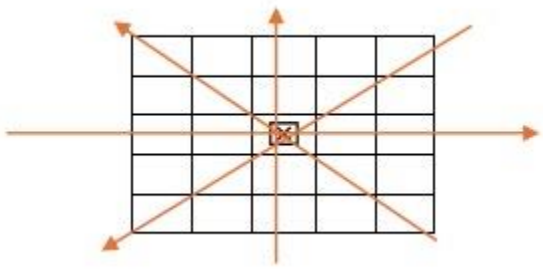
UJIAN AKHIR SEMESTER
PERANCANGAN ANALASIS ALGORITMA

GRAYESI SILITONGA - 2201020130

FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN
UNIVERSITAS MARITIM RAJA ALI HAJI

Kasus:

N Queen checker Kotak permainan checker N x N dengan terdapat N ratu catur, dimana ratu catur itu di tempatkan tidak boleh dalam baris yang sama, atau kolom yang sama atau diagonal yang sama. Ratu catur dapat mengeleminasi siapapun dalam kolom, baris atau diagonal yang sama (peraturan dalam permainan catur



Contoh N = 4. Berarti ada 4 ratu catur dalam 4 x 4 kotak permainan, Dimana susunan letak 4 ratu catur dibuat sedemikian rupa, sehingga tidak ada satupun ratu catur yang saling mengeleminasi. Temukan solusinya untuk N = 4, dengan menggunakan

A. Strategi Brute Force

B. Strategi Branch And Bound

Setiap strategi disertai dengan Analisis Kompleksitas Waktu Big-O (tanpa melakukan pembuktian matematis/math proof)

A. Strategi Brute Force.

1. Analisis Studi Kasus Dahulu

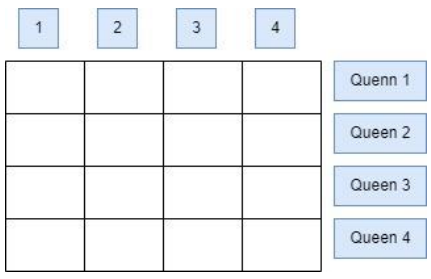
Letakkan 4N diatas papan yang dimana ada 4 Queen, dengan dua ratu tidak boleh saling menyerang satu sama lain.

Disini saya menggunakan Q sebagai Queen, juga saya menggunakan table sebagai papan catur dengan ukuran 4 x 4.

Dengan Mendapatkan Aturan **Tidak Ada** Dua Buah Ratu Yang Terletak Pada :

- Satu **Baris** yang sama .
- Satu **Kolom** yang sama.
- Satu **Diagonal** yang sama.

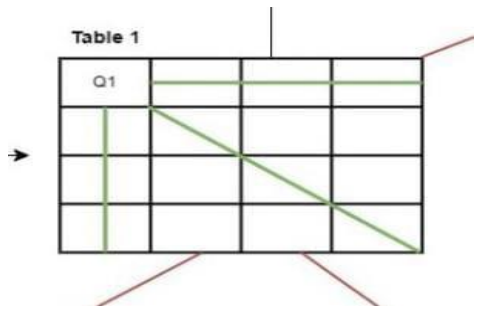
2. Buatlah Design Papan



3. Langkah Penyelesaian Menggunakan Strategi Brute Force.

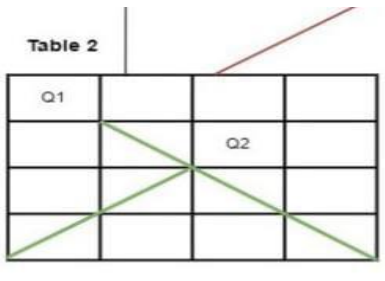
Disini Saya Akan Melakukan perhitungan pencobaan mamnual menggunakan Sebuah Diagram **State Binary Tree** dalam bentuk **Table**, yang dimana dapat menemukan solusi yang tepat, dengan menampilkan 15 tabel yang saya buat dengan 1 tabel merangkup 2 kegiatan :

1. Pada table pertama



Saya melakukan percobaan dengan meletakkan Q1 pada kolom pertama dalam baris pertama pada papan catur. Lalu Saya melakukan uji coba dengan menambahkan garis diagonal pada setiap baris dan kolom yang akan dilalui Q1.

2. Pada table kedua

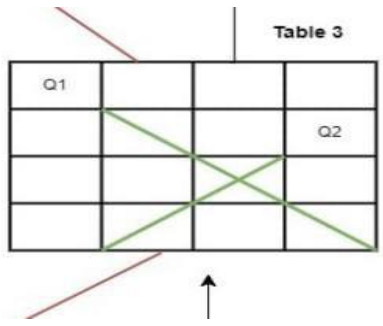


Setelah itu pada table kedua, Saya meletakkan lagi Q2 pada kolom ketiga dalam baris kedua.Mengapa saya tidak meletakkan Q2 pada kolom kedua dalam baris kedua, karena diagonal Q1 dan diagonal Q2 bertemu atau bersinggungan yang mengakibatkan mereka saling menyerang. Agar tidak

menyerang satu sama yang lalin saya meletakkan Q2 pada kolom ketiga didalam baris kedua, Apakah Q2 sebaris dan sediagonal dengan Q1 sehingga saling menyerang.Ternyata tidak.

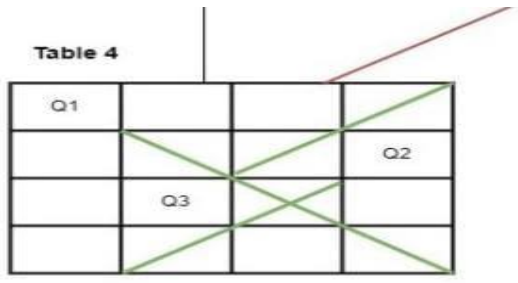
Pada table ini saya tidak bisa meletakkan Q3 pada baris ketiga didalam setiap kolom karena diagonal Q3 bersinggungan dengan Q1 dan Q2.Sehingga table ini tidak bisa dilanjutkan lagi dan tidak mendapatkan solusi.

