” having many Business state machines working together through the transition

Derivatives

Lets face it, “WE LIVE IN DERIVATIVE WORLD”

What is a Derivative?

Derivative is an instantaneous rate of change of

Something.

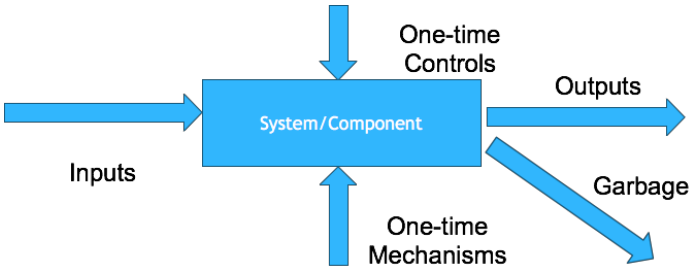
Industrial Age with Static Market Place

Author: Maya Suresh Kannan Balabisegan

Karma Capsule Network (KCN)

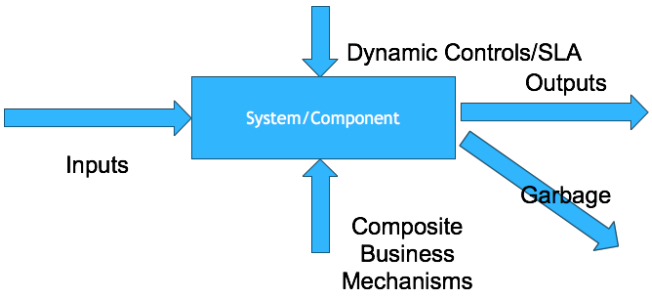
a distributed trust-less
AI-Neural Network of Outcomes

"If you keep giving what you are giving, you will keep getting what
You are getting" –So, don't change any damn thing



"If you keep giving what you are giving, you will NOT keep getting what you are getting because your system is not standalone, it is an Integral part of a larger system – ie your enterprise is an extension of the world enterprise" – So, be ready to deal with "CHANGE"

Information / Service Age with Dynamic Market Place



Change Means Transform.
So, we have to deal with Transformation for solving current
Generation business-technology problems.
But how to Handle/Measure Transformation/Change?
"PROCESS OF CHANGE" Transformation.

We get answer from Calculus.
Ie using Derivative
If x is the item, then derivative f'x = d(y)/d(x)

Author: Maya Suresh Kannan Balabisegan

While a single Business or Technology component (Resource) Say “x” represents “Order” and solves the problem of a function say “Order Management” represented by F(x) = x2 + 2x + c,

Then the “rate of change of order” is represented by (f’x)=2x+2

So, we are very clear from the above example, we should be
dealing with “x” to deal with “Order” and f’xto deal with “Order transformation”. But wait…Isnt the Enterprise a one big State Machine having many
Business state machines working together through the transition?
How are we going to handle it?
The answer again can be found from “Integral Calculus”. Integration
is the necessary thing to get business transformation done correctly.
So, we should be able to Integrate all differential components and its
Derivatives to get the integration within a limit.
Ie from aggregation of rate of change of “Orders” in various time frames
Situations, controls, we should be able to model “Order Management”
Process.

Use Case 2: Business Analytics for Companies:

An organization can repeat a Successful “Quarterly Financial Result”, if it can remember the every State of the Business Process, Functional and Data it was processing during that time period. And it can even improve upon it. It is like going back in time; seeing “Thing” in the organization with respect to the state and repeat the entire process ie Customer Communication. Process Analysis, Production Planning, quality control, Marketing etc ie “Repeat the Value Chain” of the organization.

The science behind the eAstir is building state-machine that build living models of organizations on real-time basis and then feed intelligence to it. Over a period, the BAPNA engine will take care of building future models of the organization through predictive analysis. This will helps the customers to take right business decisions and see various pro-forma scenarios. While the primary target for EAstiris private large implementations for Medium to Large size customers, the major and long run target is to make EAstiras the world’s first BAPNA network where it maintains the living models of all participating private companies and all

public companies.So, in theory, for a reasonable fee, an investor can login and look into all decision models of a particular company in which he/she is interested in investing.
It carries

1. Big Data and 2, Internal Process Model replica (creating future state from the past state).

Transportation Industry: A smart vehicle and its component's state can be remembered to recreate the scenario just before an accident to understand what really went wrong.

