Ehindistudy.com

Data structure pdf ebook in hindi

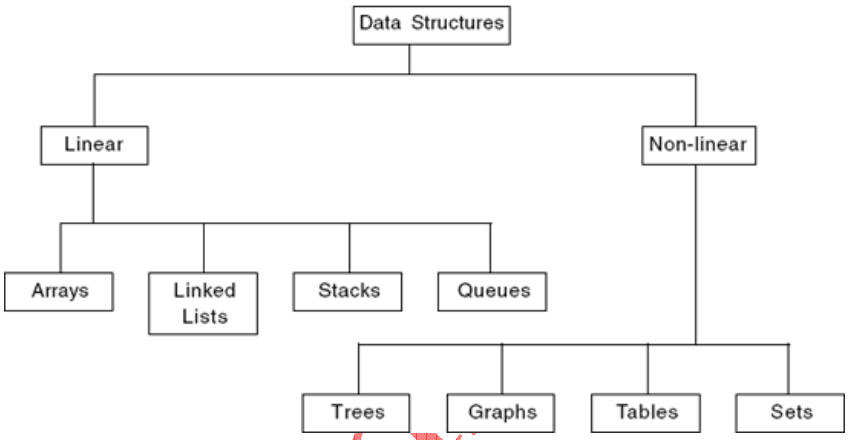
What is data structure in hindi? & classification in hindi

data structure in hindi:-

data structure किसी कंप्यूटर सिस्टम में डेटा को स्टोर तथा व्यवस्थित(organise) करने का एक तरीका होता है। जिससे कि हम डेटा का आसानी से इस्तेमाल कर सकें।

अर्थात डेटा को इस प्रकार स्टोर तथा organise किया जाता है कि उसको बाद में किसी भी समय आसानी से access किया जा सकें।

types of data structure in hindi:-



data structure दो प्रकार के होते है:-
1:-Primitive डेटा स्ट्रक्चर
2:-Non-primitive डेटा स्ट्रक्चर

1:-Primitive डेटा स्ट्रक्चर:- primitive डेटा स्ट्रक्चर वह डेटा स्ट्रक्चर होता है जिसे direct ही मशीन instructions से operate किया जा सकता है।
अर्थात यह सिस्टम तथा compiler के द्वारा डिफाइन होता है।

Non-primitive डेटा स्ट्रक्चर:- primitive डेटा स्ट्रक्चर वह डेटा स्ट्रक्चर होता है जिसे direct मशीन instructions से operate नही किया जा सकता है। ये डेटा स्ट्रक्चर primitive डेटा स्ट्रक्चर से derived होते है।
Non-primitive डेटा स्ट्रक्चर दो प्रकार का होता है:-
1:-Linear डेटा स्ट्रक्चर
2:-Non-linear स्ट्रक्चर

1:-Linear data structure:-

linear एक ऐसा डेटा स्ट्रक्चर है जिसमें डेटा items को linear(रेखीय) रूप में संग्रहित तथा व्यवस्थित किया जाता है, जिसमें एक डेटा item दूसरे से एक रेखा के रूप में जुड़ा होता है।
ex:- array, linked list, queue, stack.

2:-Non-linear data structure:-

Non-linear एक ऐसा डेटा स्ट्रक्चर है, जिसमें डेटा items को क्रमबद्ध (sequential) तरीके से व्यवस्थित नही किया जाता है।
जिसमें एक डेटा item किसी भी अन्य डेटा items के साथ जुड़ा हुआ हो सकता है।
ex:-tree, graph.

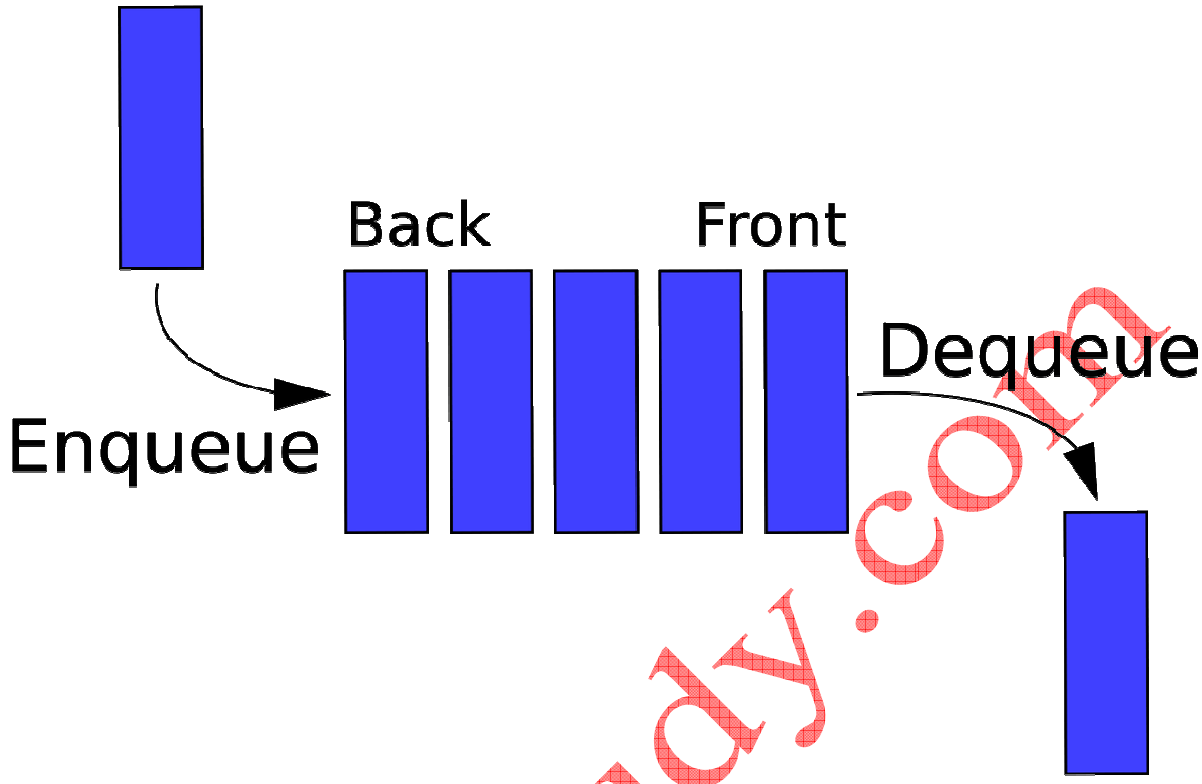
What is Queues in hindi?

data structure Queues in hindi:-

Queue एक non-primitive तथा linear डेटा स्ट्रक्चर है यह FIFO(first in first out) के सिद्धान्त पर कार्य करता है अर्थात वह item जो कि सबसे पहले add किया जाता है वही item सबसे पहले remove किया जायेगा और वह item जो कि सबसे अंत में add किया जाता है उसे अंत में ही remove किया जायेगा।

Queue को हम अपनी वास्तविक दुनिया में अक्सर ही प्रयोग करते हुए देखते है, चलिए इसका उदाहरण देखते है:-
“रेलवे का उदाहरण लेते है, एक व्यक्ति जो रेलवे में टिकट रिजर्वेशन की लाइन में सबसे पहले लगा होता है और सबसे पहले टिकट लेकर चले जाता है, वह व्यक्ति जो last में लगा हुआ रहता है वह अंत में ही बाहर जायेगा।”

Queue में दो end होते है एक front end होता है तथा दूसरा rear end होता है। Rear end में item को add किया जाता है तथा front end से item को remove किया जाता है।



Queues in hindi

Queue में दो प्रमुख ऑपरेशन होते है:-
1:-Enqueue
2:-Dequeue

जब हम Queue में कोई item डालते है तो वह प्रक्रिया Enqueue कहलाती है तथा जब हम Queue से कोई item निकालते है तो वह प्रक्रिया Dequeue कहलाती है।

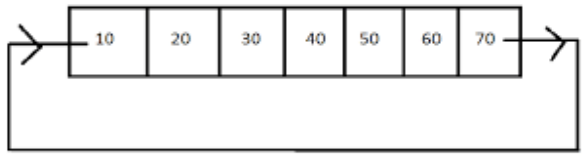
Queue की शर्तें:-Queue की निम्नलिखित शर्तें होती है:-
1:-FRONT<0 है तो, Queue खली(रिक्त) है।
2:-REAR=size of Queue है तो, Queue पूरा भरा हुआ होता है।
3:-FRONT<REAR है तो, Queue में कम से कम एक item तो होता ही है।
4:-अगर आपको Queue में कुल item की संख्या जाननी है तो:-(REAR-FRONT)+1.

Types of Queues in hindi

Types of Queues in hindi:-

डेटा स्ट्रक्चर में Queues निम्नलिखित प्रकार के होते है:-
1:-Circular queue
2:-Dequeue.

1:-Circular queue:-Circular queue को हम ring-buffer भी कहते है। Circular queue में जो अंतिम नोड होता है वह सबसे पहले नोड से जुड़ा हुआ रहता है। जिससे कि circle का निर्माण होता है। यह FIFO के सिद्धान्त पर कार्य करता है। Circular Queue में item को rear end से add किया जाता है तथा item को front end से remove किया जाता है।



Circular Queue

2:-Dequeue:-Dequeue का पूरा नाम double-ended queue है। Dequeue एक ऐसा डेटा स्ट्रक्चर है जिसमें हम items को front तथा rear end दोनों से add भी कर सकते है और remove भी कर सकते है।

Dequeue के दो प्रकार होते है जो निम्न है:-
1:-Input-restricted Dequeue
2:-Output-restricted Dequeue.

1:-Input-restricted Dequeue:- इस प्रकार के queue में items को दोनों ends से delete किया जा सकता है परन्तु केवल एक ही end से insert कर सकते है।

Dequeue

2:-Output-restricted Dequeue:-इस प्रकार के queue में items को दोनों तरफ से ही insert किया जा सकता है परन्तु केवल एक ही end से delete कर सकते है।

What is graph in hindi?

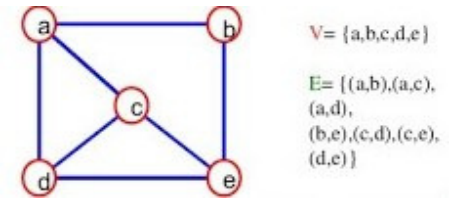
data structure Graphs in hindi:-

ग्राफ को हम निम्न बिंदुओं के आधार पर आसानी से समझ सकते है:-

1:-ग्राफ एक non-primitive, नॉन-लीनियर डेटा स्ट्रक्चर होता है।

2:-ग्राफ एक vertex(node) का समूह होता है। एक vertex दूसरे vertex के साथ जुड़ा रहता है और दो vertex के मध्य connection को हम edge कहते है। Edge दो nodes के मध्य एक कम्युनिकेशन लिंक की तरह कार्य करता है।

3:-ग्राफ (V,E) का समूह होता है जहाँ V, vertex का समूह होता है और E, Edge का समूह होता है।



Types of graph in hindi:-

डेटा स्ट्रक्चर में निम्नलिखित graph के प्रकार होते है:-

1:-Directed graph:- वह ग्राफ जिसमें edges की कोई दिशा(direction) होती है, directed ग्राफ कहलाता है। और इस प्रकार के edges को directed edges कहते है। Directed edges को arces भी कहते है। ग्राफ में edges को एक रेखा के द्वारा दर्शाया जाता है और यदि प्रत्येक रेखा में arrow का निशान बना हुआ होता है तो वह directed ग्राफ कहलाता है। Directed graph को diagraph भी कहा जाता है।

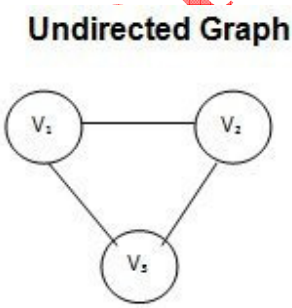


Figure 1: An Undirected Graph

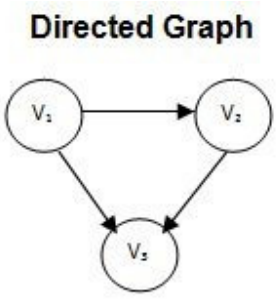
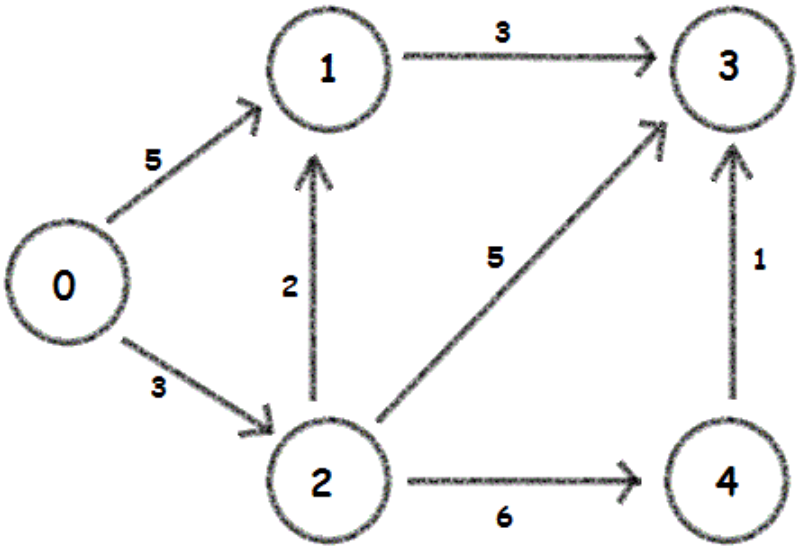


