Splay Tree

Adalah salah satu self-balancing BSTs. Untuk menjaga agar tree tetap balance, splay tree mengimplementasikan rotation di dalamnya. Spaly tree tidak memperhatikan height dari sebuah subtree melainkan melempar setiap key yang diinsert ke base root.

Time complexity: O (log(n))

Untuk search, setiap key yang berhasil ditemukan akan dilempar ke base root. Dari search ini, kita dapat mencari key lebih mudah, terutama key yang sering diakses. Karena key yang sering diakses akan berada di bagian atas tree, jadi tidak memerlukan proses search yang terlalu lama. Sebaliknya, key yang jarang diakses akan membutuhkan proses search yang lebih lama karena key tersebut berada di bawah tree.

Jika Splay Tree dibandingkan dengan AVL Tree dan RBT, maka dapat disimpulkan:

- Splay Tree lebih simple karena hanya mengimplementasikan rotation. Di RBT, diperlukan recolor, rotation, double black.
- Splay Tree melakukan rotation baik di search, insert maupun rotation. AVL dan RBT tidak melakukan rotation pada saat search, sedangkan Splay Tree melakukannya.
- Time Complexity dari Splay Tree, AVL Tree, dan RBT sama yaitu O (log(n)) baik di average maupun worst. Untuk Time Complexity access Splay Tree tergantung kondisi pada saat insert. Mungkin saja Splay Tree akan mengakses key lebih cepat jika proses insert dilakukan secara berurutan.

Jadi, Splay Tree akan sangat berguna jika program memperhatikan search yang cepat pada key yang sering diakses. Tapi, untuk insert dan deletion, lebih unggul AVL dan RBT karena tidak perlu lagi melempar key yang dinsert ke base root dan cukup memperhatikan height (AVL) atau warna (RBT).