Some guidelines for Implementation of Karma Capsule Network

Our solution can be implemented based on powerful algorithms for metadata analysis and artifacts management Markov Chain and Huffmans's. Through these we build and traverse through a tree of various informational entities from various sources of information from state, central agencies, private sources etc to build the profile. Every profile is analyzed for pattern matching by traversing through their links through relational parameters

Algorithms

Implementation of Huffman's Compression algorithm for Modem Protocol.

Requirements:

Language : C/C++

O/S : UNIX

Hardware : 486 & above

Concepts : Data Structure (Trees) using C/C++, File Handling in C/C++.

The first in the list of compression techniques is the Huffman's Compression algorithm.It is a powerful algorithm with less error.The MNP has combination of Huffman's algorithm and run-lenght algorithm.It deals with compression of the data according to the frequency of occurence of each character.The most frequent character will have shortest code and least frequent character will have the longest code.

Author: Maya Suresh Kannan Balabisegan

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

Huffman's Tree:

Building of huffman's tree is very important in the application.The type of the tree we need to create is extended binary tree.Only the leaf nodes will carry the information.

Steps in building Huffman's tree:

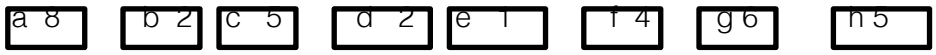
Consider You have 8 characters in your file with following frequencies.

a---8, b---2, c--5, d---2, e---1, f---4, g---2, h---5

The structure for tree is

```
Struct tree
{
    char info;
    int  freq;
     int flag;
     struct tree * left;
     struct tree * right;
};
```

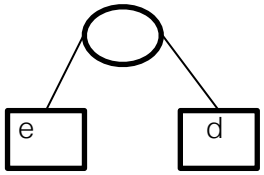
Step1: Create nodes for each character.



Step 2:

find the two least frequent nodes.In this case e & b or e & d

create a new non-info node and make these two as children of that node.



Author: Maya Suresh Kannan Balabisegan

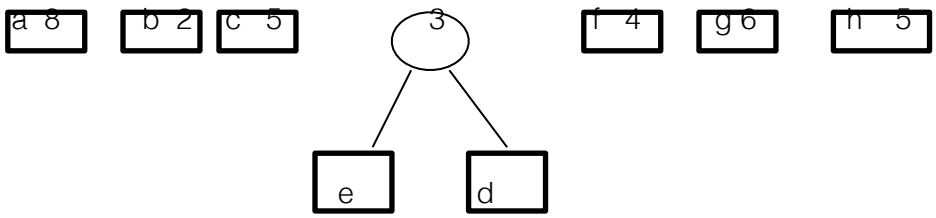
Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

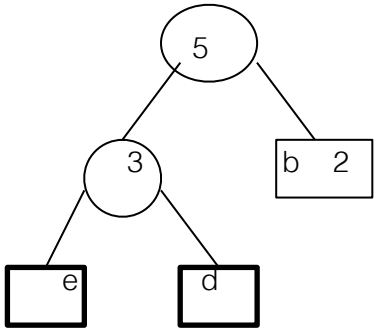
Step 3:

Make the frequency of newly created node to be added frequency of the children. In this case the frequency of the newly created node is 1+2=3.

Once again see for the two least frequent nodes replacing the children by the parent node in the list.



Create a new non-info node and make the least frequent nodes as children to it.