Figure 2: A Directed Graph

2:-Undirected graph:- वह ग्राफ जिसमें edges की दिशा नही होती है अर्थात इसमें arrow का निशान नही बना हुआ होता है। Undirected graph कहलाता है।

3:-Weighted graph and non-weighted graph:-कभी-कभी graphs में edges होते है वे weight को carry करते है। ये weight वास्तविक नंबर होते है। directed और undirected graph दोनों ही weighted ग्राफ हो सकते है।



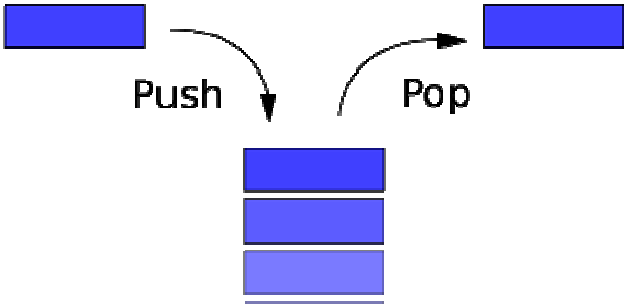
Weighted Directed Graph

वे ग्राफ जो weight को carry नहीं करते है वे ग्राफ non-weighted ग्राफ कहलाता है।

What is stack in hindi?

Stack in hindi:-

स्टैक एक विशेष प्रकार का linear डेटा स्ट्रक्चर होता है जो कि LIFO(last in first out) के सिद्धान्त पर कार्य करता है अर्थात वह item जो कि सबसे अंत में add किया जाता है उसे सबसे पहले remove कर दिया जाता है तथा जो item सबसे पहले add किया जाता है उसे सबसे अंत में remove किया जाता है।



Stack in hindi

स्टैक में किसी item को डालने(add) तथा remove(हटाने) के लिए सिर्फ एक end होता है जिसे हम top end कहते है।

स्टैक में दो ऑपरेशन होते है:- Push तथा Pop.
जब स्टैक में item को insert किया जाता है तो वह push ऑपरेशन कहलाता है तथा जब स्टैक में item को delete किया जाता है तो वह Pop ऑपरेशन कहलाता है।

स्टैक को हम तब overflow कह सकते है जब वह पूरी तरह भरा हुआ होता है तथा तब underflow कह सकते है जब वह पूरी तरह खाली होता है।

What is linked list in hindi?

Linked list in hindi:-

Linked list एक non-primitive, linear डेटा स्ट्रक्चर है।

linked list, नोड्स के समूह से मिलकर बना होता है। प्रत्येक node के दो भाग होते है पहला भाग data का होता है और दूसरा pointer होता है। linked list का pointer भाग अगले node के address को hold किये रहता है।

nodes का प्रयोग डेटा को संग्रहित करने के लिए किया जाता है।

linked list एक ऐसा डेटा स्ट्रक्चर होता है जिसकी length को run-time में बढ़ाया या घटाया जा सकता है। अर्थात यह dynamic होता है।

linked list का प्रयोग tree तथा graph को बनाने के लिए किया जाता है।

Types of linked list in hindi in hindi:- Linked list निम्नलिखित तीन प्रकार के होते है:-

1:-Single linked list:- इसमें one-way direction होता है तथा single linked list के प्रत्येक node में दो fields होते है:-
1:-पहला वह field होता है जहां डेटा स्टोर रहता है।
2:-दूसरा pointer या लिंक होता है।

Figure 1: Element of a singly linked list

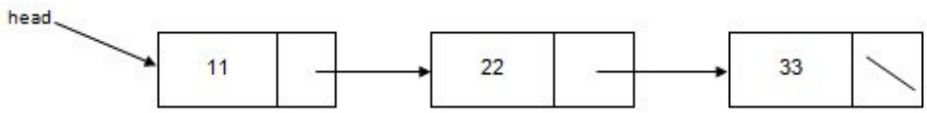


Figure 2: A singly linked list

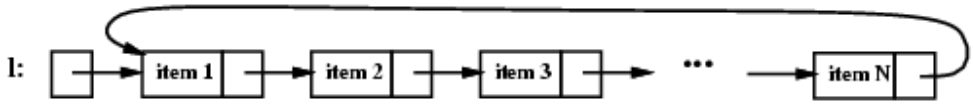
2:-Doubly linked list:- इसमें two-way direction होता है। doubly linked list के प्रत्येक node में तीन भाग होते है:-
1:-पहले भाग में डेटा स्टोर रहता है।
2:-दूसरा भाग अगले नोड के लिए link होता है।
3:-तीसरा भाग पिछले के लिए लिंक होता है।

3:-Circular linked list:- Circular linked list में प्रत्येक नोड circle के रूप में जुड़े रहते है। circular Linked list के अंत में NULL वैल्यू नही होती है।

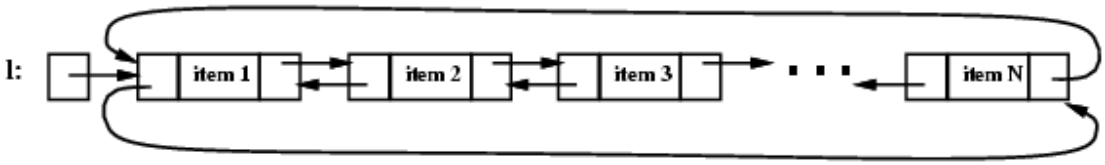
इसमें अंतिम नोड, पहले नोड के address को contain किये हुए रहता है अर्थात पहला और अंतिम नोड adjacent होते है।

Circular linked list के दो प्रकार होते है:-
1:-Single circular linked list
2:-Doubly circular linked list.

Circular, singly linked list:



Circular, doubly linked list:



Applications of binary trees in hindi

application of binary tree in hindi:-

1:-बाइनरी सर्च ट्री का प्रयोग बहुत सारीं सर्च applications में किया जाता है।

2:- Binary Space Partition का प्रयोग आजकल प्रत्येक 3D गेम्स के लिए किया जाता है।

3:-बाइनरी tries का प्रयोग प्रत्येक हाई बैंडविड्थ राऊटर में किया जाता है जो कि राऊटर टेबल्स को स्टोर करता है।

4:-Heaps का प्रयोग queues में इम्प्लीमेंट करने के लिए किया जाता है। रोबोटिक्स तथा वीडियो गेम्स में भी इसका प्रयोग किया जाता है।

5:- Huffman coding tree का प्रयोग compression algorithms(जैसे-.jpeg तथा .mp3) में किया जाता है।

6:- GMG trees का प्रयोग cryptography applications में pseudo-random numbers को generate करने के लिए किया जाता है।

7:- Treap का प्रयोग वायरलेस नेटवर्किंग तथा मेमोरी एलोकेशन में किया जाता है।

8:- T-tree का प्रयोग डेटाबेस में B-tree की तरह डेटा को स्टोर करने के लिए किया जाता है।

Height balanced tree or AVL TREE in hindi

Height balanced tree or AVL TREE in hindi:- AVL TREE एक self balancing binary search tree होती है। AVL TREE को height balanced tree भी कहा जाता है। AVL TREE का नाम इसके inventors(Georgy Adelson-Velsky और Evgenii Landis) के कारण पड़ा। AVL TREE का प्रयोग डेटा को organise करने के लिए किया जाता है।

यदि AVL TREE के N nodes हैं, तो इसकी height log2 (N + 1) होगी।

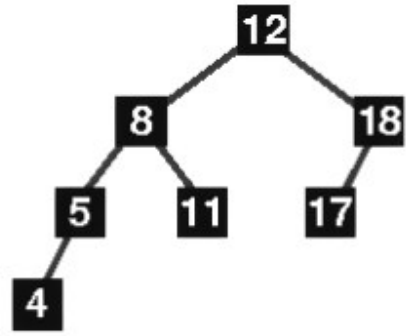


Fig:-AVL tree

एक binary tree तब height balanced होगी जब वह निम्नलिखित rules को satisfy करेगी:-

1:-यदि binary tree का left subtree balanced हो।

2:-यदि Binary tree का right subtree balanced हो।

3:-और, right subtree की height तथा left subtree की height के मध्य अंतर1 से अधिक नही होना चाहिए।

एक empty tree भी height balanced tree होती है।

B-tree in hindi

B-Tree in hindi:-

B-tree एकM-Way(multi-way) tree होता है जो कि विशेषकर disk में प्रयोग करने के लिए बनाया जाता है। B-tree को balanced tree कहा जाता है। एक M-way tree के M children हो सकते है। M-way tree एक node में multiple keys को contain कर सकती है।

यदि M-way tree का एक नोड keys की k संख्या contain करता है तो उस नोड के children की संख्या k+1 होगी।

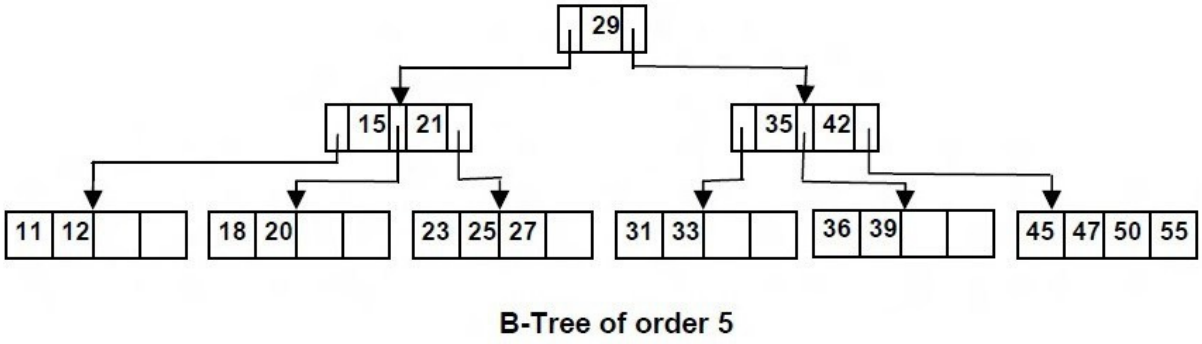


Fig:- B-tree

B-tree हमेशा perfectly height balanced होती है अर्थात B-tree के प्रत्येक leaf node की समान depth होती है

height balanced और weight balanced tree बहुत बड़े डेटा को स्टोर करने के लिए पर्याप्त नही है इसीलिए इस drawback को खत्म करने के लिए B-tree को प्रयोग में लाया जाता है। B-tree का मतलब secondary storage(जैसे-disk) से है। B-tree का प्रयोग ज्यादातर फ़ाइल सिस्टम्स और DBMS में किया जाता है। B-tree डेटाबेस में फाइल्स को locate तथा place करने की method है।

Threaded binary tree in hindi

Threaded binary tree in hindi:-

वह binary tree जिसमें वह प्रत्येक node जिसका कोई child नही होता है उसको पॉइंटर replace कर देता है जिसे हम thread कहते है।

binary tree के representation में leaf node भी होते है जिनमें null value होती है। जिनके कारण memory का waste होता है इसलिए memory wastage के इस drawback को मिटाने के लिए threaded binary tree के concept को develop किया गया।

यदि tree के left child का node रिक्त(null) हो तो यह नोड उस नोड से replace हो जायेगा जो इस रिक्त नोड के पहले वाला नोड होगा।
इसी प्रकार यदि tree के right child का node रिक्त(null) हो तो यह नोड उस नोड से replace हो जायेगा जो इस रिक्त नोड के पिछले वाला नोड होगा।

जो left thread होता है वह predecessor node देता है और जो right thread होता है वह successor node देता है।

