8-Queen n 8-Puzzle

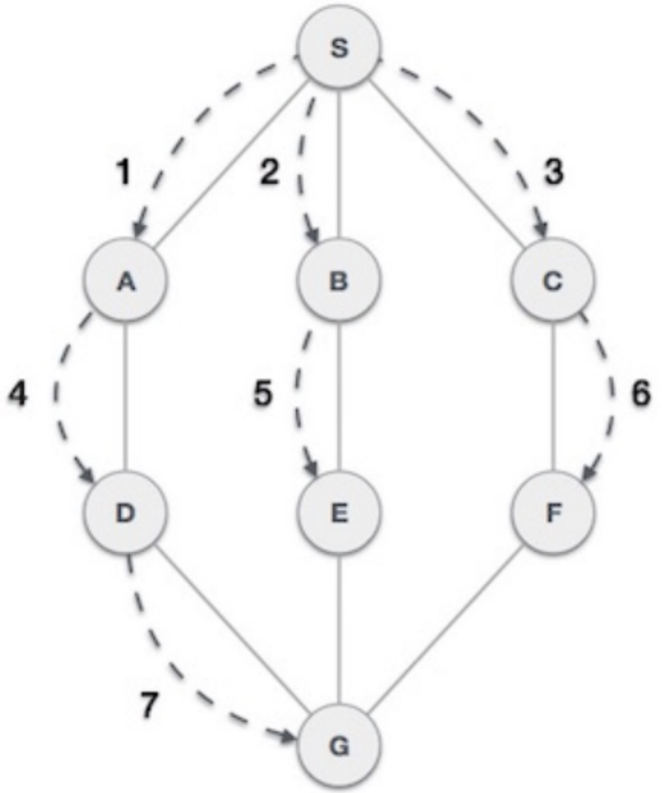
Using traversal graph (DFS&BFS)

5025211061

Helmi Abiyu Mahendra

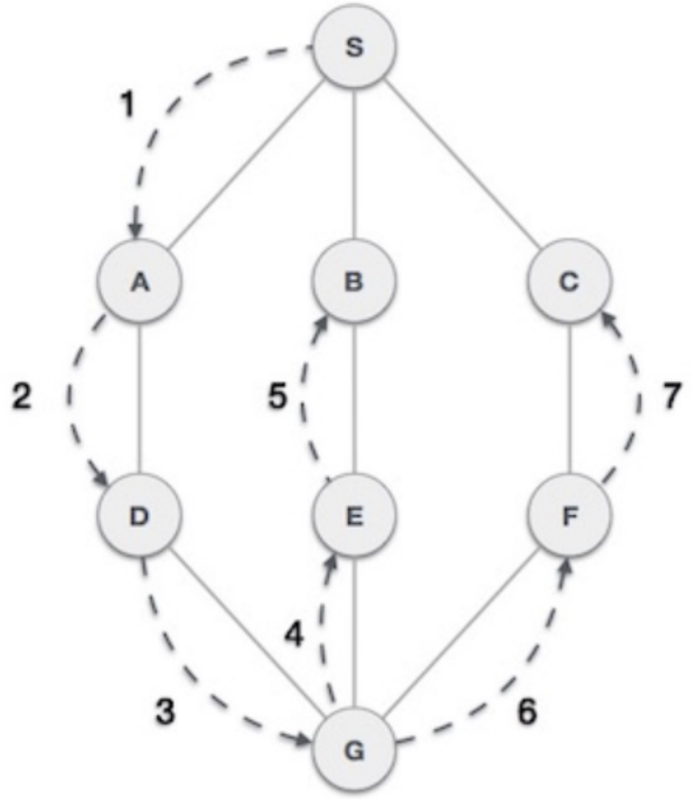
BFS
(Breadth First Search)