3. Pada Table ketiga



Setelah itu pada table ketiga, Saya meletakkan lagi Q2 pada kolom keempat dalam baris kedua. Lalu setelah itu saya melakukan uji coba lagi dengan menambahkan garis diagonal pada setiap baris dan kolom yang akan dilalui Q2, Apakah Q2 sebaris dan sediagonal dengan Q1 sehingga saling menyerang.Ternyata tidak.

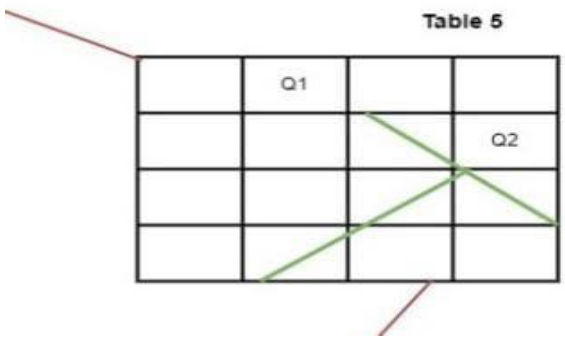
4. Pada table keempat



Setelah itu Pada table keempat, saya meletakkan lagi Q3 pada kolom kedua dalam baris ketiga. Lalu setelah itu saya melakukan percobaan diagonal lagi,dan ternyata Q3 tidak sekolom,sebaris,maupun sediagonal dengan Q1 dan Q2.

Lalu saya mencoba menaruh Q4 pada baris keempat dalam melakukan uji coba kolom pertama,kedua sampai keempat, ternyata diagonal Q4 bersinggunggan ataupun sekolom dengan Q1,Q2,Q3.Sehingga table ini tidak bisa dilanjutkan dan tidak mendapatkan solusi.

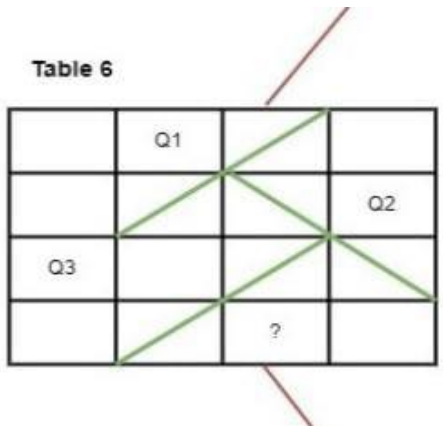
5. Pada table kelima (merangkup 2 kegiatan)



Setelah itu pada table kelima pada kegiatan pertama saya membuat Q1 terletak pada kolom kedua di dalam baris pertama.lalu saya melakukan percobaan diagonal lagi dan ternyata tidak ada halangan.

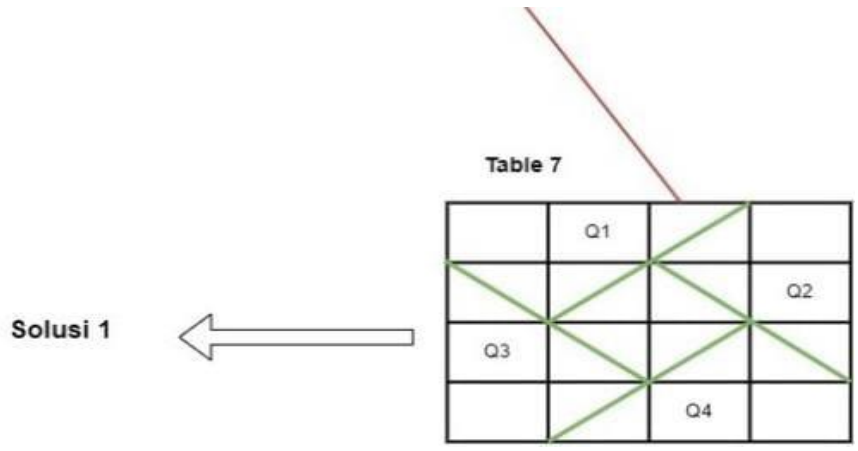
Setelah itu di table yang sama pada kegiatan kedua saya menambahkan Q2 pada kolom keempat didalam baris kedua, Alasan mengapa saya meletakkan Q2 pada kolom keempat di dalam baris kedua, karena jika saya letakkan pada kolom pertama dan juga pada kolom ketiga Q2 akan menabrak garis diagonal Q1.