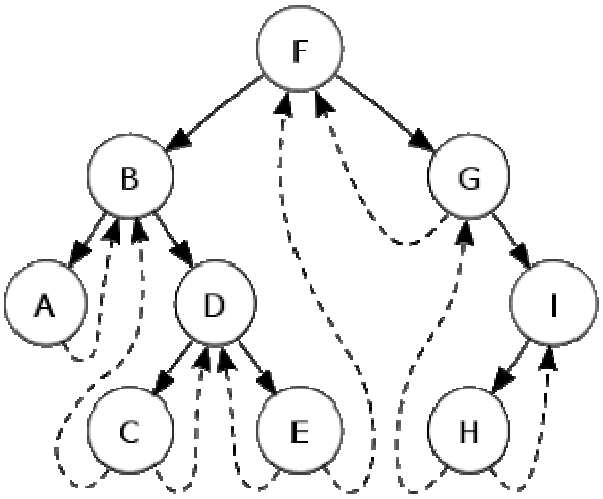


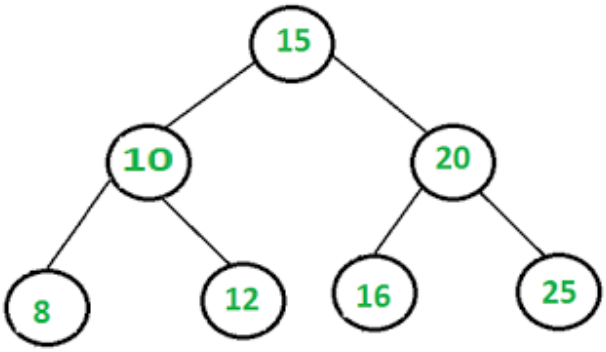
image source:threaded binary tree

Threaded binary tree दो प्रकार के होते है:-
1:-Single threaded binary tree
2:-Double threaded binary tree

Binary tree in hindi

binary tree in hindi:-

Data structure में, Binary tree वह tree है जिसमें प्रत्येक node के केवल अधिकतम दो children होते है। जिन्हें left child और right child कहा जाता है।
जो root node होता है वह सबसे ऊपरी node होता है।

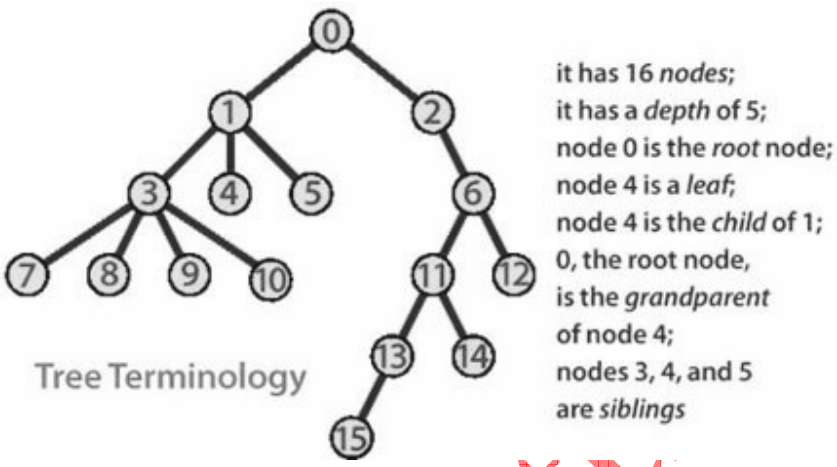


जब tree में एक भी node नही होता है उसे हम empty tree कहते है।

Tree in hindi

Tree in hindi:-

Tree को निम्नलिखित बिंदुओं के आधार पर आसानी से समझ सकते है:-



1:-"ट्री(tree) nodes का एक समूह होती है जिनमें सामान्यतया hierarchical relationship होती है।"

2:-Tree एक non-linear डेटा स्ट्रक्चर होता है।

3:-Tree में parent-child relationship होती है।

4:-Tree के प्रत्येक data item को हम node कहते है।

5:-Tree में जो सबसे ऊपर वाला node होता है उसे हम root node कहते है।

6:-एक node का अधिकतम एक ही parent हो सकता है। लेकिन केवल root नोड का कोई parent नही होता है।

7:-एक tree में प्रत्येक node का शून्य या ज्यादा child nodes हो सकते है।

8:-ऐसे nodes जिनके एक भी child nodes नही होते है उन्हें leaf node या terminal node कहते है।

9:-वैसे तो tree हमेशा ऊपर की ओर बढ़ता है लेकिन data structure का tree हमेशा नीचे की ओर बढ़ता है।

Degree of a node:- किसी tree के ***एक नोड के subtree की संख्या*** degree of a node कहलाती है।

Binary tree traversal in hindi

Binary tree traversal in hindi:-

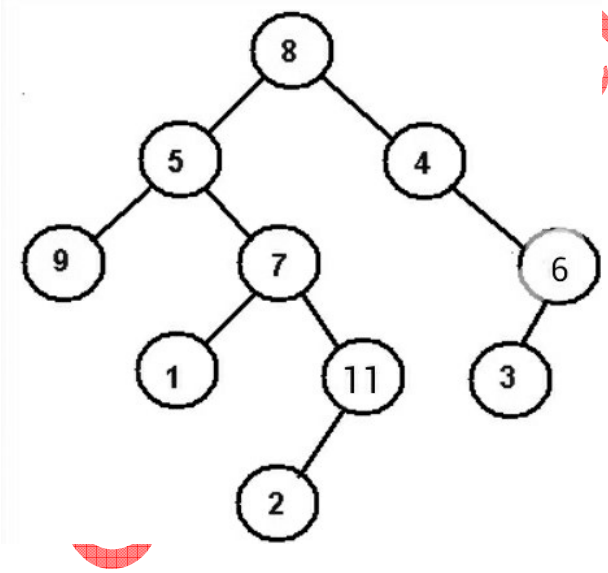
Binary tree के traversal में एक node को सिर्फ एक बार ही visit किया जाता है।

Binary tree के traversal का अर्थ है कि "tree के प्रत्येक node को किसी order में visit करना"

वैसे तो trees को विभिन्न तरीके से traverse किया जाता है लेकिन हम यहां तीन मुख्य traversal की चर्चा करेंगे:-
1:-In-order traversal
2:-Pre-order traversal
3:-post-order traversal

1:-Inorder traversal:-Inorder traversal के लिए निम्नलिखित बिंदु है:-
(1):-Left children या left subtree को traverse किया जाता है।
(2):-Root या parent को visit किया जाता है।
(3):-Right subtree या right children को traverse किया जाता है।

अब हम इसको उदहारण के द्वारा समझ सकते है:-



Binary tree traversal in hindi

इस चित्र का output निम्न है:-
9,5,1,7,2,11,8,4,3,6

2:-Preorder traversal:- Pre-order traversal के लिए निम्नलिखित बिंदु है:-
(1):-सबसे पहले root या parent को visit करते है।

(2):-उसके बाद left subtree या left children को visit करते है।
(3):-उसके बाद right subtree या right children को visit करते है।

उदहारण ऊपर वाले चित्र का output:-
8,5,9,7,1,11,2,4,6,3

3:-Post order traversal:- Post-order traversal के लिए निम्नलिखित बिंदु है:-
(1):-सबसे पहले left subtree या left child को visit करते है।
(2):-उसके बाद right subtree या right child को visit करते है।
(3):-उसके बाद root(parent) को visit करते है।

उदहारण के लिए ऊपर वाले चित्र का output:-
9,1,2,11,7,5,3,6,4,8

dfs depth first search in hindi

DFS (depth first search) in hindi:-

DFS भी BFS की तरह ग्राफ डेटा स्ट्रक्चर को travers तथा search करने की एक अल्गोरिथम है.

BFS में नोड्स को depth wise (गहराई से) विजिट किया जाता है.

डेटा स्ट्रक्चर में DFS को implement करने के लिएstack का प्रयोग किया जाता है.

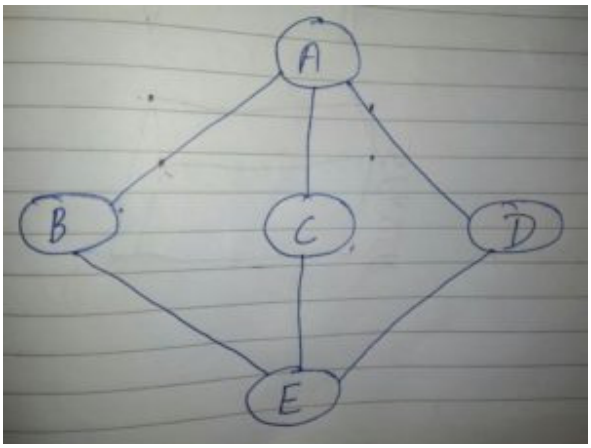
DFS एक recursive अल्गोरिथम है जो कि backtracking के सिधांत पर कार्य करती है.

नेटवर्कों को analyze करने, routes को map करने तथा अन्य कंप्यूटर विज्ञानं की परेशानियों को solve करने के लिए DFS का प्रयोग किया जाता है.

DFS algorithm in hindi:-

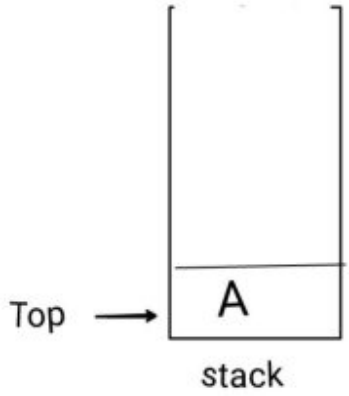
DFS में हम सबसे पहले शुरूआती node को stack के द्वारा विजिट करते है. फिर इसकी समस्त adjacent nodes को stack में डालकर, stack को top में स्थित नोड को विजिट करके, उसके समस्त adjacent node को stack में डाल देते है और प्रक्रिया तब तक दोहराते जब तक कि stack खाली नहीं हो जाता है.

इसकी अल्गोरिथम को निम्नलिखित उदाहरण के द्वारा समझते है.

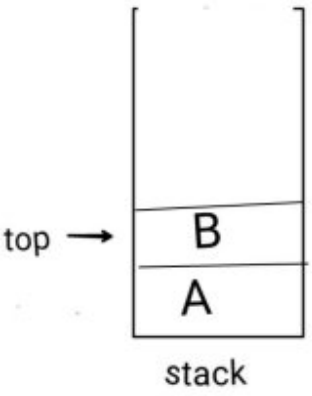


स्टेप 1:– stack को intialize किया जाता है.

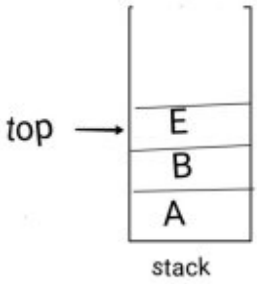
स्टेप 2:– नोड A को visited मार्क करते है और उसे स्टैक में डालते हैं. नोड A के तीन adjacent नोड है B, C, तथा D. हम इनमें से किसी भी नोड को चुन सकते है. हम alphabetical क्रम में चुनेंगें.



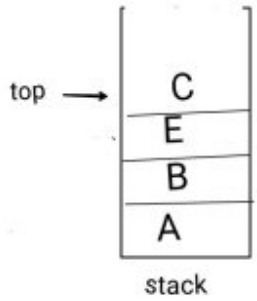
स्टेप 3:- नोड B को visited मार्क करेंगे और स्टैक में डालेंगे. अब B के किसी unvisited adjacent नोड्स को select करेंगे. B के adjacent नोड्स A तथा E है चूँकि A को पहले ही विजिट कर लिया है तो हम E को select करेंगे.



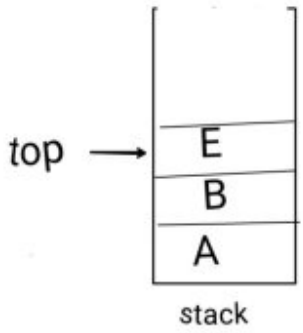
स्टेप 4:- E को विजिट करेंगे और इसे visited मार्क करेंगे और इसे stack में डाल देंगे. यहाँ पर नोड E के दो adjacent नोड C तथा D है दोनों unvisited है तो हम C को select करेंगे. (alphabetical क्रम में.)



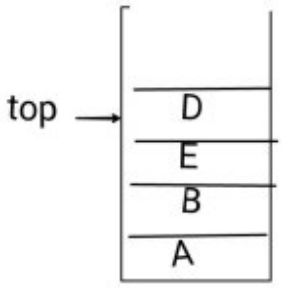
स्टेप 5:- हम c को विजिट करेंगे और इसे visited मार्क करेंगे और स्टैक में रख देंगे. यहाँ पर B का कोई unvisited adjacent नोड नहीं है तो इसे stack से निकाल देंगे.



स्टेप 6:- अब stack के top पर वापस E है अब देखेंगे कि इसका कोई unvisited adjacent node है या नही. इसका adjacent unvisited नोड D है.



स्टेप 7:- अब हम नोड D को विजिट करेंगे और इसे visited मार्क करेंगे और इसे stack में डाल देंगे.



अब विजिट करने के लिए कोई नोड नहीं बचा है अब हम सभी नोड्स को stack में से बाहर निकालेंगे और जब stack empty हो जाएगा तो प्रोग्राम समाप्त हो जाएगा.

