# METODE PELACAKAN/PENCARIAN

**Pencarian** = suatu proses mencari solusi dari suatu permasalahan melalui sekumpulan kemungkinan ruang keadaan (state space).

**Ruang keadaan** = merupakan suatu ruang yang berisi semua keadaan yang mungkin.

Untuk mengukur perfomansi metode pencarian, terdapat empat kriteria yang dapat digunakan :

- Completeness : apakah metode tersebut **menjamin penemuan solusi** jika solusinya memang ada?
- Time complexity : berapa lama **waktu** yang diperlukan?
- Space complexity : berapa banyak **memori** yang diperlukan
- Optimality : apakah metode tersebut menjamin menemukan **solusi yang terbaik** jika terdapat beberapa solusi berbeda?

**Teknik pencarian :**

**A. Pencarian buta (*blind search*)** : tidak ada informasi awal yang digunakan dalam proses pencarian :

1. Pencarian melebar pertama (*Breadth – First Search*)
2. Pencarian mendalam pertama (*Depth – First Search*)

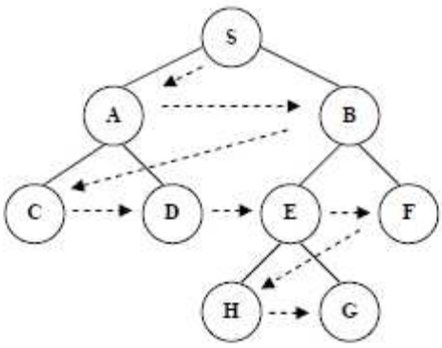
**B. Pencarian terbimbing (*heuristic search*)** : adanya informasi awal yang digunakan dalam proses pencarian

1. Pendakian Bukit (*Hill Climbing*)
2. Pencarian Terbaik Pertama (*Best First Search*)

## Pencarian Buta (blind search)

### 1. Breadth – First Search (BFS)

Semua node pada level n akan dikunjungi terlebih dahulu sebelum mengunjungi node-node pada level n+1. Pencarian dimulai dari node akar terus ke level 1 dari kiri ke kanan, kemudian berpindah ke level berikutnya dari kiri ke kanan hingga solusi ditemukan.



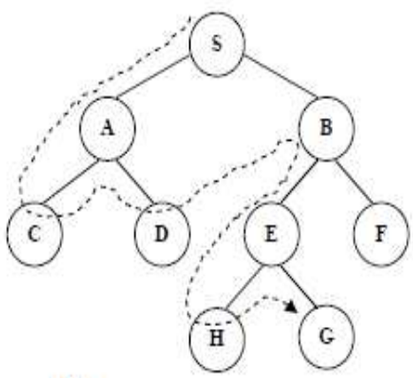

**Keuntungan :**

- tidak akan menemui jalan buntu, menjamin ditemukannya solusi (jika solusinya memang ada) dan solusi yang ditemukan pasti yang paling baik
- jika ada 1 solusi, maka breadth – first search akan menemukannya,jika ada lebih dari 1 solusi, maka solusi minimum akan ditemukan.
- Kesimpulan : **complete** dan **optimal**

**Kelemahan :**

- membutuhkan **memori yang banyak**, karena harus menyimpan semua simpul yang pernah dibangkitkan. Hal ini harus dilakukan agar BFS dapat melakukan penelusuran simpul-simpul sampai di level bawah
- membutuhkan **waktu yang cukup lama**

**2. Depth – First Search**




Pencarian dilakukan pada suatu simpul dalam setiap level dari yang paling kiri.

Jika pada level yang paling dalam tidak ditemukan solusi, maka pencarian dilanjutkan pada simpul sebelah kanan dan simpul yang kiri dapat dihapus dari memori.

Jika pada level yang paling dalam tidak ditemukan solusi, maka pencarian dilanjutkan pada level sebelumnya. Demikian seterusnya sampai ditemukan solusi.

**Keuntungan :**

- membutuhkan **memori relatif kecil**, karena hanya node-node pada lintasan yang aktif saja yang disimpan
- Secara kebetulan, akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan, jadi jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka DFS akan menemukannya dengan cepat → **waktu cepat**

**Kelemahan :**

- Memungkinkan tidak ditemukannya tujuan yang diharapkan, karena jika pohon yang dibangkitkan mempunyai level yang sangat dalam (tak terhingga) → **tidak complete** karena tidak ada jaminan menemukan solusi
- Hanya mendapat 1 solusi pada setiap pencarian, karena jika terdapat lebih dari satu solusi yang sama tetapi berada pada level yang berbeda, maka DFS tidak menjamin untuk menemukan solusi yang paling baik → **tidak optimal**.