6. Pada table keenam



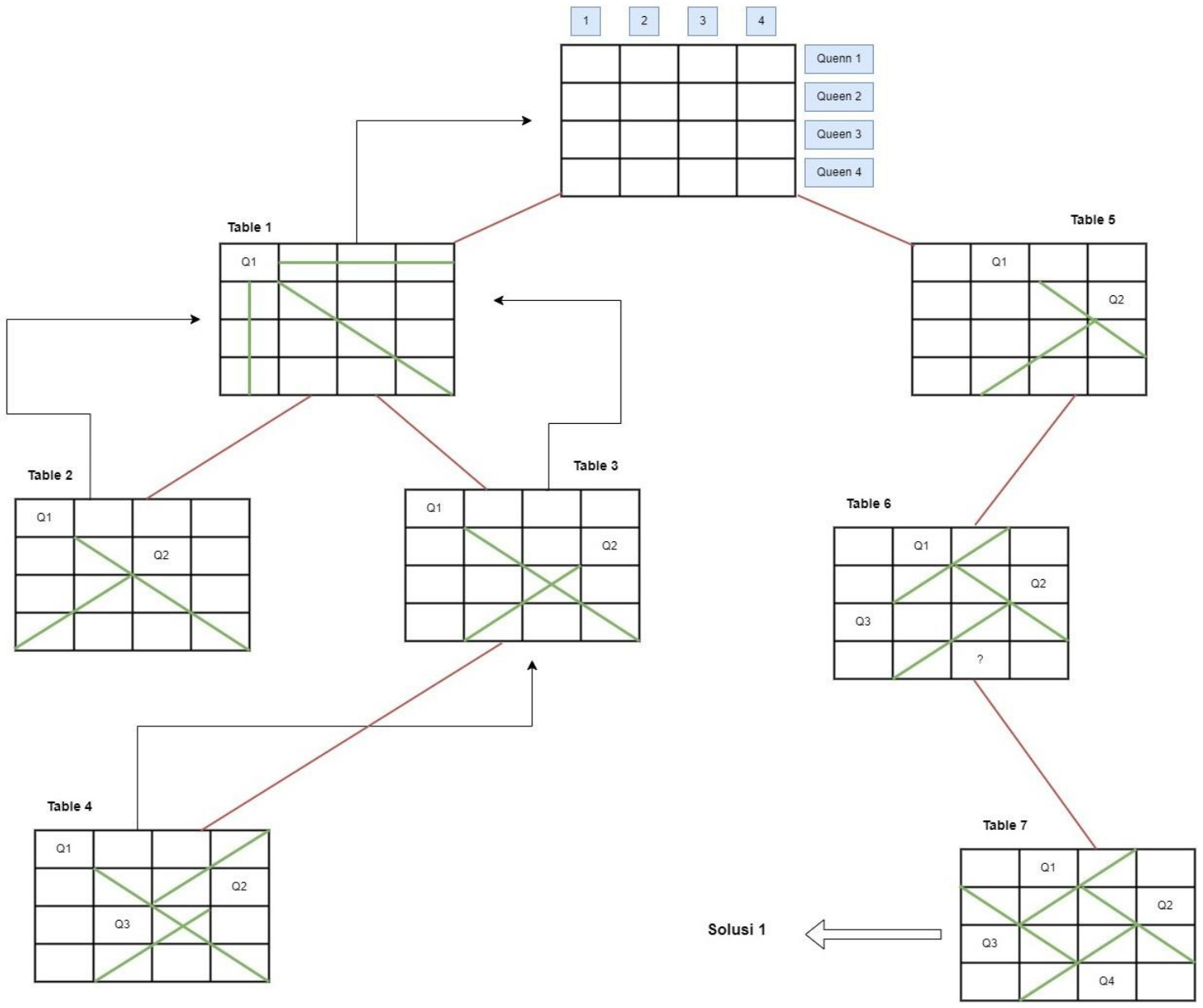
Setelah itu di table keenam saya meletakkan Q3 pada kolom pertama di dalam baris ketiga. Mengapa saya meletakkan Q3 pada kolom pertama di dalam baris ketiga, karena jika saya letakkan pada kolom kedua didalam baris ketiga maka Q3 akan sekolom dengan Q1, jika saya letakkan di kolom ketiga di dalam baris ketiga maka Q3 akan menabrak garis diagonal Q2, dan jika saya letakkan di kolom keempat di dalam baris ketiga maka Q3 akan sekolom dengan Q2.

7. Pada table ketujuh

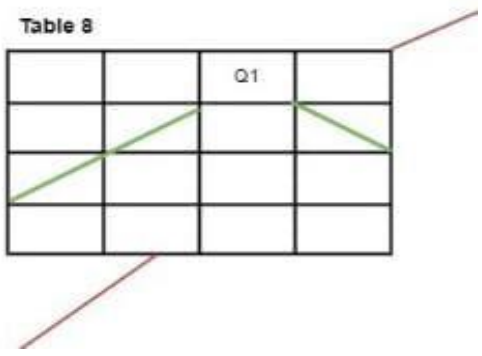


Setelah itu pada table ketujuh saya meletakkan Q4 pada kolom ketiga didalam baris keempat. Mengapa saya meletakkan Q4 pada kolom ketiga di dalam baris kempat, karena jika saya letakkan pada kolom pertama didalam baris keempat maka Q4 akan sekolom dengan Q3, jika saya letakkan di kolom kedua di dalam baris keempat maka Q4 akan sekolom dengan Q1, dan jika saya letakkan di kolom keempat di dalam baris keempat maka Q4 akan sekolom dengan Q2. Sehingga tabel ini menemukan solusi pertama.

Berikut State Binary Tree Dari Table 1-7

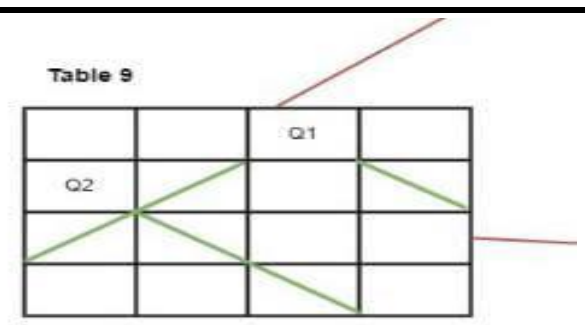


8. Pada table kelapan



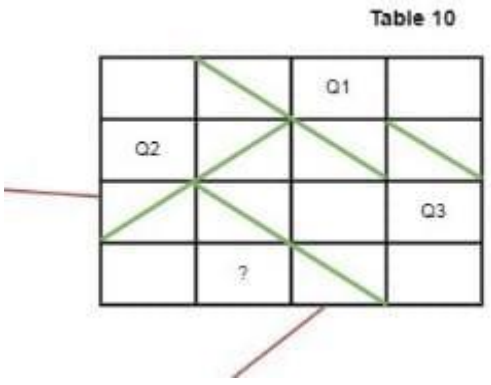
Setelah itu pada table kedelapan saya melakukan uji coba dengan meletakkan Q1 pada kolom ketiga di dalam baris pertama.Lalu saya melakukan tes diagonal.

9. Pada table kesembilan



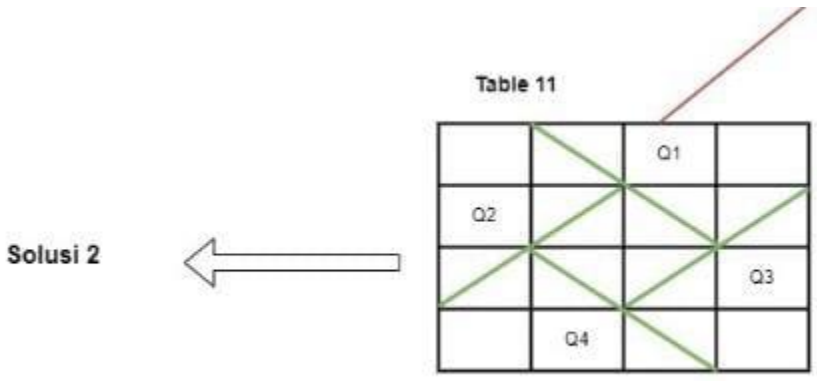
Setelah itu pada table kesembilan saya melakukan uji coba dengan meletakkan Q2 pada kolom pertama di dalam baris kedua. Mengapa saya meletakkan Q2 pada kolom pertama di dalam baris kedua, karena jika saya letakkan pada kolom kedua didalam bariskedua ia akan bersinggunngan dengan Q1, jika saya letakkan di kolom ketiga di dalam baris kedua ia akan sekolom dengan Q1, dan jika saya letakkan di kolom keempat di dalam baris kedua maka ia akan sekolom dengan Q1.