BFS (breadth first search) in hindi

Graph traversal in hindi:-

graph traversal का अर्थ है ग्राफ के प्रत्येक node को visit करना. यहाँ पर हम दो प्रकार के traversal की बात करेंगे. जो कि निम्नलिखित है:-
1:- BFS (breadth first search)

2:- DFS (depth first search)

1:- BFS (breadth first search) in hindi:-

BFS ग्राफ डेटा स्ट्रक्चर को travers तथा search करने की एक अल्गोरिथम है.

इसका प्रयोग ग्राफ में shortest path को ढूँढने तथा puzzle गेम्स को solve करने के लिए किया जाता है.

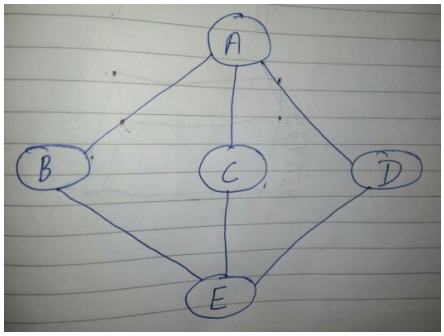
डेटा स्ट्रक्चर में, BFS को implement करने के लिए queue का प्रयोग किया जाता है.

BFS में nodes को breadth wise (चौड़ाई से) visit किया जाता है.

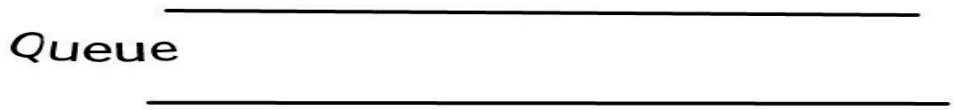
BFS में पहले किसी भी एक node को visit किया जाता है तथा उसके बाद उसके adjacent (नजदीक) के नोड्स को visit किया जाता है. इसके बाद इन adjacent नोड के भी सभी adjacent node को विजिट किया जाता है. और यह प्रक्रिया तब तक चलती है जब तक कि सभी nodes को विजिट नहीं कर लिया जाता है.

BFS algorithm:-

इसकी अल्गोरिथम को निम्नलिखित उदहारण के द्वारा समझते है. माना कि हमारे पास निम्नलिखित ग्राफ है जिसे हमें traverse करना है।

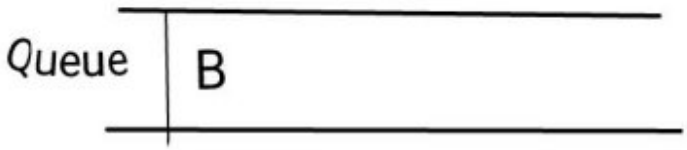


स्टेप 1:- queue को initialize किया जाता है.

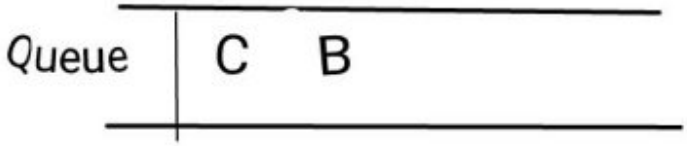


स्टेप 2:– सबसे पहले हम node A (शुरूआती नोड) को विजिट करते है और इसे visited मार्क करते है.

स्टेप 3:– इसके बाद हम A के adjacent nodes को देखते है. इसके adjacent नोड्स B, C तथा D है. इस चित्र में हम सबसे पहले B को विजिट करते है और उसे queue में रखते है.



स्टेप 4:– इसके बाद हम नोड A के adjacent node C को विजिट करते है और उसे queue में रखते है.



स्टेप 5:- इसके बाद A के अंतिम adjacent नोड D को विजिट करते है और उसे queue में रखते है.

स्टेप 6:- इसके बाद में A के कोई adjacent नोड नहीं बचे इसलिए हम B को queue से निकालते है और उसके adjacent को search करते है.

स्टेप 7:- नोड B का adjacent नोड E है तो हम E को विजिट करते है और उसे queue में रखते है.

अब हमारे पास विजिट करने के लिए कोई भी नोड बही बचा है परन्तु हमें सभी nodes को queue से निकालना होगा. और जब queue खाली हो जायेगा तो प्रोग्राम समाप्त हो जायेगा.

data structure searching in hindi & its types linear & binary searching in hindi

Data structure searching in hindi:-

Searching (सर्चिंग) :- जैसा कि आपको पता ही होगा सर्चिंग का अर्थ है "ढूंढना" या "खोजना" .

डेटा स्ट्रक्चर में 'searching' वह प्रक्रिया है जिसमें किसी element को लिस्ट में खोजा जाता है जो कि एक या एक से अधिक condition को संतुष्ट करता हो.

types of searching in hindi:-

डेटा स्ट्रक्चर में searching के लिए हम दो तकनीकों का प्रयोग करते हैं जो कि निम्नलिखित हैं:-

1:- linear search (लीनियर सर्च)
2:- binary search (बाइनरी सर्च)

1:- Linear search in hindi:-

इसको sequential search भी कहते है.

इस searching तकनीक में दिए गये डेटा element को तब तक एक एक करके लिस्ट के प्रत्येक element के साथ compare किया जाता है जब तक कि element मिल नहीं जाता.

इसमें सबसे पहले दिए गये element को लिस्ट के प्रथम element के साथ compare किया जाता है यदि दोनों element एक समान है तो वह index value रिटर्न करता है नहीं तो -1 रिटर्न करता है.
फिर इसके बाद दिए गये element को लिस्ट के दुसरे element के साथ compare किया जाता है. यदि दोनों element समान है तो वह index value रिटर्न करता है नहीं तो -1 रिटर्न करता है.

इसी प्रकार पूरी लिस्ट को compare किया जाता है जब तक कि element मिल नहीं जाता है. अगर पूरी लिस्ट compare करने के बाद भी element नहीं मिलता है तो सर्च unsuccessful हो जाएगा.

यह सबसे सरल searching तकनीक है परन्तु इसमें समय बहुत लगता है. क्योंकि linear search की औसत case complexity O(n) है.

linear search algorithm:-

```
step1          i=1 {i=0}
step2           if i>n, go to step 7
step3           if A[i]=x, go to step 6
step4           i=i+1
step5           go to step 2
step6           return i
step7           return -1
step8           exit
```

उदाहरण के द्वारा हम इसे आसानी से समझ सकते है.

माना कि हमारे पास निम्नलिखितarray लिस्ट है.

21 70 15 30 56 78 80

और हमें इसमें 30 को खोजना है.

step1:- दिए गये element (30) को लिस्ट के प्रथम element (21) के साथ compare (तुलना) किया जाता है.

21 70 15 30 56 78 80

दोनों एकसमान नहीं है तो हम दुसरे element में जायेंगे.

step2:- 30 की लिस्ट के दुसरे element (70) के साथ तुलना करेंगे

21 70 15 30 56 78 80

दोनों एकसमान नहीं है तो हम अगले element में जायेंगे.

step3:- 30 की तुलना 15 के साथ करेंगे.

21 70 15 30 56 78 80

दोनों एक समान नहीं है तो हम अगले element में जायेंगे.

step4:- अब हम दिए गये element (30) की तुलना अगले element 30 के साथ करेंगे.

21 70 15 30 56 78 80

दोनों एक समान है तो हम तुलना करना बंद कर देंगे और index 3 रिटर्न करेंगे.

Binary search in hindi:-

जब कोई बड़ा डाटा स्ट्रक्चर होता है तो linear search में बहुत अधिक समय लग जाता है. इसलिए linear search की कमी को दूर करने के लिए binary search को विकसित किया गया.

binary search बहुत ही तेज searching अल्गोरिथम है जिसकी time complexity O(log n) है. यह divide & conquer सिद्धांत पर आधारित है.

binary search केवल उसी लिस्ट में की जा सकती है जो कि sorted (क्रमानुसार) हों. इसका प्रयोग ऐसी लिस्ट में नहीं कर सकते जो कि sorted order में नहीं है.

इस सर्चिंग तकनीक में दिए गये element की लिस्ट के middle element के साथ तुलना की जाती है. यदि दोनों एकसमान है तो वह index value रिटर्न करता है.
यदि एक समान नहीं है तो हम check करते है कि दिया गया element जो है वह middle element से बड़ा है या छोटा.
यदि वह छोटा है तो हम लिस्ट के छोटे भाग में यही प्रक्रिया दोहराएंगे.
और यदि वह बड़ा है तो हम लिस्ट के बड़े भाग में यही प्रक्रिया दोहराएंगे. और यह तब तक करेंगे जब तक कि element मिल नहीं जाता.

उदाहरण:- इसको हम भलीभांति उदाहरण के द्वारा समझ सकते है.
माना हमारे पास निम्नलिखित array लिस्ट है.

3 5 11 17 25 30 32

हमें दिया गया element 5 है जिसे हमने लिस्ट में ढूंढना है.

step1:- सबसे पहले हम दिए गये element 5 की तुलना middle element 17 से करते है.

3 5 11 17 25 30 32

दोनों एकसमान नहीं है और 5 जो है वह 17 से छोटा है.

तो हम लिस्ट के बाएं वाले भाग (छोटे वाले भाग) में ही search करेंगे.

3 5 11

step2:- दिए गये element 5 को middle element 5 के साथ compare करेंगे.

3 5 11

दोनों एकसमान है तो हम तुलना करना बंद कर देंगे. और `index 1` रिटर्न करेंगे.

इसे भी पढ़ें:- data structure operations

c operator in hindi & types of operators in hindi

Operators in hindi (‘सी’ भाषा में ऑपरेटर्स क्या है?):-

किसी भी प्रोग्रामिंग भाषा में प्रयोग किये जाने वाले operators वे संकेत होते हैं जो कि कंप्यूटर कम्पाइलर को गणितीय या लॉजिकल संगणनाएं करने के लिए निर्देश देते है.

सी भाषा में भी operator का प्रयोग गणना करने तथा निर्णय लेने के लिए ही किया जाता है. operator का प्रयोग वेरिएबल अथवा संख्याओं के साथ किया जा सकता है.

types of operators in hindi (ऑपरेटर्स के प्रकार):-

‘सी’ प्रोग्रामिंग भाषा में operators के निम्नलिखित प्रकार होते है:-

1:- Arithmetic Operator (अरिथमेटिक ऑपरेटर)

2:- Relational Operator (रिलेशनल ऑपरेटर)

3:- Logical Operator (लॉजिकल)

4:- Bitwise Operator (बिटवाइज)

5:– Assignment Operator (असाइनमेंट)

6:– increment & decrement operators (इन्क्रीमेंट तथा डिक्रिमेंट)

7:- अन्य ऑपरेटर्स

1:- arithmetic operators (अंकगणितीय ऑपरेटर):-

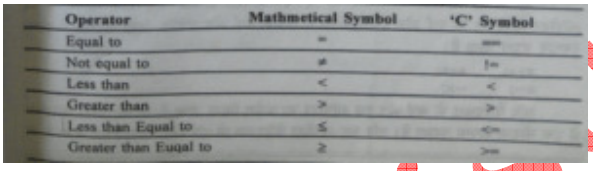
arithmetic ऑपरेटर्स का प्रयोग आंकिक गणनाओं के लिए किया जाता है. ‘सी’ में arithmetic ऑपरेटर + का प्रयोग जोड़ (addition) के लिए, – का प्रयोग घटाने (substraction) के लिए, * का प्रयोग गुणा (multiply) के

लिए, / का प्रयोग भाग (dividation) तथा % का प्रयोग भाग-अवशेष (modulo division) के लिए किया जाता है.

Arithmetic Operators/Operation	Example
+ (Addition)	A+B
– (Subtraction)	A-B
* (multiplication)	A*B
/ (Division)	A/B
% (Modulus)	A%B

2:- Relational operators (रिलेशनल ऑपरेटर):-

जब दो संख्याओं में असमानता अथवा समानता प्रकट करते हुए लिखना होता है तब हम रिलेशनल ऑपरेटर का प्रयोग करते है. 'सी' भाषा में प्रयोग होने वाले रिलेशनल ऑपरेटर निम्नवत हैं:-



Operator	Mathmetical Symbol	'C' Symbol
Equal to	=	==
Not equal to	≠	!=
Less than	<	<
Greater than	>	>
Less than Equal to	≤	<=
Greater than Euqal to	≥	>=

3:- logical operators (लॉजिकल ऑपरेटर्स):-

'सी' में लॉजिकल ऑपरेटर का प्रयोग variables में लॉजिकल ऑपरेशन करने के लिए किया जाता है.

operators	Example/Description
&& (logical AND)	(a>6)&&(b<6) यह true दिखाता है यदि दोनों कंडीशन सत्य(true) हो तो.
|| (logical OR)	(a>=12)||(b>=12) यह true दिखाता है यदि एक कंडीशन सत्य हो तो.
! (logical NOT)	!((a>6)&&(b<6)) यह true return करता है जब conditions satisfy नहीं होती है तो

4:- assignment operators (असाइनमेंट ऑपरेटर):-

जब किसी वेरिएबल को मान प्रदान किया जाता है, तो असाइनमेंट operator का प्रयोग किया जाता है. 'सी' भाषा में यह ऑपरेटर (=) है.

int x = 5;

इस ऑपरेटर के साथ अंकगणितीय ऑपरेटर (+, -, *,/ तथा, %) का प्रयोग करके बहुत छोटे स्टेटमेंट द्वारा वेरिएबल को मान प्रदान किया जा सकता है. जैसे, यदि हमें लिखना है-

Int x = x + 5;

इस स्टेटमेंट को हम इस प्रकार भी लिख सकते हैं.

int x += 5;

5:- bitwise operators (बिटवाइज ऑपरेटर):-

bit लेवल के ऑपरेशन करने के लिए c लैंग्वेज में बिटवाइज ऑपरेटर का प्रयोग किया जाता है.

