

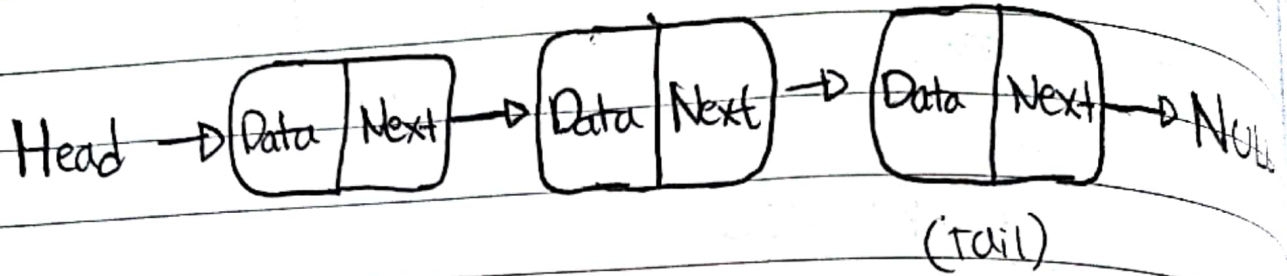
# Linked List

Page :

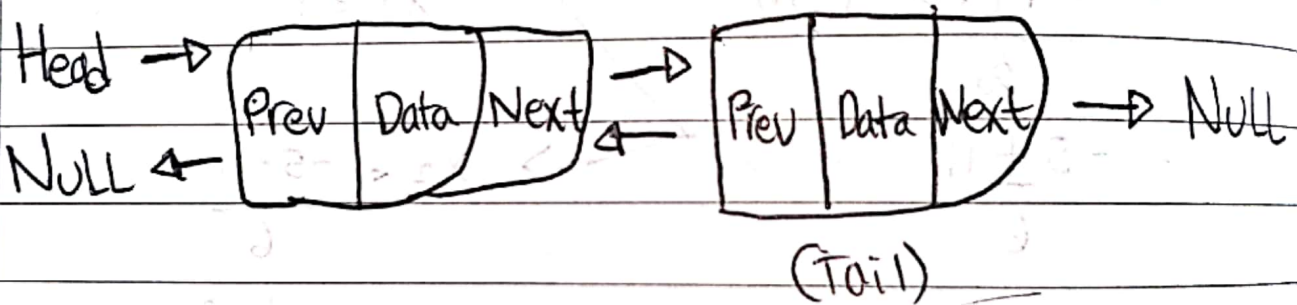
Date :

No.

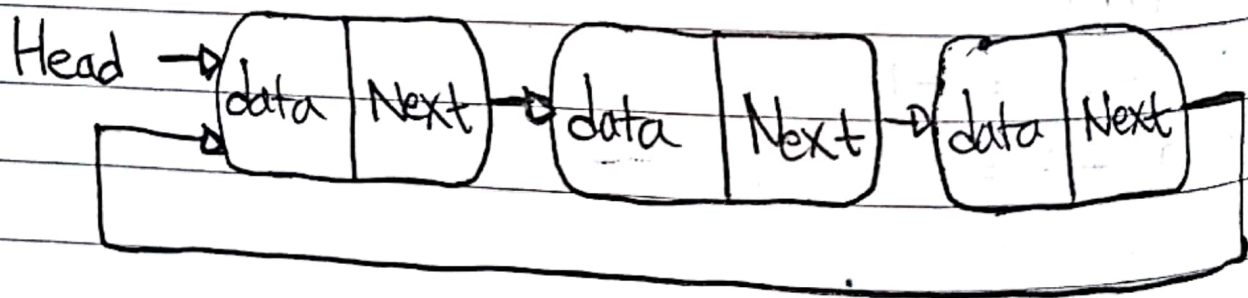
1.) Single linked list




double linked list



Circular linked list



No.

2.) Array dan linked list merupakan data structure. Array merupakan Primitive data structure, sedangkan linked list merupakan non-Primitive data structure, dan memiliki kumpulan elemen yang tidak berurutan. Mengakses data lebih cepat jika menggunakan array dibandingkan dengan linked list. Misalkan kita ingin mengakses elemen ke empat, jika kita menggunakan array, kita hanya perlu ~~menulis~~ menulis index dari array yang ingin diakses, tetapi jika menggunakan linked list, kita harus memulai dari head  hingga elemen ke 4. Operasi seperti insertion dan deletion di dalam array membutuhkan waktu yang lama, tetapi di dalam linked list sangat cepat. Array memiliki size yang fixed size artinya sizenya tidak dapat berubah dengan sendirinya, sebaliknya linked list memiliki size yang dynamic dan fleksibel dan dapat memperbesar dengan sendirinya jika dibutuhkan.



2.)