# Heuristic Search

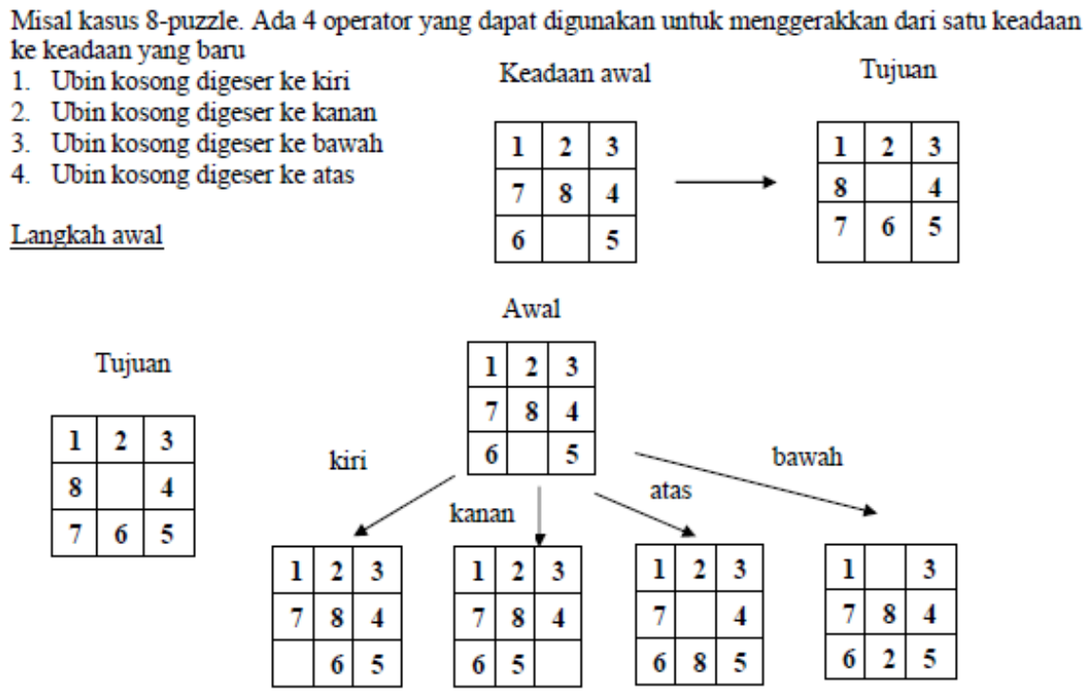
Pencarian buta tidak selalu dapat diterapkan dengan baik, hal ini disebabkan waktu aksesnya yang cukup lama & besarnya memori yang diperlukan. Untuk masalah dengan ruang masalah yang besar, teknik pencarian buta bukan metode yang baik karena keterbatasan kecepatan komputer dan memori.

- Metode heuristic search diharapkan bisa menyelesaikan permasalahan yang lebih besar.
- Metode heuristic search menggunakan suatu fungsi yang menghitung biaya perkiraan (estimasi) dari suatu simpul tertentu menuju ke simpul tujuan disebut **fungsi heuristic**
- Aplikasi yang menggunakan fungsi heuristic : Google, Deep Blue Chess Machine

Misal kasus 8-puzzle. Ada 4 operator yang dapat digunakan untuk menggerakkan dari satu keadaan ke keadaan yang baru