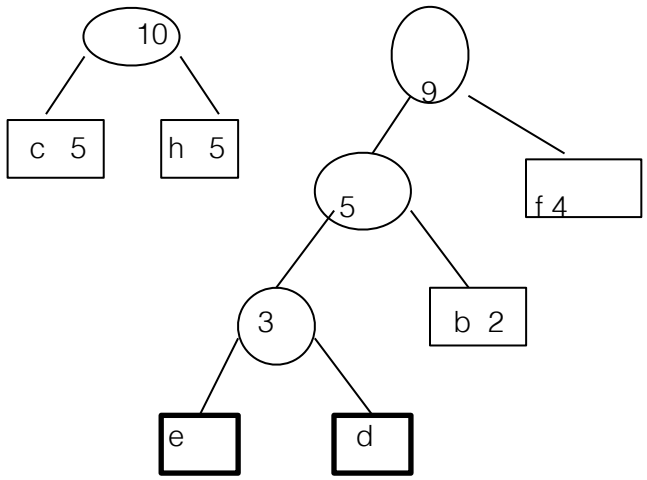


Author: Maya Suresh Kannan Balabisegan

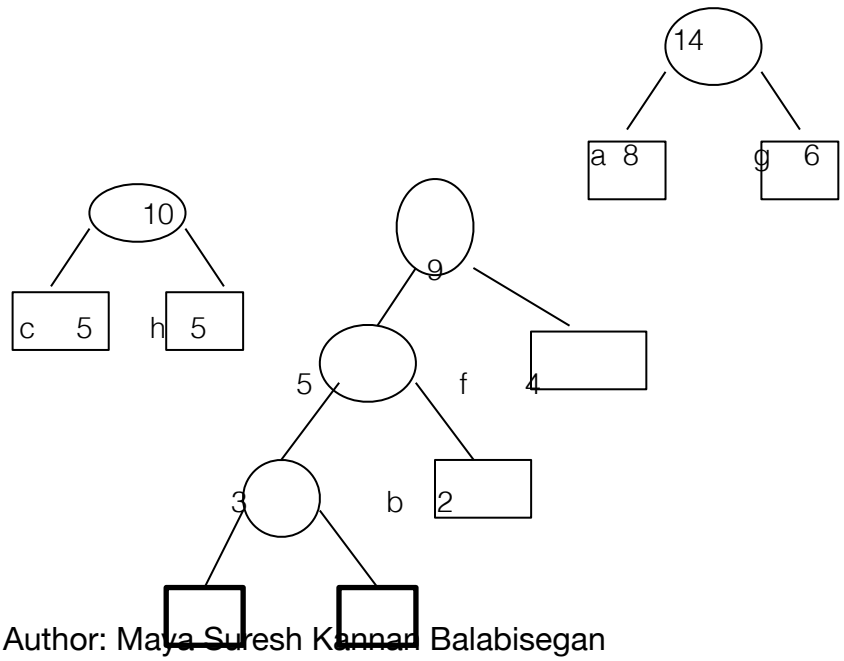
Step 4

Repeating the above procedure will give the final tree as below.