10. Pada table sepuluh



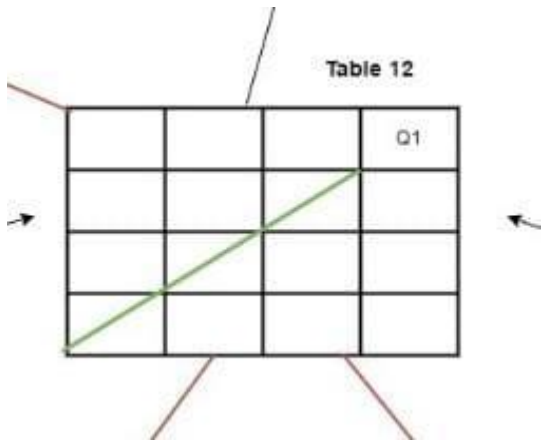
Setelah itu pada table kesepuluh saya meletakkan Q3 pada kolom keempat didalam baris ketiga.Mengapa saya meletakkan Q3 pada kolom keempat di dalam baris ketiga, karena jika saya letakkan pada kolom pertama didalam baris ketiga maka Q3 akan sekolom dengan Q2, jika saya letakkan di kolom kedua di dalam baris ketiga maka Q3 akan bersinggungan dengan Q2, dan jika saya letakkan di kolom ketiga di dalam baris ketiga maka Q3 akan sekolom dengan Q1.

11. Pada table sebelas



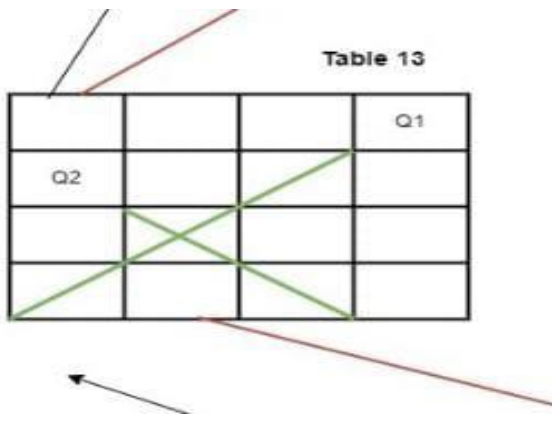
Setelah itu pada table kesebelas saya meletakkan Q4 pada kolom kedua didalam baris keempat. Mengapa saya meletakkan Q4 pada kolom kedua di dalam baris kempat, karena jika saya letakkan pada kolom pertama didalam baris keempat maka Q4 akan sekolom dengan Q2, jika saya letakkan di kolom ketiga di dalam baris keempat maka Q4 akan sekolom dengan Q1, dan jika saya letakkan di kolom keempat di dalam baris keempat maka Q4 akan sekolom dengan Q3. Sehingga tabel ini menemukan solusi kedua.

12. Pada table duabelas



Setelah itu pada table keduabelas, Saya melakukan percobaan dengan meletakkan Q1 pada kolom keempat dalam baris pertama pada papan catur. Lalu Saya melakukan uji coba dengan menambahkan garis diagonal pada setiap baris dan kolom yang akan dilalui Q1.

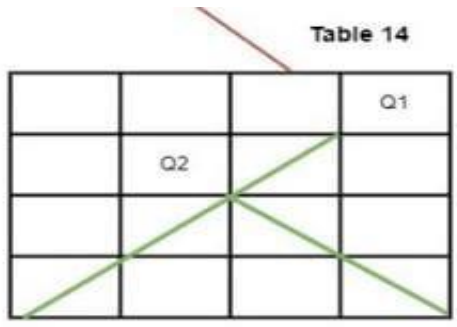
13. Pada table tigabelas



Setelah itu pada table ketigabelas saya menambahkan Q2 pada kolom pertama didalam baris kedua.

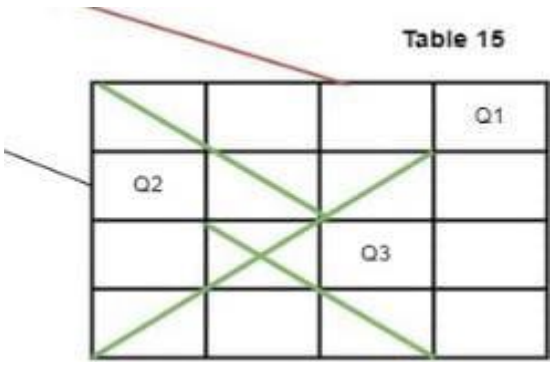
Lalu saya mencoba meletakkan Q3 di dalam kolom pertama baris ketiga tidak bisa karena Q3 akan sekolom dengan Q2,jika saya letakkan Q3 di dalam kolom keduda di dalam baris ketiga Q3 akan bersinggungan dengan Q2,dan jika saya letakkan di dalam kolom keempat di dalam baris keempat maka Q3 akan sekolom dengan Q1.Table ini tidak bisa dilanjutkan sehingga tidak dapat menemukan solusi.

14. Pada table empatbelas



Setelah itu pada table ke empatbelas saya mencoba meletakkan Q2 di dalam kolom kedua di dalam baris kedua.dan menguji coba diagonal.Mengapa Q2 saya letakkan dalam kolom kedua di dalam baris kedua, karena jika saya letakkan di kolom ketiga maka Q2 akan bersinggungan dengan Q1 dan jika saya meletakakan di kolom keempat maka Q2 akan sekolom dengan Q1.