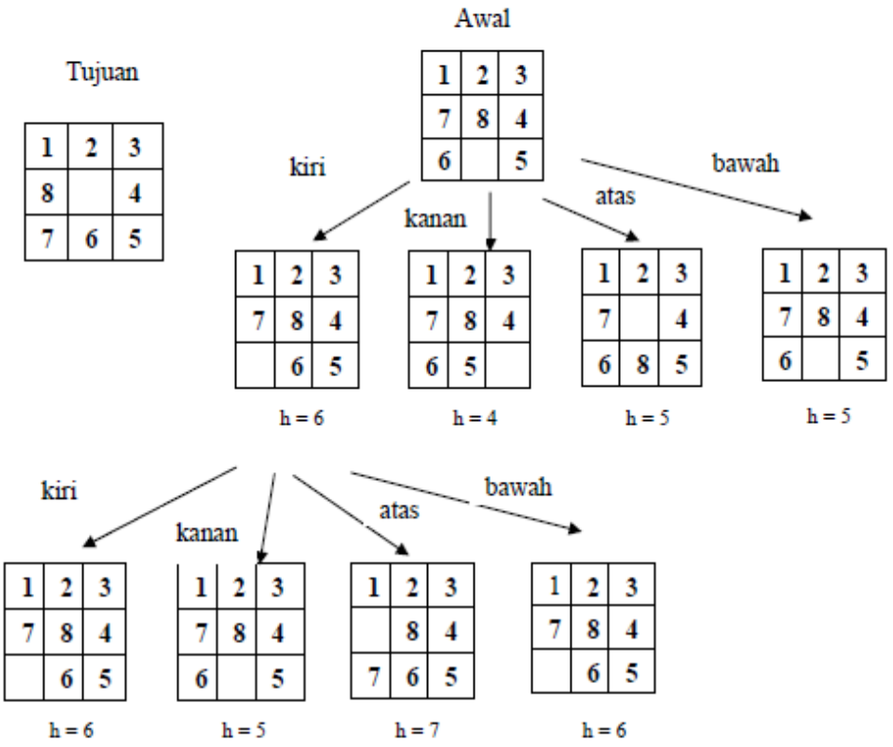
Tujuan Keadaan awal

1. Ubin kosong digeser ke kiri
2. Ubin kosong digeser ke kanan
3. Ubin kosong digeser ke bawah
4. Ubin kosong digeser ke atas

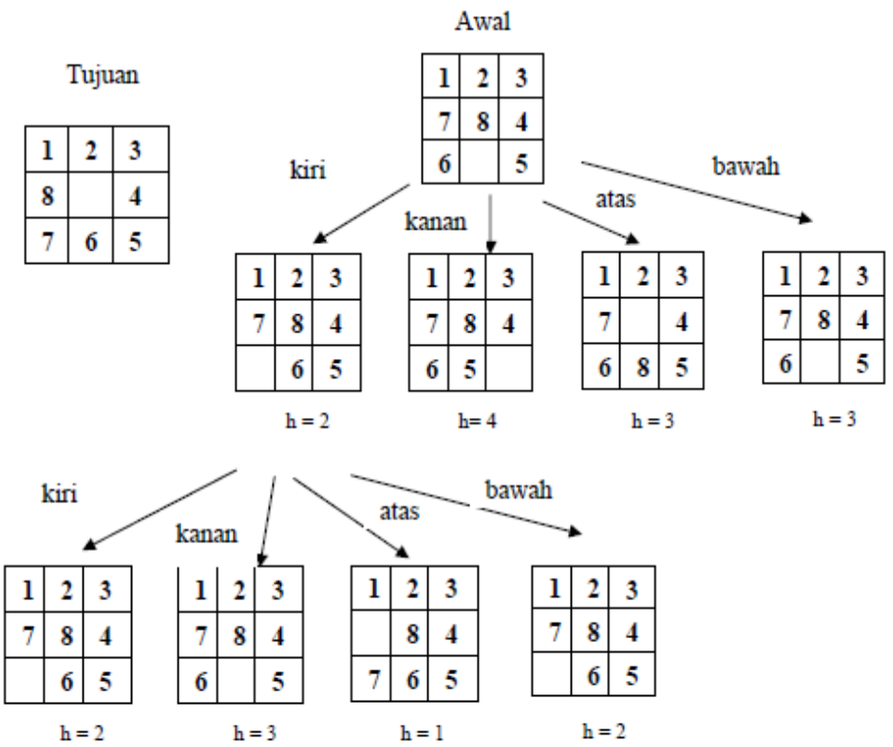


Pada pencarian heuristik perlu diberikan informasi khusus, yaitu :

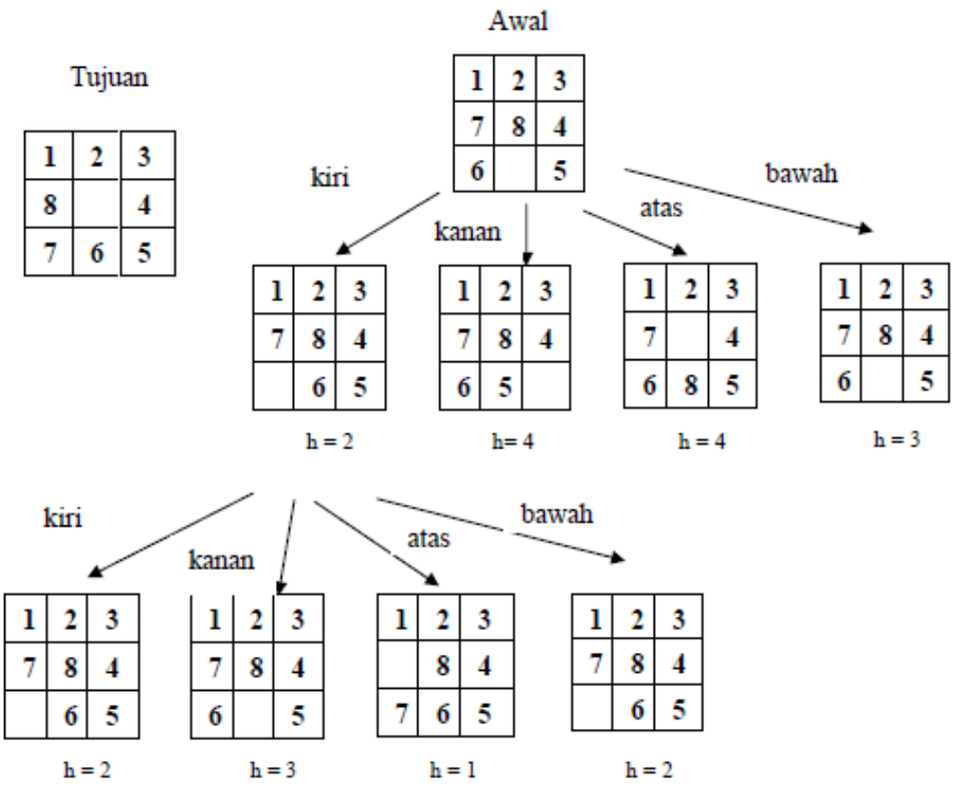
- Untuk jumlah ubin yang menempati posisi yang benar
  Jumlah yang lebih tinggi adalah yang lebih diharapkan (lebih baik)