Pencarian dengan memeriksa node di sekitar root secara merata



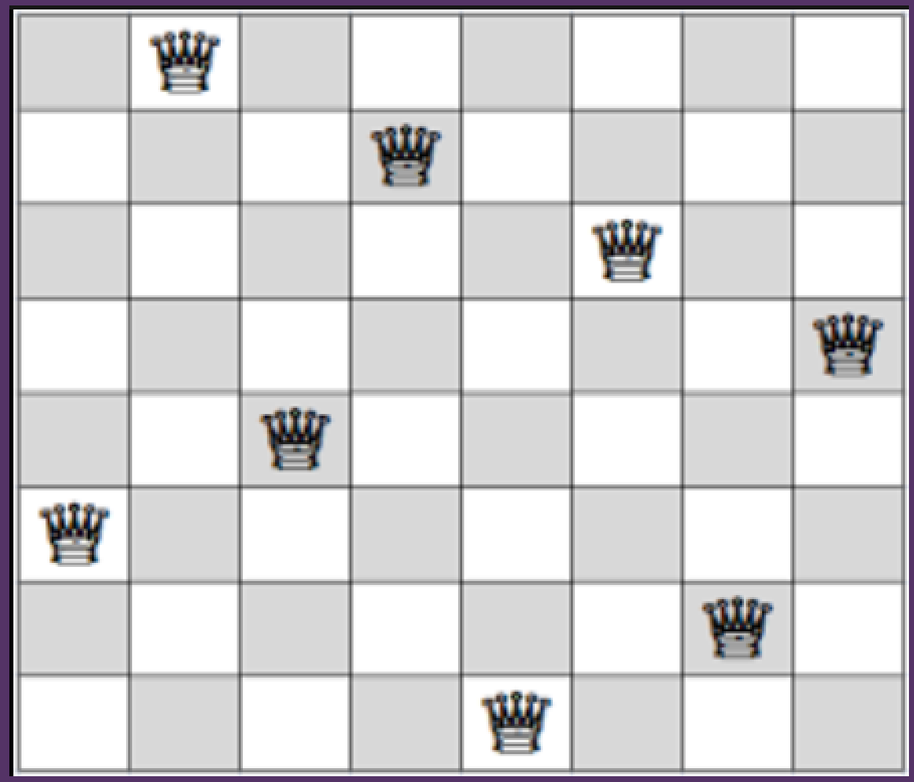
DFS
(Depth First Search)

Pencarian dengan memeriksa node terjauh dari root



8-Queen dengan DFS

Setiap queen tidak boleh ditempatkan pada jalur queen lain.
(jalurnya vertikal, horizontal, dan diagonal)



Helmi Abiyu Mahendra_5025211061

```
void searchCol(int col) {


    for(int i = 0; i < N; i++) {
        //tidak bertabrakan
        if(board[i][col] > -1) {
            //mark pada lokasi queen dan jalurnya
            board[i][col] = 1;
            for(int j = 0; j < N; j++) {
                if(j != col) board[i][j]--;
                if(j != i) board[j][col]--;
            }
            for(int j = 1; j < N; j++) {
                if(i - j >= 0 && col - j >= 0) board[i - j][col - j]--;
                if(i + j < N && col + j < N) board[i + j][col + j]--;
                if(i - j >= 0 && col + j < N) board[i - j][col + j]--;
                if(i + j < N && col - j >= 0) board[i + j][col - j]--;
            }

            Queens++;
            searchCol(col + 1);
```