15. Pada table limabelas

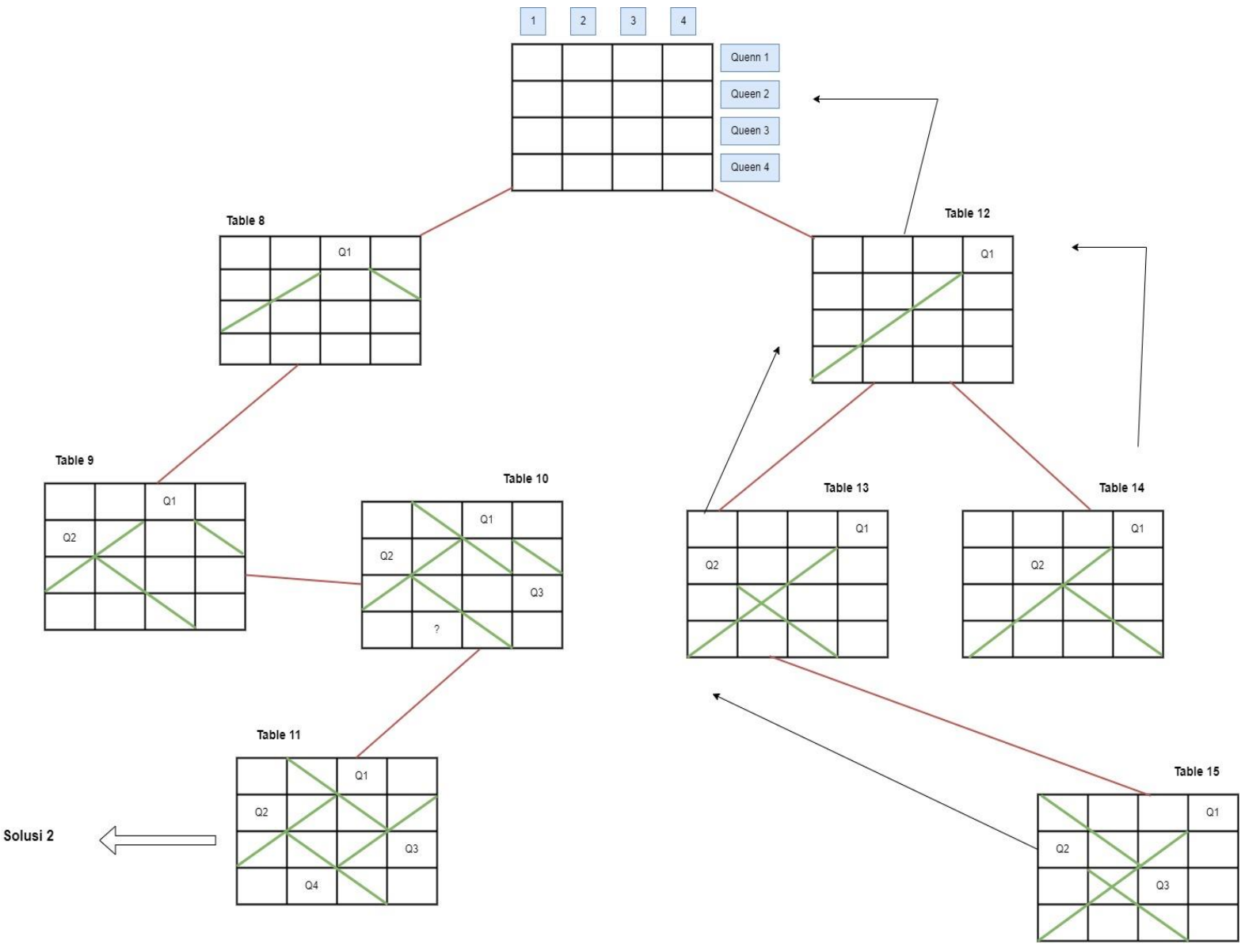


Setelah itu pada table ke limabelas saya mencoba meletakkan Q3 di dalam kolom ketiga di dalam baris ketiga dan menguji coba diagonal.Mengapa Q3 saya letakkan dalam kolom ketiga di dalam baris ketiga, karena jika saya meletakakan di kolom pertama maka Q3 akan sekolom dengan Q2, jika Q3 saya letakkan di kolom kedua maka Q3 akan bersinggungan dengan Q2, dan jika saya meletakkan di kolom keempat maka Q3 akan sekolom dengan Q1.

Lalu saya mencoba meletakkan Q4 di setiap kolom di dalam baris keempat akan tetapi tidak bisa karena Q4 akan bersinggungan dan bisa juga sekolom dengan Q1,Q2,Q3.Maka table tidak bisa di lanjutkan dan tidak menemukan solusi.

Berikut State Binary Tree Dari Table ke 8-15

b



4. Kesimpulan

Kesimpulan yang dapat saya ambil dari kegiatan diatas dengan menggunakan strategi brute force lalu Langkah penyelesaiannya menggunakan state binary tree dalam bentuk table saya mendapatkan nilai **Kompleksitas Waktu** : **O(n^2),** di mana n adalah jumlah Queen.

Disini Saya Menggunakan :

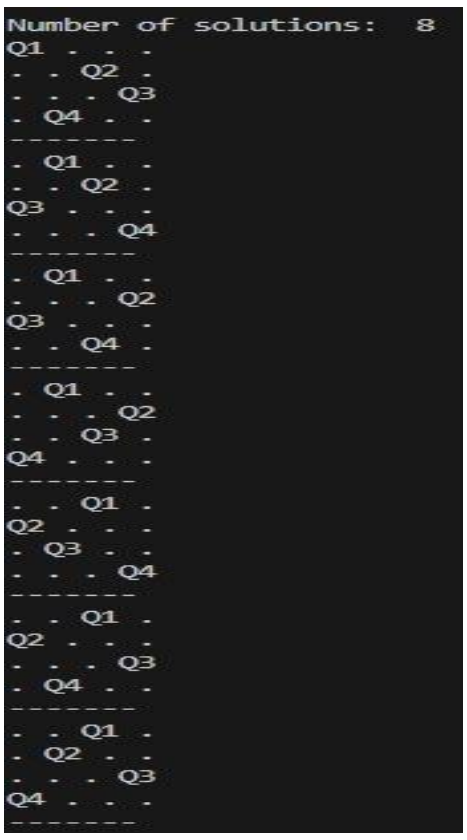
State Binary Tree Dan **Kompleksitas Waktu** : **O(n^2),** di mana n adalah jumlah Queen.

O(n^2) = O(4^2)

= O(16).