4-1

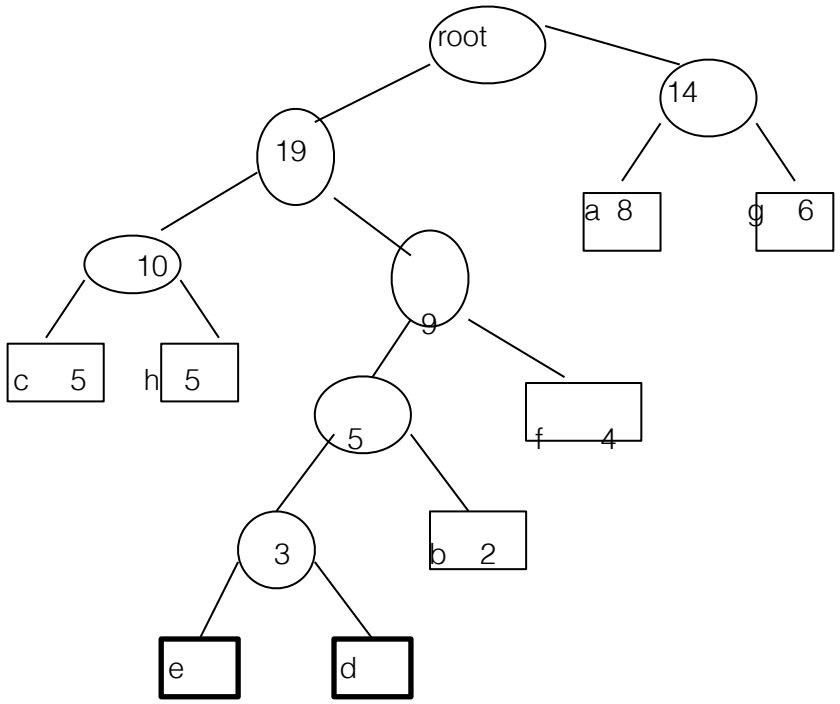


4-2



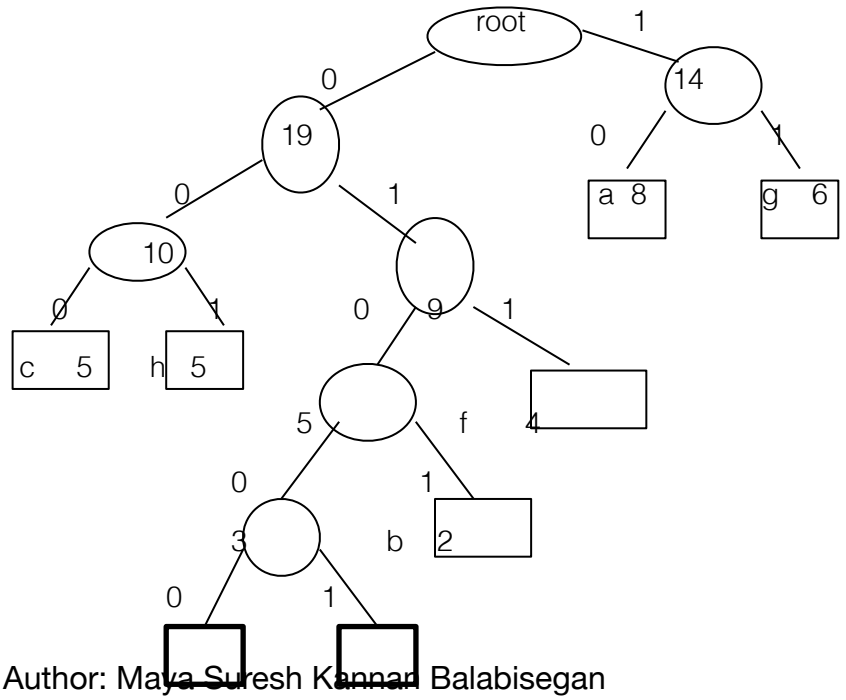
e d

4-3



Step 5

Make the flag member of every left node be 0 and every right node to 1.



Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

e d

Steps for Compressing data

Step1

Open the file whose data to be compressed.Open another file for storing the compressed data.

Step2

Read the input file and form a frequency table(list of characters and their frequency).

Step 3

Create a Huffman's tree using the characters as nodes as shown above.

Step4

Open the input file and read a character.Traverse the tree till you reach the leaf node whose info is equal to the read character.Now monitor the path you have travelled to reach it (1 or 0).That path value is the equivalent binary code for your character.

e.g

a----->10 e------>01000 f------>011

Step 5

Store the equivalent binary value into the output file.Now the character which initially occupied 8 bits in the input file now occupies only two bits.

Step 6.

Repeat the above two steps till you read all the characters from the input file and store their code into the output file.

Author: Maya Suresh Kannan Balabisegan

Steps for De-Compressing data

Step1

Open the compressed file and the output file in which the recovered information is to be stored.

Step2

Read the data bit by bit from the compressed file and traverse the Huffman's tree by matching the bit value to the flag value.Now you will reach a leaf node which is has the required character as its info.Store it in the output file.

Step3

Repeat the above steps till you recover all the data.

Frequently Asked Questions

1) Where should I keep the frequency table?

The frequency table can be stored in an array or a linked list.

2) How can I actually arrange the nodes in the Huffman's tree?

Arranging the nodes in the Huffman's tree is a tedious process.The following is the simplest method:

Author: Maya Suresh Kannan Balabisegan

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

Step1.

Create twice the number of nodes of the number of characters in an array using malloc:

struct tree *root;

root=(struct tree *)malloc(sizeof(struct tree)*n*2);

n--->number of characters in the frequency table.

Step2

Put the right and left of first n nodes to NULL.

Step3

Travel through the array to find least frequent nodes.Add their frequency and put the frequency of n+1 th array element to this value.Now put the address of the first least element into the left member and address of the second least element into the right member.

see the following table:

Info	Freq	Left	Right
a	8	NULL	NULL
b	2	NULL	NULL
c	5	NULL	NULL
d	2	NULL	NULL
e	1	NULL	NULL
f	4	NULL	NULL
g	6	NULL	NULL
h	5	NULL	NULL
Temp1	3	&e	&d
Temp2	5	&Temp1	&b

Author: Maya Suresh Kannan Balabisegan

Step4

Repeat the above step till all the n*2 nodes gets linked to form a tree.

3) How can I traverse a tree ?

Tree traversal can be of any one of the standard methods

Pre-order, In-order or Post-order.

