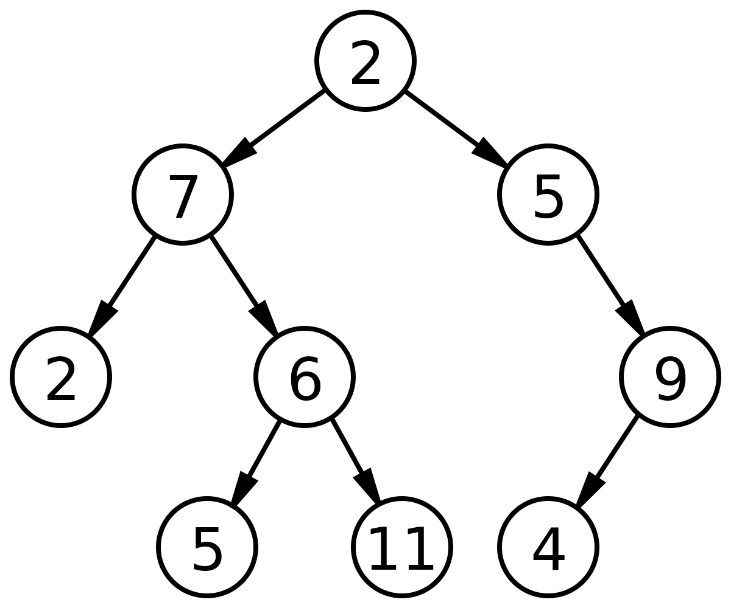
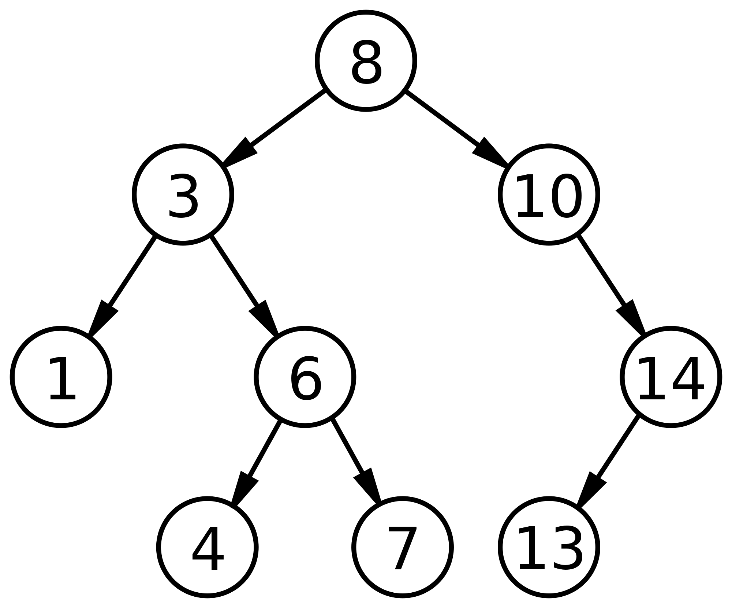
**Binary Tree**

[](https://upload.wikimedia.org/wikipedia/commons/thumb/f/f7/Binary_tree.svg/1200px-Binary_tree.svg.png)

Binary Tree atau Pohon Biner adalah sebuah pohon dalam struktur data yang bersifat hirarkis (hubungan one to many). Tree bisa didefenisikan sebagai kumpulan simpul dengan setiap simpul mempunyai paling banyak dua anak. Secara khusus, anaknya dinamakan kiri dan kanan. Binary tree tidak memiliki lebih dari tiga level dari Root.  
Binary tree adalah suatu tree dengan syarat bahawa tiap node (simpul) hanya boleh memiliki maksimal dua subtree dan kedua subtree tersebut harus terpisah. Tiap node dalam binary treee boleh memiliki paling banyak dua child (anak simpul), secara khusus anaknya  dinamakan kiri dan kanan.  
Pohon biner dapat juga disimpan sebagai struktur data implisit dalam array, dan jika pohon tersebut merupakan sebuah pohon biner lengkap, metode ini tidak boros tempat. Dalam penyusunan yang rapat ini, jika sebuah simpul memiliki indeks i, anaknya dapat ditemukan pada indeks ke-2i+1 dan 2i+2, meskipun ayahnya (jika ada) ditemukan pada indeks lantai ((i-1)/2) (asumsikan akarnya memiliki indeks kosong).