- Untuk jumlah ubin yang menempati posisi yang salah
  Jumlah yang lebih kecil adalah yang diharapkan (lebih baik)



- Menghitung total gerakan yang diperlukan untuk mencapai tujuan
  Jumlah yang lebih kecil adalah yang diharapkan (lebih baik)



## ALGORITMA BFS :

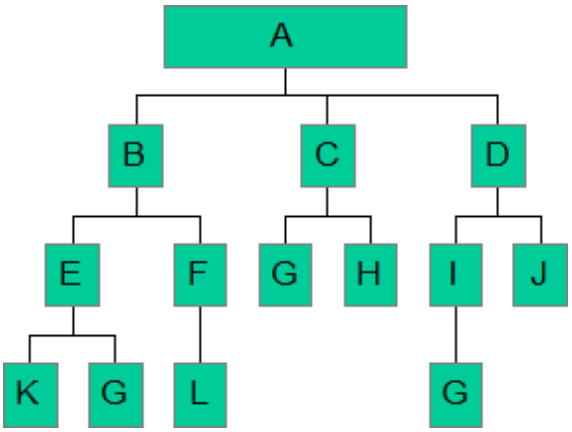
1. Masukkan node akar ke dalam ***queue***
2. Ambil node dari ***queue***, lalu cek apakah node merupakan solusi
3. Jika node merupakan solusi, pencarian selesai & hasil dikembalikan
4. Jika node bukan solusi, masukkan seluruh node anak ke dalam ***queue***
5. Jika ***queue*** kosong & setiap node sdh di cek maka pencarian selesai
6. Jika ***queue*** tidak kosong, ulangi pencarian dari langkah nomor 2

## ALGORITMA DFS :

1. Masukkan node akar ke dalam ***stack***
2. Ambil node dari ***stack*** teratas, lalu cek apakah node merupakan solusi
3. Jika node merupakan solusi, pencarian selesai & hasil dikembalikan
4. Jika node bukan solusi, masukkan seluruh node anak ke dalam ***stack***
5. Jika ***stack*** kosong & setiap node sdh di cek maka pencarian selesai
6. Jika ***stack*** tidak kosong, ulangi pencarian dari langkah nomor 2

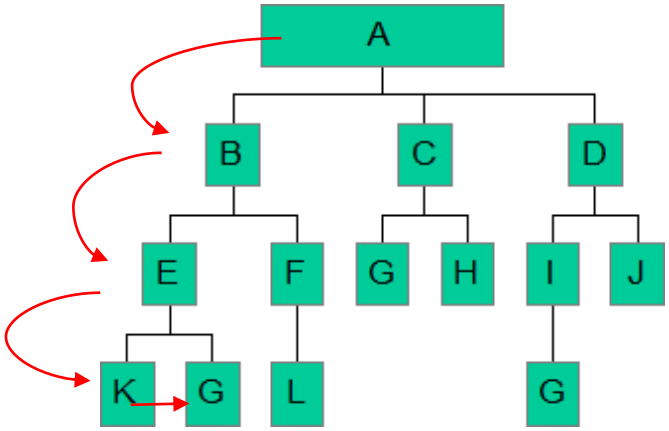
**CONTOH IMPLEMENTASI DFS :**

Perhatikan pohon pelacakan dibawah ini. Ruang pencarian dimulai dari node A, dan berakhir di node G. **Buatlah pelacakan dengan metode Dept First Search.**




**JAWABAN**

- ARAH PELACAKANNYA :




- SILAKAN DI LIHAT ALGORITMA DFS-NYA
- LANGKAH-LANGKAHNYA :