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise complement
<<	Shift left
>>	Shift right

6:- increment & decrement operators (इन्क्रीमेंट तथा डिक्रिमेंट ऑपरेटर):-

ये ऑपरेटर्स किसी एक operand पर ही कार्य करते हैं. इनको unary operator भी कहते है.

जब हमें किसी वेरिएबल में से एक घटाना अथवा एक जोड़ना हो तो हम इन्क्रीमेंट अथवा डिक्रिमेंट ऑपरेटर का प्रयोग करते हैं.

'सी' में यह ऑपरेटर '–' और '++' है. इस operator में यह ध्यान रखना चाहिए कि ऑपरेटर वेरिएबल के दायीं ओर प्रयोग करना है अथवा बायीं ओर क्योंकि दिशा बदलने से इनका स्वभाव बदल जाएगा.

a++;

++a;

a–;

–a;

यदि वेरिएबल के बायीं और इस ऑपरेटर का प्रयोग किया जाता है, तो यह पहले वेरिएबल में एक जोड़ता अथवा घटाता है. यदि ऑपरेटर वेरिएबल के दाई ओर प्रयोग किया जाता है, तो यह ऑपरेटर बाद में घटाता अथवा जोड़ता है इसे इस प्रकार समझा जा सकता है मान लेते हैं कि a = 4 और b = 0 है तो-

b = ++a;

इस स्टेटमेंट में पहले a में एक जुड़ने के बाद वह मान b को भी प्रदान हो जाएगा. अब a और b दोनों वेरिएबल्स का मान 5 हो जाएगा.

यदि इस स्टेटमेंट को इस प्रकार लिखते हैं:-

b = a++;

इस स्टेटमेंट में पहले b को वेरिएबल a का मान प्राप्त होगा और उसके बाद a में एक जुड़ेगा. इस प्रकार b का मान 4 और a का मान 5 हो जाएगा.

7:- अन्य operators:-

सी भाषा में & ऑपरेटर का प्रयोग किसी भी वेरिएबल के एड्रेस को एक्सेस करने के लिए प्रयुक्त होते है.

sizeof ऑपरेटर का प्रयोग वेरिएबल के साइज़ को एक्सेस करने के लिए किया जाता है.

expression tree in hindi prefix, infix & postfix in hindi

Expression tree in hindi (एक्सप्रेशन ट्री):-

ऐसे ट्रीज, जिनमें किसी समीकरण को प्रस्तुत किया जाता है, expression tree कहलाते है. इन ट्रीज में operands और operators दो प्रकार की सूचनायें होती हैं.

operands (ओपरेंड्स):- वे तत्व जिन पर कार्य किया जाता है.

operators (ऑपरेटर्स):- वे तत्व जिनके द्वारा operands पर कार्य किया जाता है.

अच्छी तरह समझने के लिए निम्न उदाहरण देखें.

(k-a) + (b*d^f)

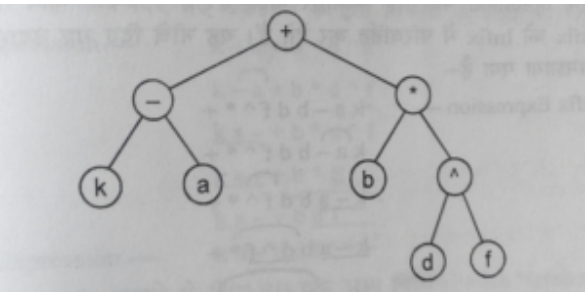
इसमें -, +, *, तथा ^ ऑपरेटर्स हैं और p, a, b, c तथा n operands है.

इस प्रकार के ट्रीज में प्रयुक्त किये जाने वाले operators की priorty (स्थान) निश्चित कर दिया गया है:-

Highest priority (सबसे ऊँचा स्थान): ^ (घात)

higher priority (ऊँचा स्थान): * (गुणा), / (भाग)

lower priority (नीचा स्थान): + (जोड़), – (घटाना)



ऊपर दिए गये expression tree का

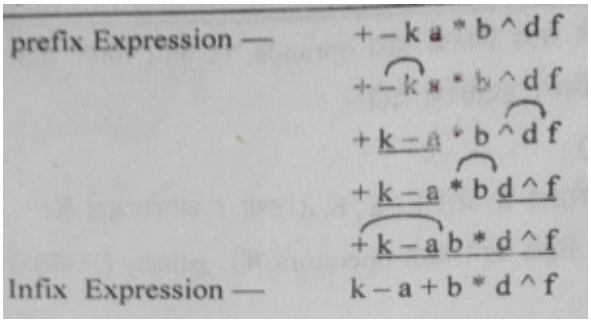
infix expression: k-a+b*d^f

prefix expression: +-ka*b^df

post fix expression: ka-bdf^*+

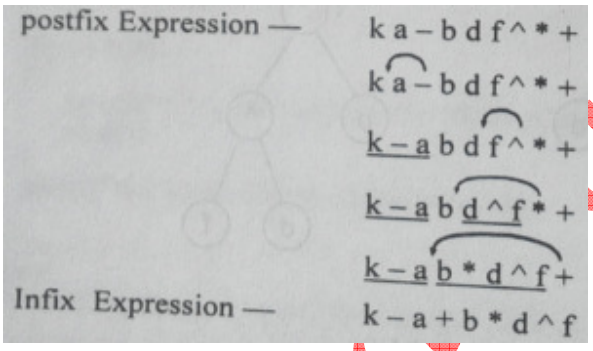
दिए गये prefix expression को infix expression में बदलना (change prefix expression tree to infix expression in hindi):-

किसी भी prefix expression को infix expression में बदलने के लिए expression को बाएं से दायें की और trace करते हैं और वे जब लगातार दो operands प्राप्त होते है, तो उनसे बिलकुल पहले प्राप्त हुए operator को दोनों operands के मध्य में रखकर इन तीनों में एक operand की तरह व्यवहार करते हैं.
इसी प्रकार आगे तक trace करते हुए prefix को infix में परिवर्तित कर देते हैं. यह आगे दिए गये उदाहरण के द्वारा विस्तार से समझाया गया है:-



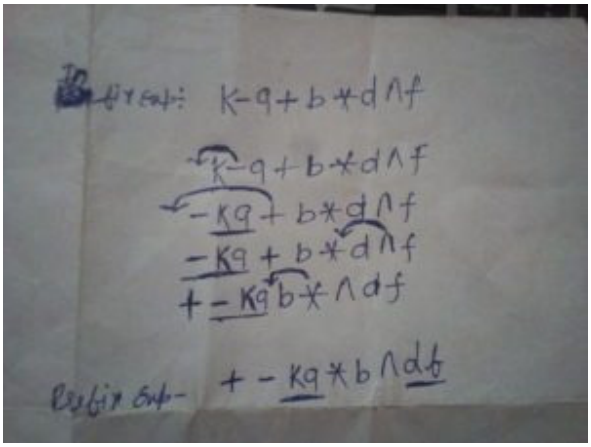
दिए गये postfix expression को infix expression में बदलना (change postfix expression to infix expression tree in hindi):-

किसी भी postfix expression को infix expression में बदलने के लिए expression को दायें से बाएं की और trace करते है और जब वे लगातार दो operands प्राप्त होते हैं, तो उनसे बाद प्राप्त हुए operator को दोनों operands के मध्य रखकर इन तीनों में से एक operand की तरह व्यवहार करते है. इसी प्रकार आगे तक trace करते हुए postfix को infix में परिवर्तित कर देते है. यह निचे दिए उदाहरण के द्वारा विस्तार से समझाया गया है:-



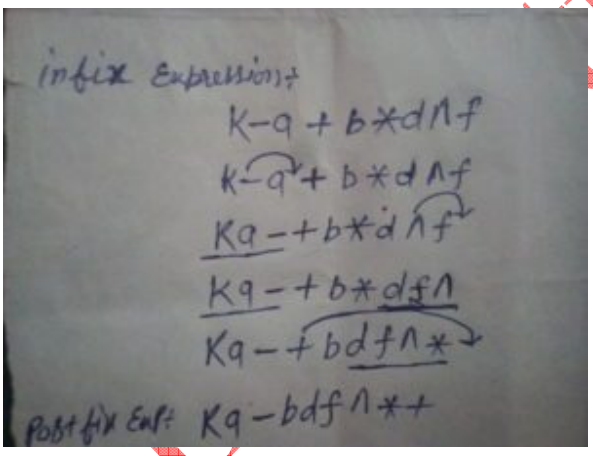
दिए गये infix expression को prefix expression में बदलना (change infix expression tree to prefix expression in hindi):-

किसी भी infix expression को prefix expression में बदलने के लिए expression को बाएं से दायें की और trace करते हैं. चूँकि prefix में root पहले होता है इसलिए पहले दो operands, जिनके बीच कोई operator भी हो, के बीच में से operator को निकालकर दोनों operands के शुरू में लगा देते हैं और इन तीनों से एक operand की तरह ही व्यवहार करते हैं.
इसी प्रकार आगे तक trace करते हुए infix को prefix में परिवर्तित कर देते हैं. यह नीचे दिए गये उदाहरण के द्वारा विस्तार से समझाया गया है:-



दिए गये infix expression को postfix expression tree में बदलना (change infix expression to postfix expression in hindi):-

किसी भी infix expression को prefix expression में बदलने के लिए expression को बाएं से दायें की और trace करते हैं. चूँकि postfix में root अंत होता है इसलिए पहले दो operands, जिनके बीच कोई operator भी हो, के बीच में से operator को निकालकर दोनों operand के बाद में लगा देते हैं.
और इन तीनों से एक operand की तरह ही व्यवहार करते हैं. इसी प्रकार आगे तक trace करते हुए infix को prefix में परिवर्तित कर देते हैं. यह निचे दिए गये उदाहरण के द्वारा विस्तार से समझाया गया है:-



pointer in c hindi

'C' Pointer in hindi (पॉइंटर):-

pointer वह variable है जो किarray के address को contain किये रहता है.
या
पॉइंटर वह variable है जो कि दूसरें variable के address को contain किये रहता है.

address किसी भी साधारण वेरिएबल में सुरक्षित नहीं हो सकते, इनको सुरक्षित करने के लिए केवल pointers ही प्रयोग में लाये जाते हैं.

pointer पर किये जा सकने वाले वैध कार्य निम्नलिखित हैं:-

1:- एक cast ऑपरेटर को प्रयुक्त करके समान डेटा प्रकार के पॉइंटर्स को assign करना.

2:- एक pointer को किसी integer के साथ जोड़ना अथवा घटाना.

3:- किन्ही दो pointers का अन्तर ज्ञात करना अथवा उनकी तुलना करना जोकि एक ही array को point करते हों.

4:- किसी भी पॉइंटर को NULL assign करना अथवा NULL से तुलना करना.

pointer को प्रयोग करने हेतु विशेष निर्देश:-

'C' भाषा में पॉइंटर का विशेष महत्व है, pointer के प्रयोग से हम variable द्वारा used bytes की स्थितियां सुनिश्चित कर सकते हैं, जिससे प्रोग्राम की जटिलता और कम हो जाती है.

'C' प्रोग्रामिंग भाषा में pointers के प्रयोग से प्रोग्राम के कार्यान्वयन अधिक हो जाती है. साथ ही यदि pointer का सही प्रयोग ना किया जाए तो यह बहुत बड़ी गलती का कारण बन सकते हैं.
अतः पॉइंटर को किसी प्रोग्राम में प्रयोग करने से पहले यह सुनिश्चित कर लेना चाहिए कि पॉइंटर प्रोग्राम में सही स्थान पर प्रयोग किया है अथवा नहीं.

यहाँ पर प्रोग्रामिंग के दौरान pointer के प्रयोग में सामन्यतया होने वाली गलतियों तथा सावधानियों के बारें में बताया गया है. जो निम्न है:-

1:- पॉइंटर variable को घोषित करने के लिए हम पॉइंटर variable से पहले स्टार का प्रयोग करते है.