The simplest way to implement them is using recurrsive function:

```
void inorder(struct tree *hdr)
  {
     if(hdr==NULL)
     return;
     else
      inorder(hdr->left);
      //do some operation .for example printing the info
      inorder(hdr->right);
     }
```

4) How can I store the compressed data in bits into the file?

The actual bits occupied by a character is 8.But if we want to store a code like 001 into a file it will be very difficult. One of the simplest approach is to pack the character variable with two or more codes and then store it in the file. This needs operations like bitwise OR and LEFT SHIFT.

5) How can I reconstruct the Huffman's tree if iam given only the compressed file?

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

The reconstruction of huffman's tree need the frequency table.The problem is every file will have a different frequency table and in turn different tree structure.So we cannot send the frequency table separately to the receiver to decode it.

So we should store the frequency table itself at the end of every compressed file.This may look absurd for small files because instead of decreasing the memory occupancy it increases it.But for large files this is a very small figure compared to the data available in the file.

Implementation of Shannon-Fano algorithm for Modem Protocol.

The shannon fano algorithm is another algorithm which gives almost similar results that of the Huffman's algorithm.

Implementing Shannon-Fano Algorithm:

Let us take the message be a,b,c,d,e and f with frequency of 1,3,2,3,1 and 2 respectively.

Step1

Define a class as below,

```
class stree
{     private:
        char info;  // storing character
      int  freq;     // storing its frequency of occurrence
      int flag ;     //stating path value as 0 or 1
        public:
       void getinfo( );
       void setinfo( );
      ……
        :
         :
        class stree ** child;     // building child nodes
};
```

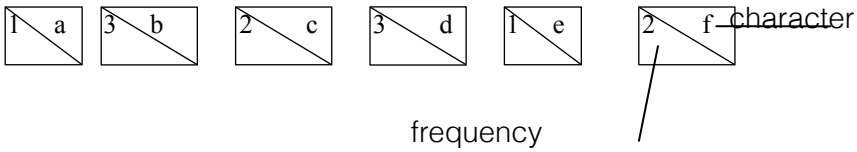
Author: Maya Suresh Kannan Balabisegan

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

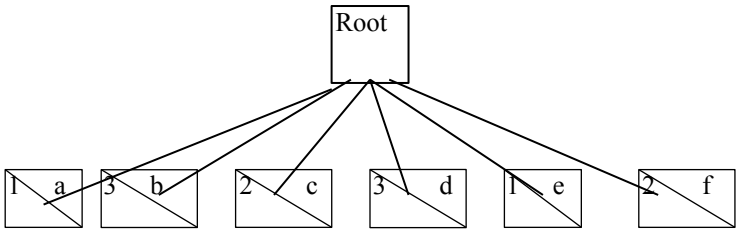
Step2:

Create five nodes (i.e) six objects to stree dynamically.



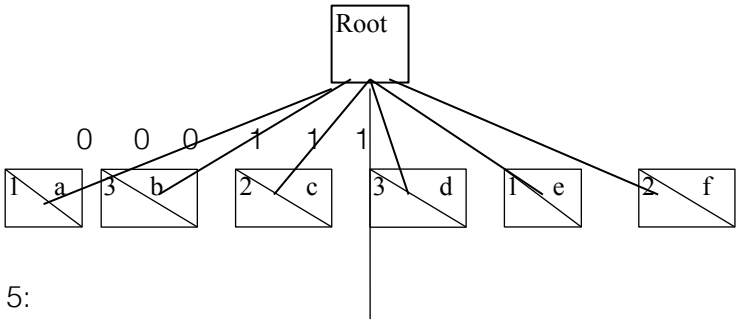
Step 3:

Create a root node and make these as its children.