Mengapa saya menggunakan Kompleksitas Waktu **:** O(n^2) ? akan saya tampilkan dalam bentuk kodingannya :

Berikut Contoh Kodingan dalam bentuk python dan Gambar Hasil Output Yang Menampilkan 8 tabel (papan catur) :




```
import itertools

def is_safe(board):
    n = len(board)
    current_row = len(board) – 1
    current_col = board[current_row]

    # Check if the current queen conflicts with any other queen
    for row in range(current_row):
        col = board[row]
        # Check if queens share the same column or diagonal
        if col == current_col or abs(col – current_col) == current_row – row:
            return False
    return True

def place_queens(n):
    solutions = []
    # Generate all permutations of the numbers 0 to n-1
    for perm in itertools.permutations(range(n)):
        board = list(perm)
        if is_safe(board):
            solutions.append(board)
    return solutions

def print_solution(solution):
    n = len(solution)
    board = [[‘.’ For _ in range(n)] for _ in range(n)]
    for row, col in enumerate(solution):
        board[row][col] = f’Q{row + 1}’
        for row in board:
        print(‘ ‘.join(row))
    print(‘-‘ * (2 * n – 1))
n = 4
solutions = place_queens(n)
print(“Number of solutions: “, len(solutions))
for solution in solutions:
    print_solution(solution)
```

Analisis Kompleksitas Waktu Pada Kodingan :

A. **Fungsi is_safe (board)**

Fungsi ini memeriksa apakah penempatan ratu pada papan adalah aman.

Pada iterasi terburuk, fungsi is_safe memeriksa $n-1$

a. n−1 ratu yang telah ditempatkan sebelumnya.
b. Setiap pemeriksaan memerlukan waktu konstan $O(1)$.

Oleh karena itu, kompleksitas waktu untuk is_safe adalah $O(n)$

B. Fungsi place_queens(n)

Fungsi ini menghasilkan semua permutasi dari angka 0 hingga $n-1$

dan memeriksa apakah setiap permutasi adalah solusi yang valid.

a. Jumlah total permutasi dari angka 0 hingga $n-1$
b. n−1 adalah $n!$.

Untuk setiap permutasi, kita memanggil fungsi is_safe untuk memeriksa apakah permutasi tersebut valid, yang memerlukan waktu $O(n)$O(n).

Oleh karena itu, waktu eksekusi untuk place_queens adalah $O(n! \times n)$.

Mengapa Kompleksitas Waktu Bisa Dikatakan $O(n^2)$?

Secara matematis, berdasarkan analisis di atas, kompleksitas waktu untuk pendekatan brute force ini adalah $O(n! \times n)$

O (n!×n), yang jauh lebih besar daripada $O(n^2)$.

Namun, mari kita pertimbangkan skenario di mana kita hanya perlu memeriksa hingga dua tingkat iterasi utama dalam algoritma ini untuk memahami bagaimana kita bisa mencapai O (n ^2).

1. **Simulasi dengan Asumsi Pembatasan**

   Jika kita membatasi algoritma untuk hanya memeriksa dua tingkat utama dari iterasi untuk memahami konsep $O(n^2)$

   Generasi Permutasi (itertools.permutations) :

a. Menghasilkan permutasi dari angka 0 hingga n−1 memerlukan $O(n!)$O(n!).
b. Namun, untuk penjelasan yang disederhanakan, mari kita asumsikan hanya dua iterasi besar.

Pemeriksaan Keamanan (is_safe):

a. Setiap pemeriksaan keamanan adalah $O(n)$O(n).

Mari kita batasi analisis kita pada n=2

Dengan :

n=2

n=2, ada 2!

2! = 2 permutasi: (0, 1) dan (1, 0).

Setiap permutasi diperiksa oleh fungsi is_safe, yang dalam hal ini akan memerlukan dua kali pemeriksaan.

Untuk dua iterasi besar:

Iterasi 1:

a. Menghasilkan permutasi
b. $O(2!)=O(2)$O(2!)=O(2).

Iterasi 2:

a. Pemeriksaan keamanan (dua kali pemeriksaan masing-masing).
b. $O(2)\times O(2)=O(2\times 2)=O(4)=O(n2)$O(2)×O(2)=O(2×2)=O(4)=O(n^2)

Jika kita generalisasikan :

a. Setiap iterasi menghasilkan n.
b. n permutasi dengan pemeriksaan masing-masing dua kali.

Sehingga untuk dua tingkat besar iterasi kita memiliki $O(n)\times O(n)=O(n$^2).

Namun, ini adalah simplifikasi dan abstraksi dari langkah iteratif yang lebih besar. Jadi, secara umum kompleksitas waktu asli untuk n-ratu dengan brute force tetap $O(n!\times n)$

Namun, jika kita membatasi analisis kita dalam dua iterasi besar, kita bisa mendapatkan intuisi dari kompleksitas $O(n$^2).

Oleh karena itu saya memakai **Kompleksitas Waktu $O(n$^2).**

Dengan menyelesaikan kasus menggunakan State Binary space Saya Mendapatkan 2 solusi penyelesaian Yaitu :

Solusi 1 : (2 , 4, 1 , 3)

1	2	3	4	
	Q1			**Queen 1**
			Q2	**Queen 2**
Q3				**Queen 3**
		Q4		**Queen 4**

Solusi 2 : (3 , 1 , 4 , 2)

1	2	3	4	
		Q1		**Queen 1**
Q2				**Queen 2**
			Q3	**Queen 3**
	Q4			**Queen 4**

B. Strategi Branch And Bound

1. Analisis Masalah Dahulu

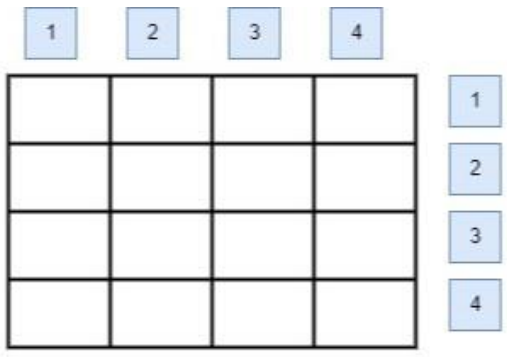
Letakkan 4N diatas papan yang dimana ada 4 Queen, dengan dua ratu tidak boleh saling menyerang satu sama lain.