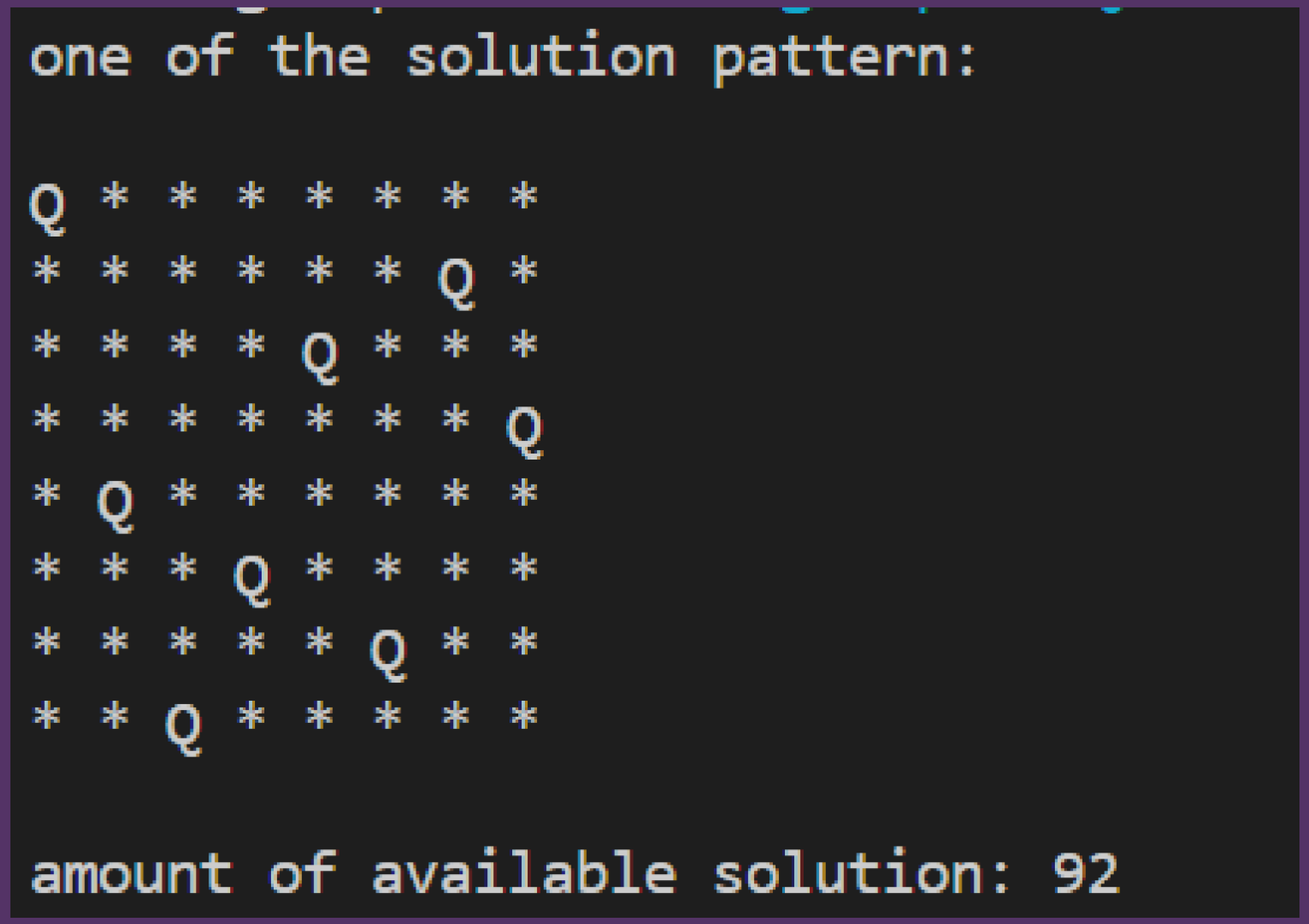
Fungsi yang digunakan untuk menempatkan queens pada jalur yang tidak bertabrakan

```
    //pembersihan papan
        board[i][col] = 0;
        for(int j = 0; j < N; j++) {
            if(j != col) board[i][j]++;
            if(j != i) board[j][col]++;
        }
        for(int j = 1; j < N; j++) {
            if(i - j >= 0 && col - j >= 0) board[i - j][col - j]++;
            if(i + j < N && col + j < N) board[i + j][col + j]++;
            if(i - j >= 0 && col + j < N) board[i - j][col + j]++;
            if(i + j < N && col - j >= 0) board[i + j][col - j]++;
        }
        Queens--;
    }
}
```

Fungsi yang digunakan untuk membersihkan papan dari mark, agar dapat digunakan untuk kombinasi baru

```
void searchCol(int col) {
    if(Queens >= N) {
        if (limout<1){
            cout<<"one of the solution pattern:\n"<<endl;
         for(int i = 0; i < N; i++) {
            for(int j = 0; j < N; j++) {
                if ((board[i][j])==1){
                    cout<<"Q ";
                }
                else cout<<"* ";
                }
            cout << "\n";
        }
        cout << "\n";
        limout++;}
        sols++;
        return;
    }
```