2:- जब तक पॉइंटर variable को मान प्रदान नहीं किया जाता वह garbage है अर्थात् उस पॉइंटर variable के मान का कोई महत्व नहीं है.

3:- पॉइंटर मेमोरी में used bytes का address कहलाता है.

4:- सामन्यतया कंप्यूटर में मैमोरी address शून्य से शुरू होते हैं. मैमोरी में सेलों की अधिक संख्या कंप्यूटर के प्रकार पर निर्भर करती है.

5:- सामन्यतया पॉइंटर में मुख्य गलती ampersand (&) ऑपरेटर की पायी जाती है जब पॉइंटर variable को मान प्रदान करते है तो हम सामान्य वेरिएबल से ampersand (&) का प्रयोग करना भूल जाते हैं. जिससे pointer में variable का address स्टोर नहीं हो पाता.

insertion sort in hindi & its algorithm hindi

Insertion sort in hindi:-

insertion sort भी एक सरल sorting तकनीक है तथा यह छोटे डेटा सेट्स के लिए सबसे उपयुक्त है. परन्तु यह बड़े डेटा सेट्स के लिए उपयुक्त नहीं है.

इस तकनीक में हम एक element को pick करते है और उसे उसके appropriate स्थान पर insert कर देते है.

इसकी औसत case complexity:- $O(n^2)$ होती है. जहाँ n, elements की संख्या है.

insertion sort एक तेज सॉर्टिंग algorithm नहीं है क्योंकि यह nested loops का प्रयोग elements को अपनी जगह में शिफ्ट करने में करती है. परन्तु यह bubble sort तथा selection sort से अच्छी सॉर्टिंग तकनीक है क्योंकि insertion sort की complexity इन दोनों से कम है.

अगर हमारे पास n elements है तो हमें उसे sort करने के लिए (n-1) pass की आवश्यकता होगी.

जिस प्रकार हम ताश के पत्तों को क्रम में arrange करते है उसी प्रकार यह सॉर्टिंग भी कार्य करती है.

insertion sort algorithm in hindi:-

insertion sort की algorithm निम्नलिखित है.

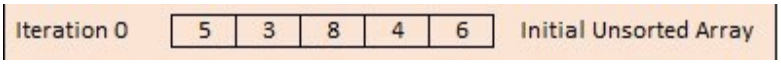
```
                INSERTION_SORT (A)
    1.  FOR j ← 2 TO length[A]
2.      DO key ← A[j]
3.          {Put A[j] into the sorted sequence A[1 . . j − 1]}
4.          i ← j − 1
5.          WHILE i > 0 and A[i] > key
6.                  DO A[i +1] ← A[i]
```

7. $i \leftarrow i - 1$

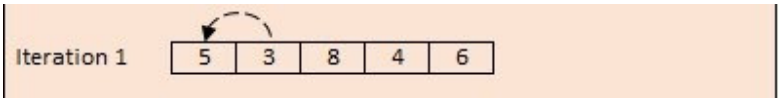
8. $A[i + 1] \leftarrow key$

Insertion sort example in hindi:-

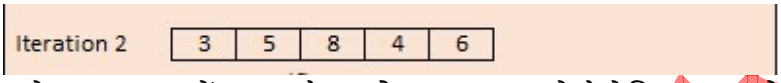
माना कि हमारे पास निम्नलिखितarray है जिसे हमने sort करना है.



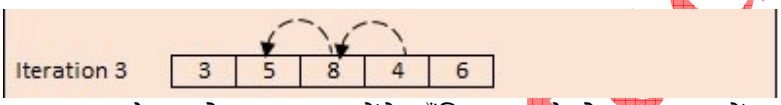
insertion sort में पहले के दो elements को compare किया जाता है.



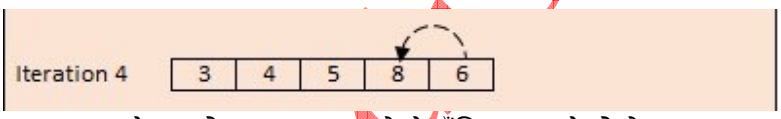
चूँकि 5 > 3 है तो यह आपस में अपना स्थान बदल लेंगे अर्थात् swap हो जायेंगे.



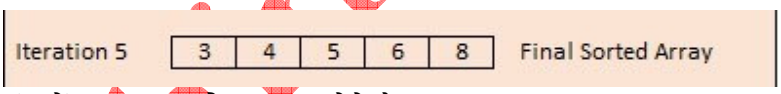
आगे इस sorting में हम 5 और 8 को compare करते है लेकिन यह पहले से ही sorted है इसलिए इसमें कोई परिवर्तन नहीं होगा.



अब हम 8 और 4 को compare करेंगे. चूँकि 8 > 4 है तो यह आपस में swap हो जायेंगे. लेकिन अब इसमें 5 और 4 unsorted हो जायेंगे इसलिए इन्हें भी आपस में swap करके sorted कर लिया जाता है.



अब हम 8 और 6 को compare करते है चूँकि 8 > 6 है तो ये आपस swap हो जायेंगें.



अंत में हम array को sort कर लेते है.

quick sort in hindi & its algorithm in hindi

quick sort in hindi:-

quick sort भी merge sort की तरह एक divide & conquer अल्गोरिथम पर आधारित सॉर्टिंग तकनीक है. इसे 1960 में Tony Hoare द्वारा विकसित किया गया था.

इस सॉर्टिंग तकनीक मेंarrays के elements को दो छोटे arrays में विभाजित किया जाता है.
quick sort जो है वह InPlace सॉर्टिंग का एक प्रकार है.

इस सॉर्टिंग में, सबसे पहले लिस्ट में से किसी भी element को select किया जाता है जिसे हम pivot कहते है. pivot से छोटे elements इसके बाएं तरफ रहेंगें. जबकि pivot से बड़े elements इसके दायीं तरफ रहेंगे.

quick sort की औसत complexity:- O (n log n) है.
तथा इसकी worst case complexity:- O (n^2) है जहाँ n, elements की संख्या है.
क्योंकि worst case में भी quick sort की complexity कम होती है इसलिए यह बहुत तेज तथा efficient है.

quick sort algorithm in hindi:-

इस सॉर्टिंग की algorithm निम्नलिखित है.

step1:- array लिस्ट में एक element को select करते है जिसे हम pivot वैल्यू कहते है.

step2:- elements को इस प्रकार दूबारा arrange करते है कि वे सभी elements जो pivot वैल्यू से छोटी है वे arrays के बायीं तरफ रहती है और वे सभी elements जो pivot वैल्यू से बड़ी होती है उन्हें array के दायीं तरफ रखा जाता है. और वह element जो pivot के समान होते है उन्हें array में किसी भी तरफ रखा जा सकता है.

step3:- array के दोनों भागों को सॉर्ट किया जाता है. दोनों भागों को दुबारा quick sort algorithm का प्रयोग करके सॉर्ट किया जाता है.

Quick sort example:-

selection sort in hindi & algorithm in hindi

selection sort in hindi:-

selection sort बहुत ही सरल तकनीक है. इस सॉर्टिंग algorithm में सबसे पहले array में से सबसे छोटे element को select किया जाता है तथा इस element को array में जो पहले स्थान पर element होता है उसके साथ बदल दिया जाता है. इसके बाद जो दूसरा अगला छोटा element होता है उसे select किया जाता है तथा उसे array में दूसरे स्थान वाले element के साथ बदल दिया जाता है और यह तब तक चलता रहता है जब तक कि पूरी array sort नहीं हो जाती है.

selection sort की case complexity:- O(n^2) है. जहाँ n, elements की संख्या है.

इस sorting तकनीक में लिस्ट दो भागों में विभाजित की जाती है.

पहला भाग sorted भाग होता है जिसको बाएं तरफ लिखा जाता है.

दूसरा भाग unsorted भाग होता है जिसे दायें तरफ लिखा जाता है.

selection sort algorithm in hindi:-

इस algorithm में निम्नलिखित steps होते है:-

step 1:- लिस्ट में सभी unsorted elements को compare किया जाता है तथा सबसे छोटे element को select किया जाता है उसे लिस्ट के पहले element के साथ बदल दिया जाता है.

step 2:- दूसरे सबसे छोटे element को select किया जाता है उसे दूसरे लिस्ट के दूसरे element के साथ बदल दिया जाता है.

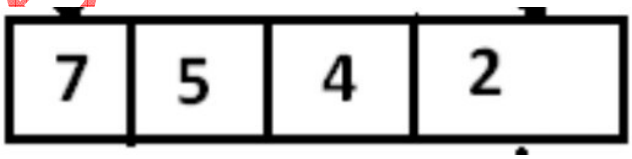
step 3:- तीसरे सबसे छोटे element को select किया जाता है use तीसरे element के साथ बदल दिया जाता है.

step 4:- इस प्रकार यह क्रम चलते रहता है जब तक कि पूरी array लिस्ट sort ना हो जाएँ.

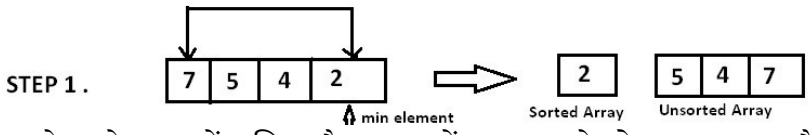
इस algorithm को selection sort इसलिए कहा जाता है क्योंकि इसमें लगातार अगले छोटे element को select किया जाता है और उसे बदल (swap) दिया जाता है.

example of selection sort in hindi:- सिलेक्शन सॉर्ट का उदाहरण

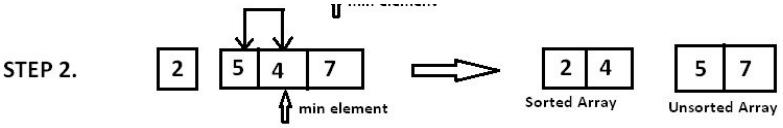
माना कि हमारे पास यह निम्न array है:-



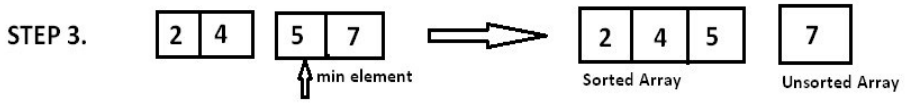
इस unsorted array में सबसे पहले स्थान पर 7 स्टोर है. हम पूरे लिस्ट में ढूँढते है तो हमें सबसे न्यूनतम वैल्यू 2 प्राप्त होती है तो हम इस 2 को 7 के साथ बदल देते है.



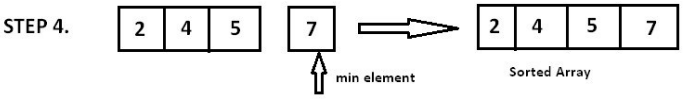
array के दूसरे स्थान में 5 स्थित है तथा इसमें दूसरा सबसे छोटा element 4 है तो हम इस 4 को 5 के साथ बदल दिया जाता है.



array में तीसरे स्थान पर 5 स्थित है परन्तु इसमें कोई परिवर्तन नहीं होगा क्योंकि इससे छोटा कोई element नहीं बचा है अर्थात यह पहले से ही sorted है.



अब हम array में चोथें स्थान पर देखते है तो वहां पर 7 है इसमें भी कोई परिवर्तन नहीं होगा क्योंकि यह लिस्ट पहले से ही sorted है.



निवेदन:- आपको यह selection sort की पोस्ट कैसी लगी हमें comment के द्वारा बताइए तथा इस पोस्ट को अपने दोस्तों के साथ जरुर share करें. धन्यवाद

---

heap sort in hindi & algorithm in hindi

Heap sort in hindi:-

heap sort पढने से पहले हम heap क्या होता है वह पढेंगे.

Heap एक tree पर आधारित डेटा स्ट्रक्चर है जिसकी कुछ विशेष गुणधर्म होते है.

heap की निम्न बेसिक जरूरतें है:-

- heap डेटा स्ट्रक्चर हमेशा एक complete binary tree (CBT) होता है अर्थात tree के सभी स्तर पूरी तरह से भरे हुए हों.

- प्रत्येक नोड में जो वैल्यू रखी है वह अपने दो children से बड़ी या बराबर होगी, इस heap को हम max heap कहते है.