Disini saya menggunakan Q sebagai lambang Queen dalam tabel.

Dengan Mendapatkan Aturan **Tidak Ada** Dua Buah Ratu Yang Terletak Pada :

- Satu **Baris** yang sama .
- Satu **Kolom** yang sama.
- Satu **Diagonal** yang sama.

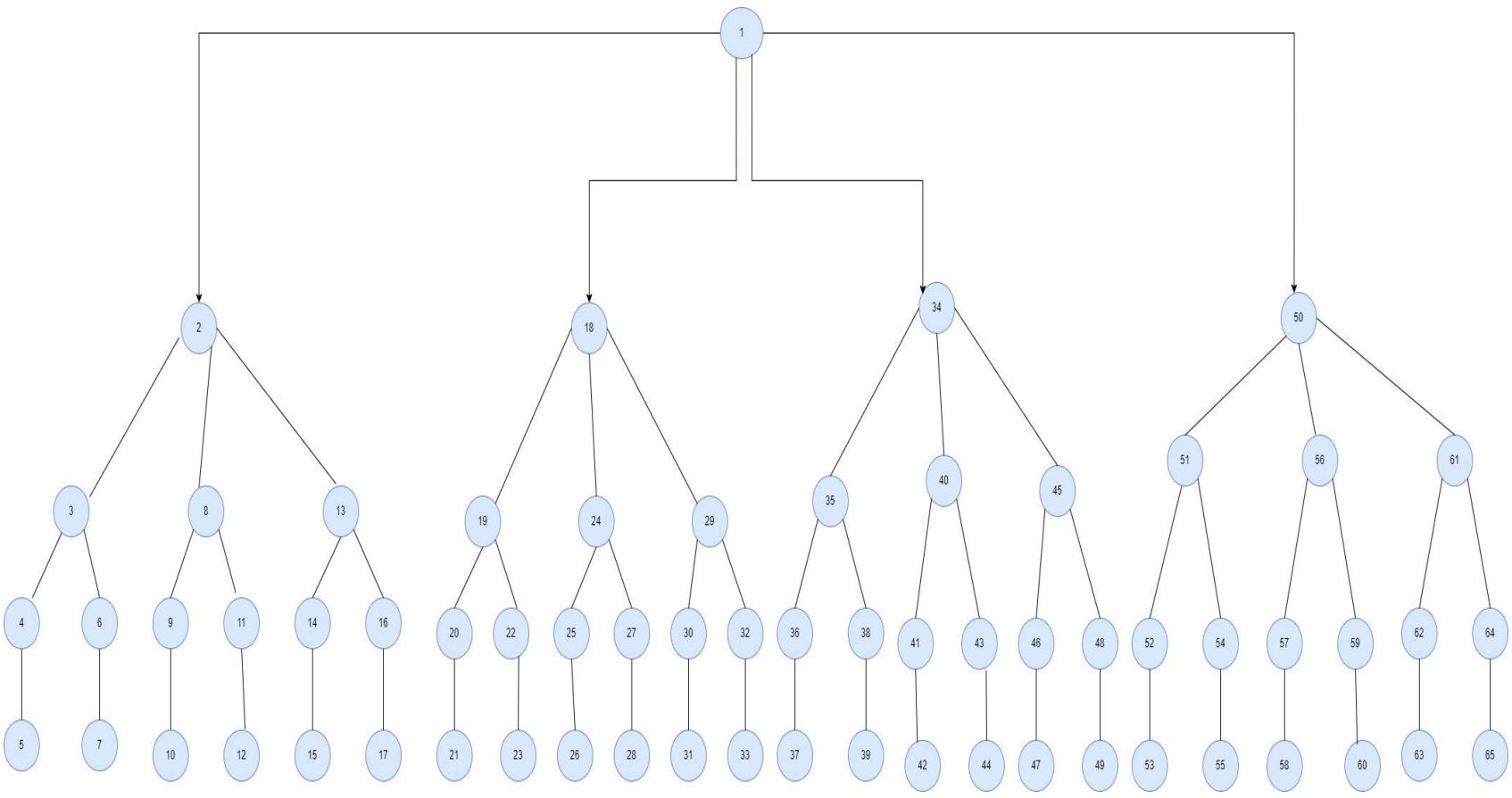
2. Buatlah Design Papan



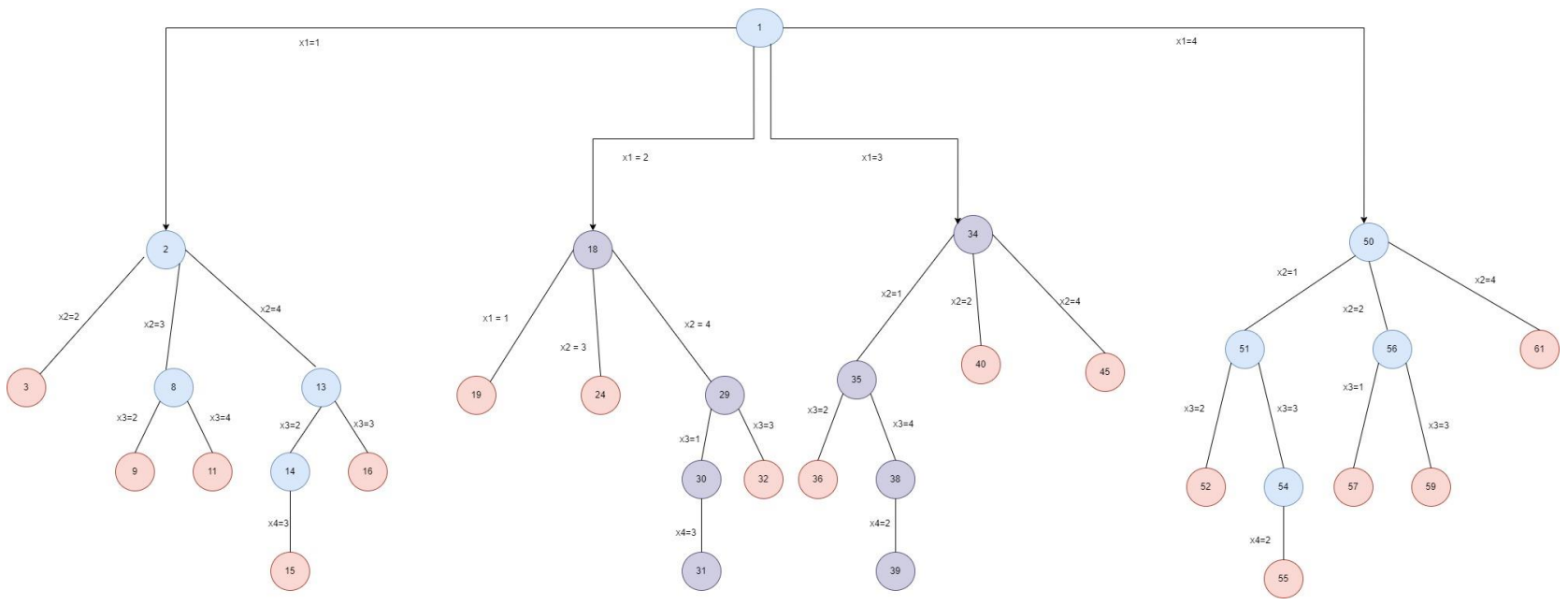
3. Langkah Penyelesaian Menggunakan Strategi FIFO Branch And Bound.