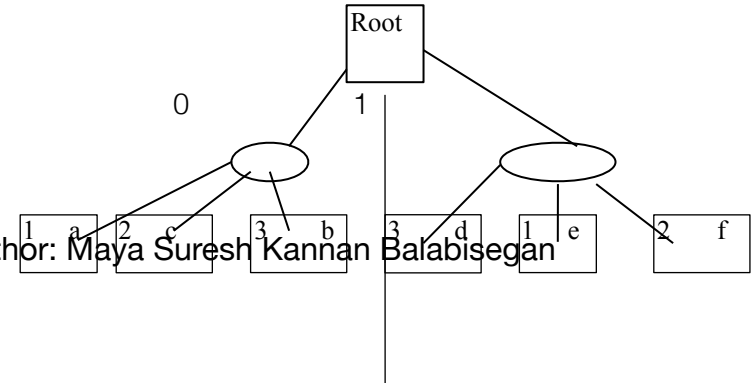
Step 4:

Divide the nodes into two sections according to equal frequencies. Make one group's flag value 0 and other 1



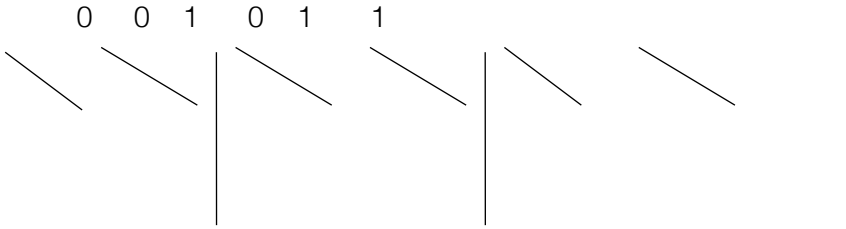
Step 5:

Create an intermediate node instead of 0 group and 1 group and make the nodes as its child 0 or 1 according to frequency.



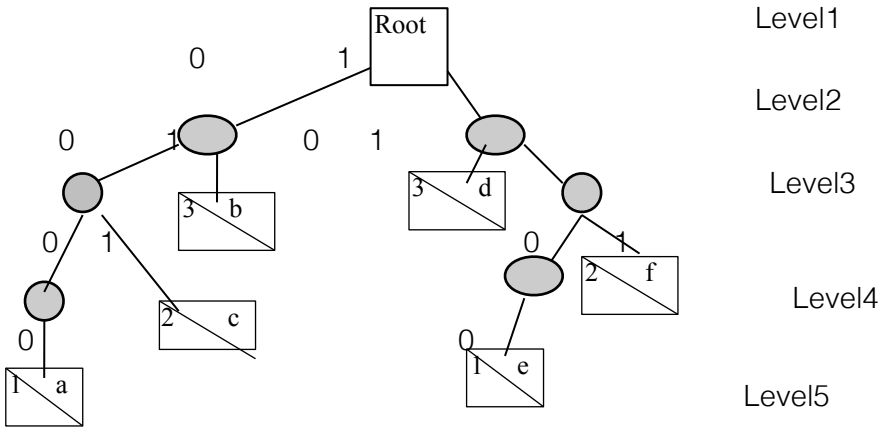
Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes



If you can't divide into equal then do the next division within the group making less frequent nodes in the group or which has more than one member in the group to the next level.

So finally the tree will look like this



So the equivalent code for

a------? 0000 b-----?01 c------?001
d----? 10 e----?1100 f------?111

After creating the Shannon's tree the steps are same as that of Huffman's algorithm implementation.

Implementation of Lempel-ziv's Compression algorithm for Modem Protocol

Author: Maya Suresh Kannan Balabisegan

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes

Till now some of the algorithms are been developed like single-character compression(ASCII Tab substitution), run-length compression, Huffman's coding algorithm(Statistical repetition coding) which are developed based on the single concept of repetition of information . But the more general approach was to look for the historical repetition of strings, i.e to encode entire strings that have previously occurred. This is the Lempel-Ziv algorithm. The gives a problem of making the output occupying more bits than the input if you have data without repeated sequences. The V.42 modem protocol which uses this algorithm overcomes this problem by switching off the compression at the time of non-repetition of sequences.

Lempel-Ziv algorithm:

The two major components in the algorithm are (1) Accumulator(a temporary storage) and (2) dictionary (large array of strings).

The input is first stored in the accumulator and compared with the existing strings in the array. If a match is not found, the unmatched strings is put into the array ,and its index in the array becomes its token. When the input string is matched, the token is output in place of the string.

The input is a 8-bit word but the output is a 12-bit word to facilitate storing the Hexadecimal dictinary token.