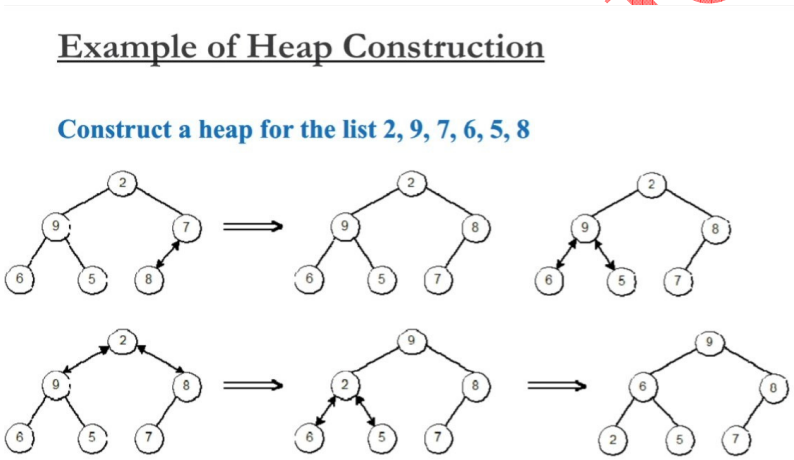
या

प्रत्येक नोड में जो वैल्यू रखी है वह अपने दो children से छोटी या बराबर होगी, इस heap को हम min heap कहते है.

अगर हम लिस्ट को ascending order (बढ़ते क्रम) में sort करना चाहते है तो हम min heap को create करते है.

अगर हम लिस्ट को descending order (घटते क्रम) में sort करना चाहते है तो हम max heap को create करते है.

heap sort की complexity merge sort की तरह O (n log n) होती है.



heap sort algorithm in hindi:-

यह algorithm max heap sort के लिए है.

step 1:- heap में नया नोड बनाओ.

step 2:- नोड को एक वैल्यू assign करो.

step 3:- child नोड की वैल्यू को parent नोड की वैल्यू के साथ compare करो.

step 4:- यदि parent node < child node से तो उन्हें आपस में बदल दो (swap कर दो)

step 5:- स्टेप 3 तथा 4 को तब तक repeat करो जब तक कि heap सही ढंग से न बन जाएँ.

merge sort in hindi & its example in hindi

Merge sort in hindi:-

merge sort जो है वह divide & conquer तकनीक का प्रयोग करता है. divide & conquer तकनीक को जॉन नयूमन्न ने 1945 में प्रस्तावित किया था.

divide & conquer एक ऐसी तकनीक है जिसमें डेटा की एक comlex (कठिन) list को sub-list में विभाजित कर लिया जाता है और इस प्रकार लिस्ट को तब तक विभाजित किया जाता है जब तक कि लिस्ट में केवल एक element बचें.
इसके बाद इन sub-lists को sort करके combine कर दिया जाता है.

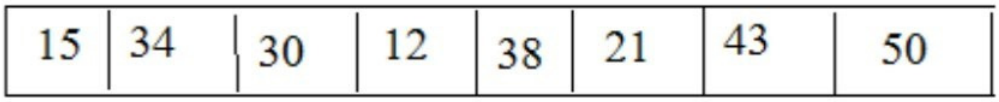
merge sort को two way sort भी कहते है.

merge sort की time complexity **O(n log n)** होती है जिस कारण merge sort को बहुत अच्छी algorithm समझा जाता है.

merge sort example in hindi:-

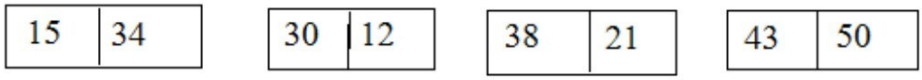
इस sort को समझने के लिए हम निम्नलिखित उदाहरण लेते है:-

यह एक unsorted array है:-

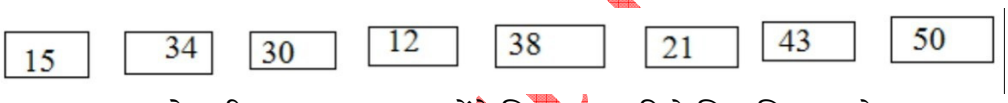


जैसा कि हम जानते है कि merge sort में सबसे पहले पूरे array को आधे भाग में विभाजित किया जाता है. हमारे पास इस array में 8 elements है तथा इस array को दो भागों में विभाजित किया जाता है जिसमें कि 4 – 4 elements होंगे.

फिर इन दो arrays को भी दो भागों में विभाजित किया जाता है.

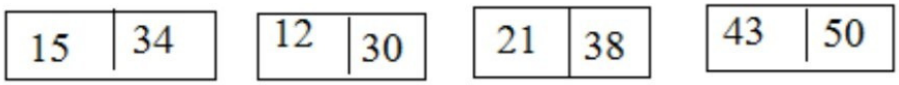


इसके बाद हम इन arrays को भी विभाजित कर देते है. इसके बाद लिस्ट में केवल एक elements बचेगा. जिसे विभाजित नहीं किया जा सकता है.



अब हम इन सब को उसी प्रकार combine करेंगे जिस प्रकार की ये विभाजित हुए थे.
सबसे पहले हम elements को प्रत्येक लिस्ट के लिए compare करते है तथा उसके बाद इन्हें sort करके combine कर दिया जाता है.

सबसे पहले 15 और 34 को compare करते है परन्तु ये तो पहले से ही sorted है. 30 और 12 को compare किया जाता है क्योंकि 30, 12 से बड़ा है इसलिए 12 को 30 से पहले लिखेंगे. 38 और 21 को compare करेंगे इसमें 21 को 38 से पहले लिखेंगे. 43 और 50 पहले से sorted है.



अब हम दो elements वाली लिस्ट को compare करेंगे.

इसके बाद अंत में इन दो लिस्ट को compare किया जाता है:-

12	15	21	30	34	38	43	50

merge sort algorithm in hindi:-

इस sort की algorithm निम्नलिखित है:-

MERGE-SORT (A, p, r)

1:- IF p<r
2:- THEN q = FLOOR [(p+q)/2] ///divide step
3:- MERGE (A, p, q) ///conquer step
4:- MERGE (A, q+1, r) ///conquer step
5:- MERGE (A, p, q, r) ///conquer step

इसे भी पढ़े:-bubble sort in hindi

sorting in hindi & types of sorting in hindi

Sorting in hindi:-

डेटा स्ट्रक्चर में sorting वह प्रक्रिया है जिसके द्वारा हम डेटा को एक logical order में arrange (क्रमबद्ध) करते है. यह लॉजिकल ऑर्डर ascending ऑर्डर भी हो सकता है या descending ऑर्डर भी हो सकता है.
ascending का अर्थ होता है कि बढ़ते क्रम में और descending का अर्थ होता है घटते क्रम में.

Sorting का संबंध ढूंढने (searching) से है, हमारी अपनी जिंदगी में बहुत सी चीजें होती है जिन्हें हम ढूंढते है **जैसे:-** google में कोई टॉपिक, किताब में कोई पेज, डिक्शनरी में कोई शब्द, किसी परीक्षा में रोल नंबर तथा हमारे मोबाइल के contacts no. आदि.
तो ये सभी चीजें जो होती है वह sorted (arrange) होती है जिससे हम आसानी से उन्हें ढूंढ लेते है.

types of sorting (सॉर्टिंग के प्रकार):-

यह दो प्रकार की होती है:-
1:- internal सॉर्टिंग
2:- external सॉर्टिंग

1:- internal sorting:- इस सॉर्टिंग में sort किये जाने वाला सभी डेटा main memory में ही रहता है.
internal sorting के प्रकार निम्नलिखित है:-

1:- bubble sort
2:- insertion sort
3:-quick sort
4:- heap sort
5:- selection sort

2:- external sorting:- इस सॉर्टिंग में sort किये जाने वाला डेटा secondary memory में रहता है. क्यूंकि डेटा इतना ज्यादा होता है कि वह main memory में नहीं आ पाता. external सॉर्टिंग का एक ही प्रकार होता है वह है merge sort

Bubble sort:-

bubble sort एक बहुत ही आसान sorting तकनीक है. इसमें शुरुवात की दो elements को compare किया जाता है. यदि left वाला एलिमेंट right वाले एलिमेंट से बड़ा है तो वे अपने स्थान को एक दूसरे से बदल लेंगे. और comparison अंत तक चलता रहेगा.

bubble sort algorithm in hindi:-

bubble sort को implement करने के लिए निम्नलिखित algorithm का प्रयोग किया जाता है.

(bubble sort) BUBBLE (a,n)
यहाँ a एक array है जिसमें n elements है.

step 1:- repeat step 2 & 3 for k = 1 to n-1 (no. of passes)

step 2:- repeat step 3 for p = 1 to n-k (no. of comparisions)

step 3:- check if (a[p] a>[a+1]) then
interchange a[p] to a[p-1]
end of if structure
end of step 2 loop
end of step 3 loop

step 4:- exit

functions in "c programming language" hindi

Functions in c:-

Function एक piece of code होता है दुसरे शब्दों में कहें तो यह प्रोग्राम में एक प्रकार से एक sub-program की भाँती कार्य करता है.

किसी प्रोग्राम को बनाते समय कभी-कभी हमें आवश्यकता होती है कि हमें कुछ कोड्स के execution से प्राप्त result को प्रोग्राम में बार बार प्रयोग करना पड़ता है, ऐसी स्थिति में कोड्स को बार-बार नहीं लिखा जाता है बल्कि फंक्शन के रूप में main() फंक्शन के बाहर परिभाषित करके एक ही स्थान पर प्रयोग कर लिया जाता है. और फंक्शन से प्राप्त result को प्रोग्राम में प्रयोग करने के लिए उस फंक्शन को call कर लिया जाता है. functions का फायदा यह है कि इससे हमारा समय तथा जगह दोनों कि बचत होती है.

फंक्शन का syntax:-

```
function name(arg1, arg2, arg3….)
{
statement1;
statement2;
statement3;
…….
}
```

programming language “c” में फंक्शन दो प्रकार के होते है.
1:- Built-in-functions (बिल्ट-इन-फंक्शन)
2:- user defined functions (यूजर डिफाइंड फंक्शन)

1:- Built-in-function in hindi:-

बिल्ट-इन-फंक्शन वे फंक्शन होते है जिनके prototype प्रोग्रामिंग भाषा “सी” की header file में सुरक्षित रहते हैं. इन फंक्शन को प्रोग्राम में बस इनका नाम लिखकर इन्हें कॉल (call) किया जाता है और ये प्रोग्राम में क्रियान्वित हो जाता है. इसके उदाहरण:- scanf();, printf();, strcat(); आदि.

ये function “सी” प्रोग्रामिंग भाषा के library functions भी कहलाते है. इन सभी फंक्शन का सम्बन्ध किसी विशिष्ट “सी” लाइब्रेरी फाइल से होता है. ये विशेष लाइब्रेरी फाइल्स header files कहलाती है एवं इनका extension नाम .h होता है. अतः “सी” की लाइब्रेरी में स्थित वे सभी फाइल्स जिनका विस्तारित नाम .h होता है, हैडर फाइल्स कहलाती है.

“सी” में अनेक हैडर फाइल्स इनमे से कुछ निम्न है:-
Stdio.h
Math.h
String.h
Conio.h
time.h
ctype.h

2:- user defined function in hindi (यूजर डिफाइंड फंक्शन):-

यूजर डिफाइंड फंक्शन वे होते है जो कि यूजर के द्वारा प्रोग्राम को लिखते समय बनाये जाते है यानी कि डिफाइन किये जाते है. यूजर को जिस प्रकार की जरुरत होती है वह अपनी आवश्यकता के अनुसार फंक्शन को create कर सकता है.

प्रोग्रामिंग लैंग्वेज "सी" में प्रोग्राम लिखने के लिए एक main() फंक्शन की आवश्यकता होती है यह फंक्शन भी यूजर डिफाइंड फंक्शन है. प्रोग्राम का execution भी इसी प्रोग्राम से शुरू होता है.

फंक्शन का नाम "सी" में प्रयोग किये जाने वाले keywords के अतिरिक्त कुछ भी रखा जा सकता है.
फंक्शन के बाद () braces लगाना आवश्यक होता है, यह फंक्शन का सूचक होता है.
किसी फंक्शन को किसी अन्य फंक्शन तथा अपने आप में call किया जा सकता है.

निवेदन:- आपको यह पोस्ट कैसी लगी हमें comment के माध्यम से बताइये तथा इसे अपने दोस्तों के साथ share करें. धन्यवाद.