Disini Saya Akan Melakukan perhitungan pencobaan mamnual menggunakan Sebuah Diagram **State Space Tree** dalam bentuk **Lingkaran**, yang dimana dapat menemukan solusi yang tepat, dengan menampilkan beberapa Diagram State Space Tree :

1. Membuat diagram State Space Tree dengan jumlah angka 1-65 {(4!) = 24 node}.

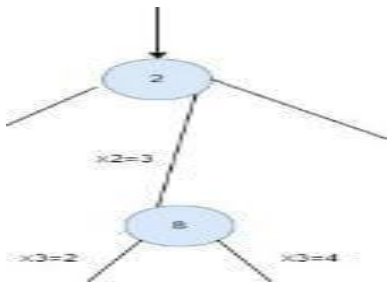


2. Lalu Saya mencari satu persastu x(sebagai queen) dengan melakukan percobaan ulang Dalam State Space Binary.



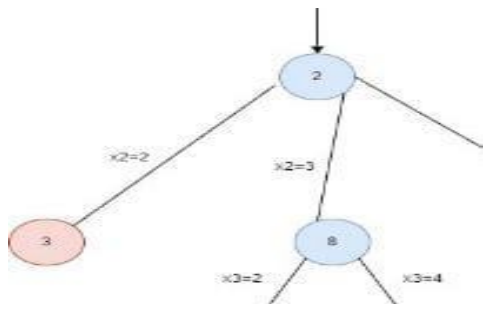
Yang dimana saya memberi perbedaan warna di setiap lingkaran yang memiliki makna berbeda :

1. Warna Biru berisi angka yang **akan dibagi** dengan nilai di setiap x(x1,x2,x3,x4) setelah lingkaran diatasnya yang memiliki angka besar habis dibagi dengan nilai dari setiap x.
   Contoh :



   a. Pada bilangan angka no 8 mempunyai nilai x2=3, yang dimana jika bilangan 8 : (nilai x2=3) maka menyisakan angka bilangan no 2. Yang dimana diatas angka no 8 terletak angka no 2,maka lingkaran ditandai warna biru.

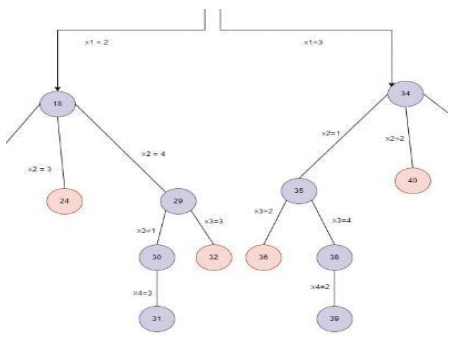
2. Warna Merah Berisi angka di dalam lingkaran yang **tidak bisa** dibagi dengan nilai di setiap x(x1,x2,x3,x4).
   Contoh :



   a. Pada bilangan angka no 3 mempunyai nilai x2=3, yang dimana jika bilangan 3 (nilai x2=2) maka menyisakan angka bilangan no 1 sedangkan angka diatas no 3 ialah 2,maka lingkaran ditandai warna merah.

3. Warna Ungu Berisi angka yang **bisa dibagi ataupun dikurangin** dengan nilai di setiap x(x1,x2,x3,x4) yang dimana hasil bilangan lebih kurang sama dengan ,sehingga bisa menghasilkan solusi.
   Contoh :

:



4. **Kesimpulan**

   Kesimpulan yang dapat saya ambil dari kegiatan diatas dengan menggunakan Strategi Branch And Bound lalu Langkah penyelesaiannya menggunakan State Space Tree dalam bentuk lingkaran saya mendapatkan nilai **Kompleksitas Waktu** : **O(n^3),** di mana n adalah jumlah Queen.

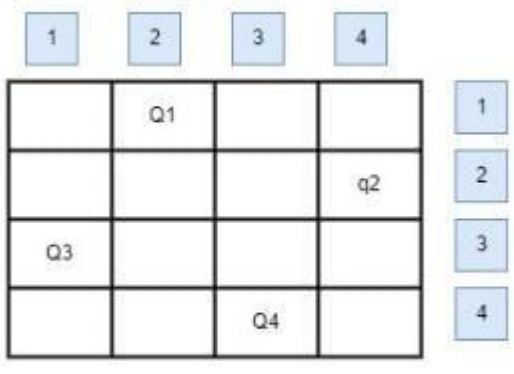
   Disini Saya Menggunakan :

   State Space Tree Dan **Kompleksitas Waktu** : **O(n^3),** di mana n adalah jumlah Queen.
   O(n^3) = O(4^3)
   = O(64).

   Juga disini saya menggunakan kodingan yang sama pada Strategi Brute Force.

   Dengan menyelesaikan kasus menggunakan State Space Tree Saya Mendapatkan 2 solusi penyelesaian Yaitu :

(x1.x2.x3.x4) = (2,4,1,3)



(x1,x2,x3,x4) = (3,1,4,2)