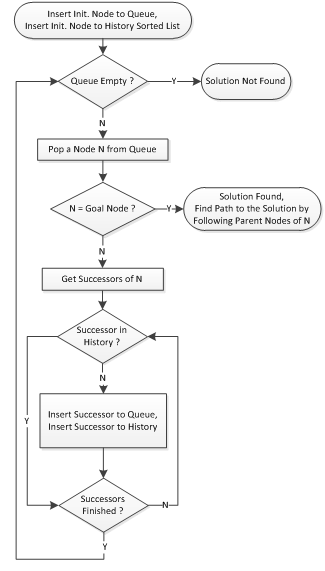
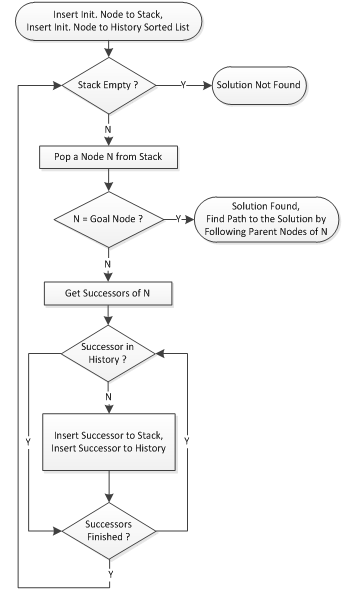
# Implementasi BFS



***Implemented BFS algorithm***

# Implementasi DFS

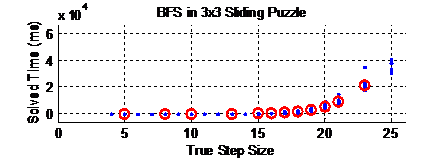


***Implemented DFS algorithm***

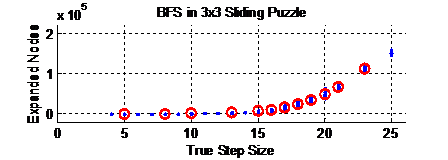
# Analisis Algoritma

Untuk perbandingan algoritma, 100 buah 3x3 puzzle dibuat dengan harapan memiliki *true steps* 5, 10, 15, 20 dan 25. Total 500 puzzle diselesaikan dengan menggunakan 2 algoritma yaitu BFS dan DFS. Hasil datanya bervariasi mulai dari 4 sampai 25 langkah. Semua data dimuat dalam grafik dengan titik biru, sedangkan rata-ratanya dengan menggunakan titik merah dalam grafik.

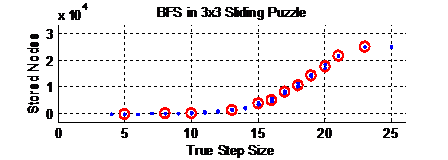
Algoritma BFS memberikan hasil eksponensial untuk semua data. Karena pencarian dilakukan hampir menggunakan seluruh *space*.



***BFS time performance at 3x3 puzzle size***

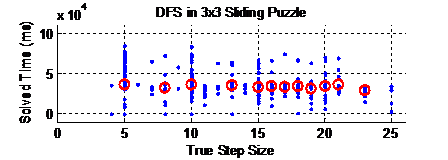


***BFS node expansion at 3x3 puzzle size***

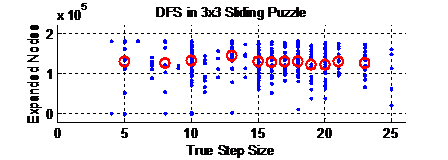


***BFS node storage at 3x3 puzzle size***

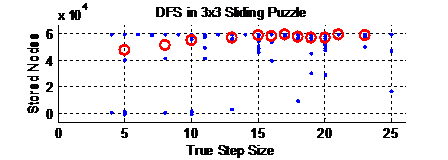
Untuk DFS, perhatikan bahwa banyaknya langkah tidak perlu diperhatikan. Pencarian DFS dilakukan dengan mencari seluruh *space* atau dapat mencari hasil secara cepat. Tetapi ketika data yang digunakan rata-rata, maka pencarian dengan melakukan DFS hasilnya hampir sama. Hasil DFS sudah seperti yang diharapkan. Gambar di bawah akan menunjukkan perbandingannya dimana DFS tampak lebih *inferior* daripada BFS. Itu disebabkan karena kita biasanya awalnya memasukkan data pada *very low step solution*. Tapi jika kita memberikan puzzle yang memiliki langkah yang cukup besar, BFS *solution time* akan tetap naik sedangkan DFS *average solution time* akan tetap menunjukkan setengah dari waktu yang dibutuhkan dari melakukan pencarian seluruh *space*. Untuk atribut lainnya akan seperti itu juga.



***DFS time performance at 3x3 puzzle size***



***DFS node expansion at 3x3 puzzle size***



***DFS node storage at 3x3 puzzle size***