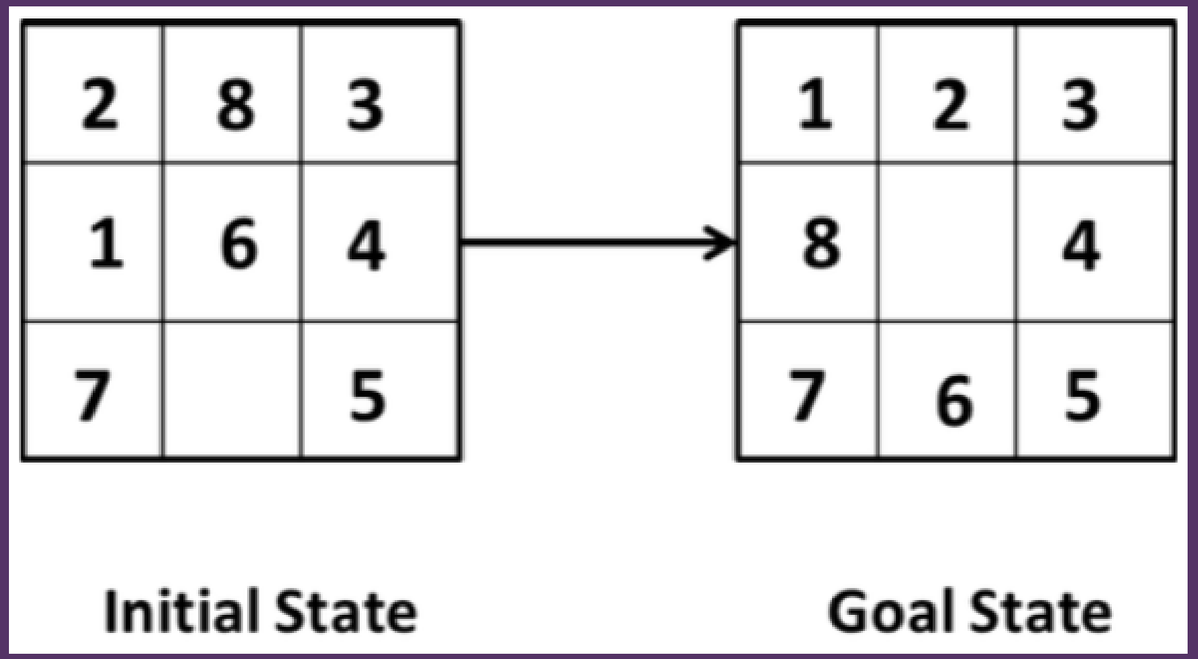
Fungsi untuk menampilkan papan dengan kombinasi queens yang sesuai aturan (dibatasi pencetakan 1 papan dengan limout)



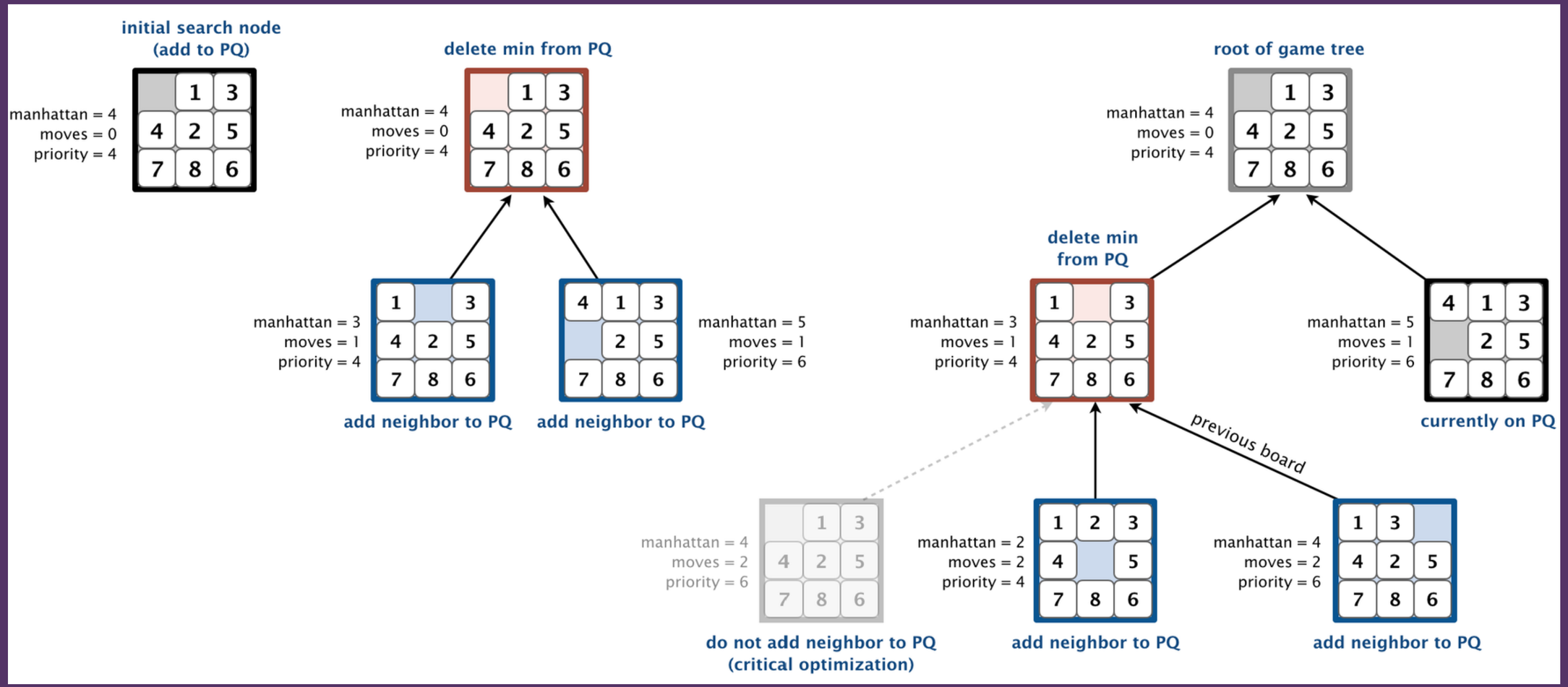
Output berupa 1 contoh pattern, dan jumlah pattern yang dapat dihasilkan berdasarkan jumlah baris dan kolom

