1. **Deskripsi Algoritma *Backtracking***

Runut-balik (*backtracking*) adalah algoritma yang

berbasis pada DFS untuk mencari solusi persoalan secara

lebih mangkus. Runut-balik merupakan perbaikan dari

algoritma *exhaustive-search* yang secara sistematis akan

melakukan pencarian solusi permasalahan di antara semua

kemungkinan solusi yang ada. Hanya pencarian yang

mengarah ke solusi saja yang akan dipertimbangkan.

Akibatnya, waktu pencarian dapat dihemat. Runut-balik

lebih alami dinyatakan dalam algoritma rekursif.

Untuk memfasilitasi pencarian ini, maka ruang solusi

diorganisasikan ke dalam struktur pohon. Tiap simpul

pohon menyatakan status (state) persoalan, sedangkan sisi

(cabang) dilabeli dengan nilai – nilai x. Lintasan dari akar

ke daun menyatakan solusi yang mungkin. Seluruh

lintasan dari dari akar ke daun membentuk ruang solusi.



**2. pseudo :**

Boolean cekKolom (int x,int y,int nilai){

// n adalah ukuran papan

Boolean found1= true;

int j = 0;

while (found1 && (j < (n \* n)))

{

if (j == y) {

// do nothing

}

else{

if (getSel(x, j) == nilai){

found1 = false;

}

}

j++; }

Return found1;

}

Boolean cekBaris (int x,int y,int nilai){

// n adalah ukuran papan

Boolean found2= true;

int i = 0;

while (found2 && (i < (n \* n)))

{

if (i == x)

{

// do nothing

}

else{

if (getSel(i, y) == nilai){

found1 = false;

}

}

i++;

}

Return found2;

}

**3. souce code :**

void BackTrack(){

//ArrBool [i,j]==true,artinya kotak tersebut

// telah diisi dengan masukan user

nilai = 1;

int i = 0; int j = 0;

berhasil = true;

while (i < (n \* n) && i >= 0 && berhasil){

while (j < (n \* n) && berhasil){

if (ArrBool[i, j] == false){

while ((!(cekMemenuhi(i, j, nilai))) &&

nilai <= (n \* n))){

nilai++;

setSel(i, j, nilai);

}

if (nilai > (n \* n)){

setSel(i, j, 0);

do{

j--;

if (j < 0){

j += (n \* n);

i--;

if (i < 0){

berhasil = false;

i++;

}

}

}while (ArrBool[i, j] == true);

nilai = getSel(i, j);

setSel(i, j, 0);

nilai++;

}else{

setSel(i, j, nilai);

j++;

nilai = 1;

}

}

else{

j++;

}

}

i++;

j = 0;

}

if (berhasil == false){

output("Tidak ditemukan solusi!");

} else {

output("Solusi berhasil ditemukan!");

}

}

**4. kompleksitas waktu :**