Data Structure Operations in hindi

Data structure operations in hindi:-

डेटा स्ट्रक्चर में डेटा को process करने के लिए विभिन्न operations का प्रयोग किया जाता है जो निम्नलिखित है:-

1:-Traversing:-डेटा स्ट्रक्चर के प्रत्येक element को केवल एक बार visit करना traversing कहलाता है।

2:-Searching:-डेटा स्ट्रक्चर में किसी element को खोजना जो कि एक या एक से अधिक condition को संतुष्ट करता हो।

3:-Inserting:-डेटा स्ट्रक्चर में समान प्रकार के element को जोड़ना(insert) insertion कहलाता है। डेटा स्ट्रक्चर में element को कही भी add किया जा सकता है।

4:-Deleting:-डेटा स्ट्रक्चर में से element को remove करना Deletion कहलाता है। डेटा स्ट्रक्चर में element को कही से भी remove किया जा सकता है।

5:-Sorting:-डेटा स्ट्रक्चर में elements को ascending तथा descending क्रम में arrange(क्रमबद्ध) करना Sorting कहलाता है।

6:-Merging:-दो भिन्न-भिन्न डेटा files में स्थित elements को एक डेटा file में combine कर स्टोर करना Merging कहलाता है।

Memory Allocation in hindi

Memory Allocation in hindi:-

मेमोरी एलोकेशन एक ऐसी प्रक्रिया जिसमें कंप्यूटर प्रोग्राम्स को मेमोरी allocate की जाती है।

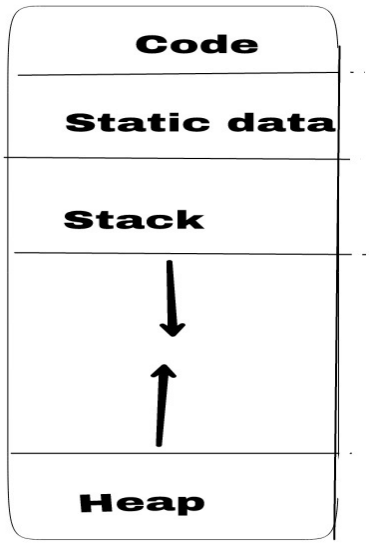
मेमोरी एलोकेशन दो प्रकार की होती है:-
1:-Static Memory Allocation
2:-Dynamic Memory Allocation

1:-Static Memory Allocation:-static memory allocation में मेमोरी को compile time में ही allocate कर दिया जाता है।

इस allocation का प्रयोग तब किया जाता है जब मेमोरी की साइज़ नियत हो।

इसमें हम execution के दौरान मेमोरी को allocate और deallocate नही कर सकते है तथा जो variables होते है वह हमेशा के लिए allocate हो जाते है।

stacks तथा heaps के द्वारा इस allocation को implement किया जाता है।



2:-Dynamic Memory Allocation:-वह प्रक्रिया जिसमें मैमोरी runtime में allocate की जाती है Dynamic memory allocation कहलाती है।

Data segments के द्वारा इस allocation को implement किया जाता है।

Advantage and disadvantage of array in hindi

advantage of array in hindi:-

array के निम्नलिखित advantage होते है:-

1:-Array को आसानी से implement किया जा सकता है।
2:-एक ही प्रकार के विभिन्न डेटा items को केवल एक नाम के द्वारा प्रदर्शित किया जा सकता है।
3:-array एक ही समय में अनेक डेटा items को स्टोर कर सकता है।
4:-2D arrays का प्रयोग matrices को प्रदर्शित करने के लिए किया जाता है।

Disadvantage of arrays in hindi:-

arrays के निम्नलिखित advantage होते है:-

1:-Array के द्वारा मैमोरी का waste होता है।
2:-Array एक static डेटा स्ट्रक्चर है जिसके कारण इसका size पहले से ही define होता है।
3:-हमें array में एक element को delete तथा insert करने के लिए पूरे array को traverse करना पड़ता है।

What is array in hindi & types of arrays in hindi?

Array in hindi:-

array एक non-primitive तथा linear डेटा स्ट्रक्चर है जो कि एकसमान(similar) डेटा items का समूह होता है, अर्थात यह सिर्फ एक ही प्रकार के डेटा को ही स्टोर करेगा (या तो यह सिर्फ सभी integer डेटा को स्टोर करेगा या फिर सभी floating point को)।

Array डेटा स्ट्रक्चर का प्रयोग डेटा ऑब्जेक्ट्स के समूह को संग्रहित करने के लिये किया जाता है।

“Arrays एक static डेटा स्ट्रक्चर है अर्थात् हम केवल compile time में ही मेमोरी को allocate कर सकते है और इसे run-time में बदल नही सकते है।”

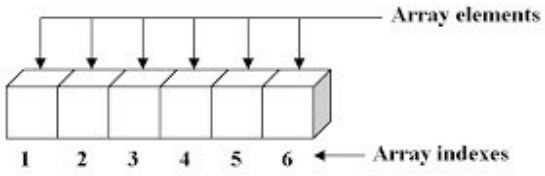
Types of array in hindi:-

Arrays निम्नलिखित तीन प्रकार का होता है:-

1:- one dimensional arrays.
2:- two dimensional arrays.
3:- Multi dimensional arrays.

1:- one dimensional(1-D) arrays:-

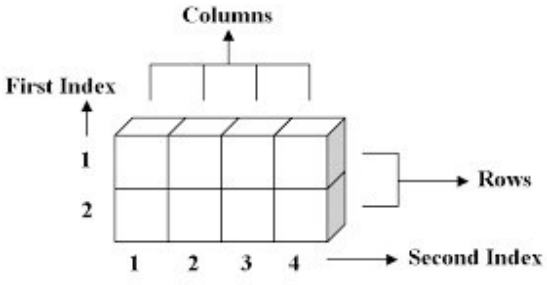
वह arrays जिसमे सिर्फ एक subscript होती है उसे one dimensional arrays कहते है। इसका प्रयोग linear रूप में डेटा को स्टोर करने के लिए किया जाता है।



One-dimensional array with six elements

2:- two dimensional(2-D) arrays:-

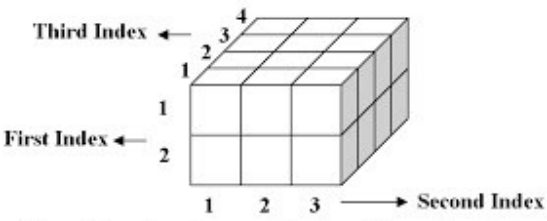
वह arrays जिसमें दो subscript होती है उसे two dimensional array कहते है। two dimensional arrays को matrix तथा table भी कहते है।



Two-dimensional array with eight elements

3:- Multi dimensional(M-D) arrays:–

वह arrays जिसमें दो से ज्यादा subscript होती है वह Muti-dimensional arrays कहलाता है।



Three-dimensional array with twenty four elements

Algorithm for ENQUEUE and DEQUEUE in hindi

Algorithm of Enqueue in hindi:-

Queue में किसी item या element को add करने के लिए निम्नलिखित एल्गोरिथ्म का प्रयोग किया जाता है:-

```
1. If ( REAR = size ) then //Queue is full
2. print "Queue is full"
3. Exit
4. Else
5. If ( FRONT = 0 ) and ( REAR = 0 )
then //Queue is empty
6. FRONT = 1
7. End if
8. REAR = REAR + 1 // increment REAR
9. Que[ REAR ] = ITEM
10. End if
11. Stop
```

Fig:-Enqueue algorithm

Algorithm of Dequeue in hindi:-

Queue में किसी item को remove करने के लिए निम्नलिखित एल्गोरिथ्म का प्रयोग किया जाता है:-

```
1. If ( FRONT = 0 ) then
2. print "Queue is empty"
3. Exit
4. Else
5. ITEM = Que [ FRONT ]
6. If ( FRONT = REAR )
7. REAR = 0
8. FRONT = 0
9. Else
10. FRONT = FRONT + 1
11. End if
12. End if
13. Stop
```

data structure multiple choice questions MCQ in hindi

Data structure MCQ in hindi:-

मैंने यहाँ पर कुछ महत्वपूर्णdata structure mcq दिए है जो कि प्रतियोगी परीक्षाओं में बहुत उपयोगी साबित हो सकते हैं तो चलिए पढ़ते है.

1:- निम्नलिखित में से कौन सा non – linear डेटा स्ट्रक्चर है?

1. string
2. list
3. tree
4. stack

उत्तर:- tree

2:- निम्नलिखित में से कौन सा internal sort नहीं है?

1. heap सॉर्ट
2. इंसर्शन सॉर्ट
3. quick सॉर्ट
4. merge सॉर्ट

उत्तर:- merge सॉर्ट

3:- two dimensional array (द्विविमीय ऐरे) को और क्या कहते है?

1. मैट्रिक्स ऐरे
2. टेबल ऐरे
3. उपर के दोनों
4. इनमें से कोई नहीं

उत्तर:- उपर के दोनों

4:- स्टैक से सम्बन्धित इन में से कौन है?

1. push
2. pop
3. FIFO
4. ये सभी

उत्तर:- ये सभी

5:- लिंक्ड लिस्ट किस प्रकार का डेटा स्ट्रक्चर है?

1. non – linear
2. linear
3. hierarchical
4. इनमें से कोई नहीं

उत्तर:- linear

6:- अगर लिस्ट में कोई item (नोड) नहीं है तो उसे क्या कहते है?

1. null लिस्ट
2. empty लिस्ट
3. जीरो लिस्ट
4. इनमें से कोई नहीं

उत्तर:- null लिस्ट

7:- quick sort की worst case complexity कितनी है?

1. O (n log n)
2. O (n^2)
3. O (log n)
4. O (n)

उत्तर:- O(n^2)

8:- निम्न prefix एक्सप्रेशन का post fix form क्या है?

-M/N*P$QR

1. MNPQR$*/-
2. M-NPQR$*/
3. MNP$QR/-
4. M-NQR$*/

उत्तर:- MNPQR$*/-

9:- एक full बाइनरी ट्री जिसमें n leaves है में नोड्स होंगे?

1. 2n-1 नोड्स
2. $2n^2$ नोड्स
3. n नोड्स
4. log n नोड्स

उत्तर:- 2n-1 नोड्स

10:- M*NP+/Q का post fix फॉर्म होगा.

1. MN*PQ/+
2. M*NP+P/Q
3. *MN/PQ+

4. MNPQ+/*

उत्तर:- MN*PQ/+

11 से 20 data structure mcq

11:- RECURSION को implement करने के लिए इनमें से किसका प्रयोग किया जाता है?

1. stack
2. graph
3. queue
4. array

उत्तर:- stack

12:- Queue में नए नोड कहाँ से जोड़े जाते है?

1. आगे से
2. पीछे से
3. मध्य से
4. आगे पीछे दोनों से

उत्तर:- पीछे से

13:- direct serch के लिए तकनीक है.

1. linear सर्च
2. tree सर्च
3. binary सर्च
4. hashing

उत्तर:- hashing

14:- merge sort की worst case complexity है.

1. O (n log n)
2. O (n^2)
3. O (log n)
4. O (n)

उत्तर:- O (n log n)

15:- निम्नलिखित में से सबसे धीमी सॉर्टिंग अल्गोरिथम है.

1. selection सॉर्ट

2. bubble सॉर्ट
3. quick सॉर्ट
4. heap सॉर्ट

उत्तर:- bubble सॉर्ट

16:- bubble सॉर्ट अल्गोरिथम की case complexity है.

1. O (log n)
2. O (n log n)
3. O (n)
4. O (n^2)

उत्तर:- O(n^2)

17:- इनमें से किस डेटा स्ट्रक्चर में infix नोटेशन को post fix नोटेशन में बदलने की जरुरत होती है.

1. ट्री
2. queue
3. स्टैक
4. ऐरे

उत्तर:- स्टैक

18:- स्टैक में डेटा को जोड़ने को कहते है.

1. POP
2. add
3. push
4. इनमे से कोई नहीं

उत्तर:- push

19:- निम्न डेटा स्ट्रक्चर में से किसमें elements को delete कर सकते है?

1. stack
2. queue
3. dequeue
4. tree

उत्तर:- dequeue

20:- निम्नलिखित में से कौन सा डेटा स्ट्रक्चर homogeneous डेटा आइटम्स स्टोर करता है.

1. pointer

2. array
3. record
4. इनमें 0से कोई नहीं

उत्तर:- record

21 से 27 data structure mcq

21:- वह कौन सी स्थिति है जब हम डेटा स्ट्रक्चर में item डालना चाहते है परन्तु इसमें कोई जगह नहीं होती.

1. overflow
2. underflow
3. housefull
4. saturated

उत्तर:- overflow

22:- insertion सॉर्ट की औसत case complexity है.

1. O (n)
2. O (log n)
3. O (n log n)
4. O (n^2)

उत्तर:- O (n^2)

23:- डायनामिक मैमोरी क्षेत्र है.

1. स्टैक
2. हीप
3. हार्ड डिस्क
4. इनमें से कोई नहीं

उत्तर:- हीप

24:- यदि font = rear तब queue है.

1. भरा हुआ
2. आधा भरा हुआ
3. खाली
4. इनमें से कोई नहीं

उत्तर:- खाली

25:- suffix एक्सप्रेशन है.

1. infix
2. postfix
3. prefix
4. ये सभी

उत्तर:- postfix

26:- polish एक्सप्रेशन है.

1. infix
2. postfix
3. prefix
4. इनमें से कोई नहीं

उत्तर:- infix

27:- doubly लिंक्ड लिस्ट कितने पॉइंटर्स प्रयोग करता है.

1. तीन
2. चार
3. एक
4. दो

उत्तर:- दो.