8-Puzzle dengan BFS

Setiap angka pada puzzle disusun berurutan (melingkari angka 0), dengan 0 sebagai patokan ruang kosong yang dapat diisi guna melakukan perpindahan



Helmi Abiyu Mahendra_5025211061



Penyusunan puzzle dilakukan dengan switch setiap node yang diperlukan dengan node (kosong) 0

```
//untuk mencari node yang bisa digerakkan
int findzero(){
    for (int i=0; i<9; i++){
        if (a[i]==0) return i;
    }
}
```

Pertama dilakukan pencarian terhadap node isi 0 sebagai target switch

Fungsi expand dengan 4 arah kemungkinan pergeseran node 0

```
//step ekspansi geser
void expand(deque<Node> *deque){
    //temukan node untuk digeser
    int p = s.findzero();
    //moveup    (0)
    if ((p!=0 && p!=1 && p!=2) && action!=1){
        Node n(s.exch(p,p-3), this, 0, depth+1);
        (*deque).push_back(n);
    }
    //movedown  (1)
    if ((p!=6 && p!=7 && p!=8) && action!=0){
        Node n(s.exch(p,p+3), this, 1, depth+1);
        (*deque).push_back(n);
    }
    //moveright (2)
    if ((p!=2 && p!=5 && p!=8) && action!=3){
        Node n(s.exch(p,p+1), this, 2, depth+1);
        (*deque).push_back(n);
    }
    //moveleft  (3)
    if ((p!=0 && p!=3 && p!=6) && action!=2){
        Node n(s.exch(p,p-1), this, 3, depth+1);
        (*deque).push_back(n);
    }
}
```

```
//cek node yang diekspansi
int expanded(deque<State> *deque){
    int max=(*deque).size()>depth?depth:(*deque).size();
    for (int i=0; i<max; i++){
        if ( s.equal( (*deque)[i] ) ){
            return 1;
        }
    }
    return 0;
}
```

Pengecekan yang dilakukan pada setiap tindak ekspansi

Pengecekan terakhir apakah ekspansi dapat diselesaikan, dan tujuannya tercapai

```
int bfs(){
    deque<Node> toexpand;
    deque<State> expanded;

    toexpand.push_back(*this);
    while ( !toexpand.empty() ){
        if ( toexpand.front().s.goal()==1 ){
            cout << "BFS" << endl;
            cout << "Bisa diselesaikan" << endl;
            toexpand.front().print();
            Step = toexpand.front().Step;
            toexpand.clear();
            return Step;
        }
        else{
            if ( !(toexpand.front().expanded(&expanded)) ){
                toexpand.front().expand(&toexpand);
                expanded.push_front( toexpand.front().s );
                toexpand[1].Step=toexpand[0].Step+1;
            }
            toexpand.pop_front();
        }
    }
    if ( toexpand.empty() ) cout << endl << "Tidak bisa diselesaikan" << endl;
    return 0;
}
```

Fungsi goal sebagai patokan pembanding terhadap kondisi susunan node saat ini (dalam fungsi bfs)

```
    //puzzle yang tersusun dengan benar
    // 1    2   3            _>>   _>>    _v
    // 8    0   4            X     0      _v
    // 7    6   5            ^_    <<_   <<_
    int goal(){
        int g[9] = {1,2,3,8,0,4,7,6,5};
        for (int i=0; i<9; i++){
            if (a[i]!=g[i]) return 0;
        }
        return 1;
    }

};
```

Sekian

Terimakasih