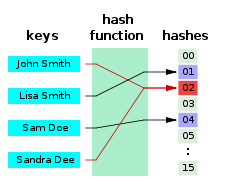
**Binary Search Tree**

[](https://upload.wikimedia.org/wikipedia/commons/thumb/d/da/Binary_search_tree.svg/2000px-Binary_search_tree.svg.png)

Binary Search Tree adalah tree yang terurut (ordered Binary Tree). Binary Search Tree juga sering disebut dengan Sorted Binary Tree yang berfungsi untuk menyimpan informasi nama atau bilangan yang disimpan di dalam memory. Dengan ini data dibagi menjadi dua dengan mencari titik tengah seagai patokannya. Binary tree terdiri dari simpul utama yang disebut dengan istilah root. Kemudian dari root tersebut terdapat bagian kiri dan bagian kanan. Data disimpan setelah root disimpan berdasarkan nilai perbandingan dengan root tersebut. Pengurutan dapat dilakukan bila BST ditelusuri (traversed) menggunakan metode in-order. Detail dari proses penelusuran ini akan dibahas pada pertemuan selanjutnya. Data yang telah tersusun dalam struktur data BST juga dapat dicari dengan mudah dan memiliki rata-rata kompleksitas sebesar O(log n), namun membutuhkan waktu sebesar O(n) pada kondisi terjelek dimana BST tidak berimbang dan membentuk seperti linked list  
Binary search tree memungkinkan pencarian dengan cepat, penambahan, juga menghapus data yang ada di dalamnya, bisa juga digunakan sebagai implementasi sejumlah data dinamis, atau pencarian table data dengan menggunakan informasi kunci atau key.  
Agar data benar-benar tersusun dalam struktur data BST, dua aturan yang harus dipenuhi pada saat data diatur dalam BST adalah sebagai berikut:

* Semua data dibagian kiri sub-tree dari node t selalu lebih kecil dari data dalam node t itu sendiri.
* Semua data dibagian kanan sub-tree dari node t selalu lebih besar atausama dengan data dalam node t.

**Hash**

[](https://upload.wikimedia.org/wikipedia/commons/thumb/5/58/Hash_table_4_1_1_0_0_1_0_LL.svg/240px-Hash_table_4_1_1_0_0_1_0_LL.svg.png)

Hash atau Hashing berarti memenggal dan kemudian menggabungkan. Hash menggunakan memori penyimpanan utama berbentuk array dengan tambahan algoritma untuk mempercepat pemrosesan data. Hash merupakan suatu metode yang secara langsung mengakses record-record dalam suatu tabel dengan melakukan transformasi aritmatik pada key yang menjadi alamat dalam tabel tersebut.  
Hashing digunakan sebagai metode untuk menyimpan data dalam sebuah array agar penyimpanan data, pencarian data, penambahan  data dan penghapusan data dapat dilakukan dengan cepat.  
Fungsi hash haruslah stabil (referential transparent), artinya, jika ia dipanggil dua kali oleh masukan yang benar-benar sama(sebagai misal,string yang mengandung sekuen karakter yang sama), maka ia haruslah memberi hasil yang sama pula.  
Pelacakan dengan menggunakan Hash terdiri dari dua langkah utama, yaitu:

**Menghitung Fungsi Hash**

Fungsi Hash adalah suatu fungsi yang mengubah key menjadi alamat dalam tabel. Fungsi Hash memetakan sebuah key ke suatu alamat dalam tabel. Idealnya, key-key yang berbeda seharusnya dipetakan ke alamat-alamat yang berbeda juga. Pada kenyataannya, tidak ada fungsi Hash yang sempurna. Kemungkinan besar yang terjadi adalah dua atau lebih key yang berbeda dipetakan ke alamat yang sama dalam tabel. Peristiwa ini disebut dengan collision (tabrakan). Karena itulah diperlukan langkah berikutnya, yaitu collision resolution (pemecahan tabrakan).

**Collision Resolution**

Collision resolution merupakan proses untuk menangani kejadian dua atau lebih key di-hash ke alamat yang sama. Cara yang dilakukan jika terjadi collision adalah mencari lokasi yang kosong dalam tabel Hash secara terurut. Cara lainnya adalah dengan menggunakan fungsi Hash yang lain untuk mencari lokasi kosong tersebut.