(e.g)

Sno	Accumulator	Input (8 bit)	Output (12 bit)	Dictionary (Hex)
1	--	x	--	--
2	x	y	x	100=xy
3	y	x	y	101=yx
4	x	y	--	--
5	xy	m	312	102=xym

Author: Maya Suresh Kannan Balabisegan

6	m	k	m	103=mk
7	k	l	k	104=kl

Certain procedural rules has to be followed :

1. The first 265 tokens are reserved for single,unmatched bytes.

2. Attempted matching begins when the accumulator contains two characters.

3. Input characters gather in the accumulator until a dictionary match fails.

4. After the match fails, the new string accumulator is placed in the dictionary, and the token representing the partial match is output.The most recent character (the one that caused the match to fail) remains in the accumulator to begin a new match.

In the above example:

1) The first character x is moved on to the Accumulator.Neither the dictionary nor the accumulator is altered on the first character because there are two characters in the accumulator(Rule 2). The x remains in the accumulator to start a new match.

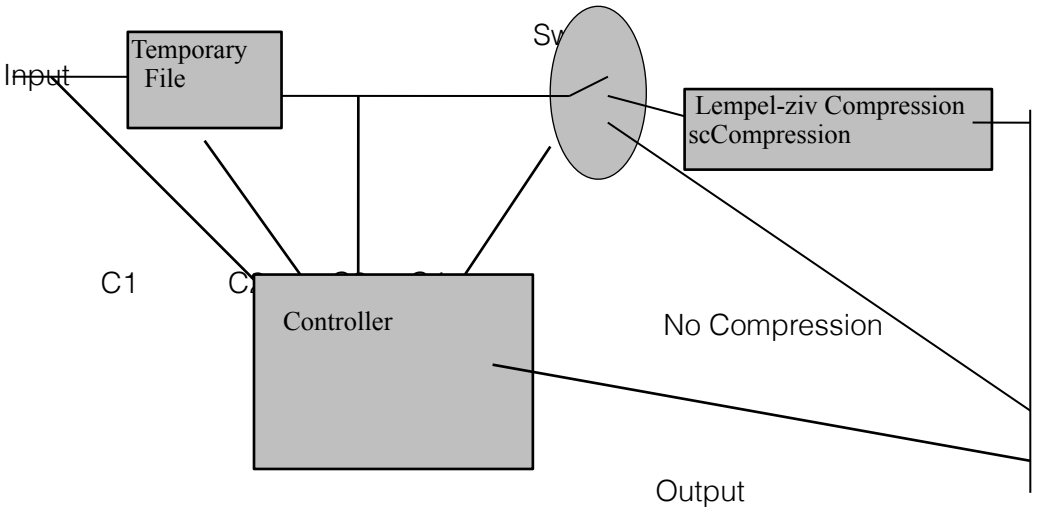
2) The second character y is appended to the x already in the accumulator.The dictionary is searched for xy and no match is found.This new string is copied into the dictionary and assigned the next available token number,100 hex(Rule 1) .The x is output and the y remains in the accumulator to start a new string match.

From the above example we can see that row5 substitutes 312, a 12-bit word instead of xy which is 8+8=16 bit word. This is because of the repetition of the sequence xy. But row2,row3,row6 and row7 goes on generating new sequence, but their output is a 12-bit word whereas the input is only 8-bit. This gives rise to negative Compression. The V.42 bis when comes across situation like this, simply switches OFF the compression, simply giving out the 8-bit data as the output thus avoiding negative compression.

So the system should look like below:

Karma Capsule Network (KCN)

a distributed trust-less
AI-Neural Network of Outcomes



C1- control for reading input according to size of file

C2- control taking decision on the file's lines(i.e) to mark signal for non-repeated sequences.

C3 - Control to send output to switch

C4- Control to Operate switch

C5- Control to Add signal to indicate absence of compression

The Controller is a process which keeps on monitoring the happenings around the system using the controls C1-C5. The switch is a daemon process which acts according to the information it is getting through C4. This desperately needs IPC and signal interrupts.The control C5 plays a roll of embedding some signal characters to the compressed file that the compression is stopped.The dictionary which has been developed by the controller should be included into the compressed file in order to facilitate de-compression.If we get the dictionary we can de-compress the data very easily.

Karma Capsule Network has evolved over the ten years since 2007 and got matured in October 2016. The practical implementation is an evolution process and 2018 October is the biggest evolution year and month. In the next few months Karmacapsule will be getting complete practical shape and i Will be releasing the final version of the Paper.

By: Maya Suresh Kannan Balabisegan (suresh.balabisegan@gmail.com)

Author: Maya Suresh Kannan Balabisegan