Prefix notation = Prefix notation adalah cara Penulisan operator di depan operand, cth : + AB, - + ABC

Infix notation = Infix notation adalah cara Penulisan Operator di tengah 2x dari 2 operand.

Postfix notation = Cara Penulisan operator dibelakang operand.

\* Prefix dan Post fix tidak memerlukan tanda kurung.

## Hashing and Hash Tables

1.)

hash table adalah Sebuah data Structure Yang menyimpan Sebuah data menggunakan key Value yang didapat dari nilai data itu sendiri.

hash function adalah satu function yang fungsinya berguna untuk mendapatkan hash value dari key value suatu data.

Collision adalah ada ~~lebih~~ lebih dari satu data yang memiliki hash index yang sama, padahal seharusnya satu index hanya dapat menyimpan satu data saja.

# Linear Probing :

Misalkan kita mempunyai hash Function "key mod 5"

dan key nya = 51, 27, 700, 32, 50

0		0		0	
1		1	51	1	51
2		2		2	27
3		3		3	
4		4		4	

table kosong      masukan 51      masukan 27

0	700	0	700	0	700	0	700
1	51	1	51	1	51	1	51
2	27	2	27	2	27	2	27
3		3		3	32	3	32
4		4		4		4	

masukan 700      masukan 32 (terjadi collision) maka kita akan masukan ke free slot selanjutnya yaitu ③

masukan 50 (terjadi collision) maka kita masukan ke free slot selanjutnya.

0	700
1	51
2	27
3	32
4	50

→ Maka akan mendapatkan Hash table seperti ini.

No.

Chaining :

Misalkan kita mempunyai hash Function "key mod 5"

dan keynya = 51, 27, 700, 32, 50

0		0		0		0	700
1		1	51	1	51	1	51
2		2		2	27	2	27
3		3		3		3	
4		4		4		4	

Initial table

masukan  
51masukan  
27masukan  
700

0	700	0	700	0	700	→	50
1	51	1	51	1	51		
2	27	2	27	→	32	→	32
3		3					
4		4					

masukan 32

Masukan 50

(terjadi collision)

maka kita akan membuat Pointer yang menunjuk

kepada linked list yang dapat diisi dengan

• keys yang ~~hasil~~ hasil dari hash Functionnya sama.

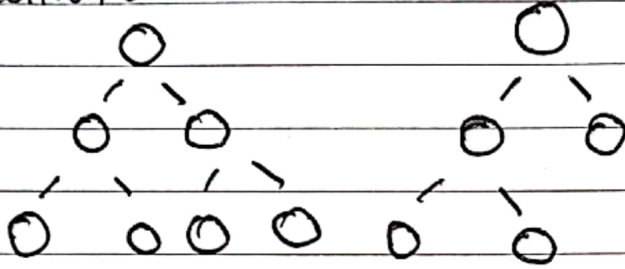


# Binary Search tree

## ① ~~Full Binary Tree~~ Full Binary Tree

~~Full~~ Full Binary tree adalah Binary tree yang memiliki node nya 0 atau 2.

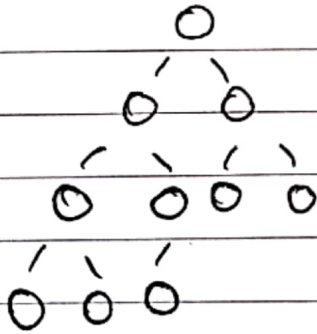
Contoh :



## Complete Binary tree

adalah Binary tree yang semua level terpenihi except level terakhir yang akersnya harus berada di sebelah kiri mungkin.

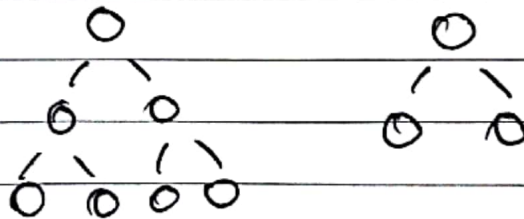
Contoh :



## Perfect Binary Tree

adalah Binary tree yang memiliki Internal nodes 2 dan harus mempunyai level yang sama.

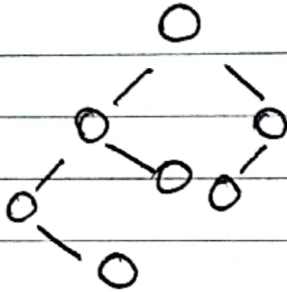
Contoh :



## Balance Binary Tree

adalah Binary tree yang memiliki tinggi  $O(\log N)$  dimana  $N$  adalah banyaknya Node. • contoh: AVL Tree, Red-Black tree.

Contoh:



## Degenerate Binary Tree

adalah Binary tree yang hanya memiliki 1 anak.

Contoh:

