Vol. 01,No. 01 (2019),pp. xx-xx
p-ISSN 2656-6303
URL:http://journal.unirow.ac.id/index.php/mv

PERBANDINGAN ALGORITMA *SYSTEMS OF DISTINCT REPRESENTATIVE* (SDR) DENGAN *BACKTRACKING* DALAM *N-ROOK PROBLEM*

Zaidatun Ni'mah[1], Zainullah Zuhri*[2]
Matematika UIN Sunan Ampel[1,2]
E-Mail zainullah.zuhri@gmail*

***Abstrak*—*Rook* (benteng) merupakan salah satu pion dalam permainan catur yang perlu diperhatikan penempatannya agar tidak dimakan oleh lawan. Oleh karena itu dalam permainan catur, diperlukan beberapa strategi untuk memenangkannya. Salah satu konsep dalam matematika yang dapat diterapkan dalam penempatan *rook* pada permainan catur adalah *Systems of Distinct Representative* (SDR). Selain menggunakan konsep SDR, konsep lain yang dapat digunakan adalah *Backtracking*. Pada penelitian sebelumnya konsep atau algoritma *Backtracking* telah digunakan untuk mencari penempatan *rook*. Oleh karena itu diperlukan sebuah perbandingan antara dua konsep tersebut. Hasil yang didapat menunjukkan bahwa kedua konsep tersebut memberikan solusi yang sama yaitu terdapat dua himpunan solusi untuk penempatan *rook* dalam permainan catur berdasarkan data yang digunakan, diantaranya $Y_1 = \{1,4,3,2\}$ dan $Y_2 = \{4,2,3,1\}$.**

***Kata kunci*—Backtracking, Systems of Distinct Representative.**

I. PENDAHULUAN

Rook (benteng) merupakan salah satu pion dalam permainan catur yang dapat bergerak secara horizontal atau vertikal. Apabila benteng tersebut terletak dalam baris atau kolom yang sama dengan benteng lawan maka akan dimakan oleh benteng lawan [1]. Oleh karena itu setiap pemain harus memiliki strategi tertentu agar pion-pionnya tidak dimakan oleh pion lawan khususnya pion *rook* karena keberadaannya yang didepan dan mudah dimakan.

Penelitian terkait mengenai cara menentukan langkah *rook* dalam permainan catur yaitu penelitian Herman (2011) yang menunjukkan bahwa deskripsi langkah benteng menggunkaan algoritma runut balik dalam papan catur $n \times n$ adalah sebanyak $p(n) = (n-0)(n-1)(n-2)\ldots(n-(n-1))$ dimana $p(n)$ berlaku untuk $n \leq 3$ [2]. Penelitian terkait lainnya yaitu penelitian Arinta (2007) mengenai Penggunaan Algoritma *Backtracking* dalam Pencarian Koefisien *Rook Polynomial* yang menunjukkan bahwa metode *rook polynomial* mampu menyelesaikan banyak permasalahan kombinatorika yang menyangkut perhitungan permutasi dari suatu persoalan dengan beberapa area terlarang dan *algoritma backtracking* cukup mangkus dalam penyelesaian permasalahan tersebut [1].

Berdasarkan penelitian sebelumnya yang telah dilakukan oleh Arinta (2007) maka dalam penelitian ini akan dilakukan perhitungan banyaknya penempatan *rook* menggunakan konsep yang berbeda yaitu konsep SDR *(Systems of distinct Representative)*, sehingga akan dilakukan perbandingan hasil dari dua konsep tersebut. Pada dasarnya konsep *backtracking* dengan SDR sama yaitu menghitung banyaknya cara penempatan *rook* dalam permainan catur dengan tujuan agar tidak saling serang. Perbedaan mendasar dalam dua konsep tersebut yaitu dalam *backtracking* perhitungan dilakukan menggunakan konsep pohon, sedangkan dalam SDR perhitungan dilakukan dengan cara mencari pasangan dari dua himpunan dan setiap pasangan harus berbeda satu sama lain. Permasalahannya kemudian apakah ada jaminan bahwa hasil perhitungan meggunakan konsep *backtracking* akan sama dengan hasil

Bidang Penelitian: Kombinatorika
Tanggal masuk: 25-02-2019; Revisi: 7-03-2019
Diterima: 23-03-2019

perhitungan menggunakan konsep SDR. Oleh karena itu dalam penelitian ini, penulis melakukan penelitian mengenai perbandingan algoritma *backtracking* dengan *Systems of distinct Representative* (SDR) dalam menentukan *n-rook problem*.

II. Tinjauan Pustaka

A. Systems of Distinct Representative (SDR)

SDR merupakan salah satu algoritma dalam kombinatorika yang erat hubungannya dengan penyelesaian solusi dari permasalahan *non attacking rooks* yaitu berapa banyak jumlah benteng yang tidak saling serang yang dapat ditempatkan di tempat bebas dalam papan catur berukuran $n \times n$.

Definisi II.1. [1]
Diberikan E adalah sebuah himpunan berhingga, dan $A \subseteq E$, dengan $A = (A_1, A_2, \ldots, A_n)$. Elemen $E = (e_1, e_2, \ldots, e_n)$ yang termasuk dalam elemen A disebut system of representatives (SR), dimana:
e_1 adalah anggota A_1, e_2 anggota $A_2, \ldots, e_n$ anggota A_n
Apabila dalam SR, elemen $e_1, e_2, \ldots, e_n$ semuanya berbeda, maka $e_1, e_2, \ldots, e_n$ disebut system of distinct representatives (SDR) [4].

Pada umumnya dapat dirumuskan syarat perlu yang harus dipenuhi himpunan A untuk memiliki SDR, yaitu dimisalkan $e_1, e_2, \ldots, e_n$ adalah SDR dari $A_1, A_2, \ldots, A_n$. k adalah bilangan bulat yang memenuhi $1 \leqslant k \leqslant n$. Apabila diambil k himpunan dari $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$ dengan $1 \leqslant i_1 \leqslant i_2 \leqslant \ldots \leqslant i_k \leqslant n$ maka $e_1, e_2, \ldots, e_n$ adalah himpunan yang terdiri dari k unsur yang merupakan subset dari $A_{i_1} \bigcup A_{i_2} \bigcup \ldots \bigcup A_{i_k}$. Sehingga $A_{i_1} \bigcup A_{i_2} \bigcup \ldots \bigcup A_{i_k}$ mengandung k unsur yang berbeda [5].

B. Algoritma Backtracking

Backtracking merupakan algoritma umum untuk mencari sebagian atau seluruh solusi dari suatu persoalan khususnya persoalan yang dalam penyelesaiannya harus memenuhi beberapa kondisi tertentu [3]. Algoritma *backtracking* merupakan perbaikan dari algoritma *brute force* yaitu dengan cara menghentikan penelusuran cabang apabila sudah dipastikan tidak akan mengarah ke solusi. Oleh karena itu dalam algoritma *backtracking*, tidak perlu memeriksa semua kemungkinan solusi yang ada melainkan hanya mencari penyelesaian yang mengarah ke solusi yang dipertimbangkan, sehingga dapat meminimalisir waktu pencarian [2].

Dalam penerapan algoritma backtraking dibutuhkan properti-properti berikut [1]:

1. Solusi persoalan
   Solusi dari persoalan dinyatakan dalam sebuah vektor
   $Y = (y_1, y_2, \ldots, _n)$, $y_i \in S_i$. S_i boleh sama
   S_i merupakan himpunan nilai y_i yang merupakan kemungkinan dari solusi. Sebagai contoh diberikan himpunan $S_i = \{a, b\}$ maka kemungkinan solusinya adalah $y_i = a$ atau b.

2. Fungsi pembangkit nilai y_k
   Fungsi untuk pembangkit nilai y_k yang merupakan komponen dai vektor solusi.

3. Fungsi pembatas
   Fungsi pembatas bertujuan untuk menentukan apakah $(y_1, y_2, \ldots, y_k)$ mengarah pada vektor solusi atau tidak. Apabila mengarah pada vektor solusi maka $(y_1, y_2, \ldots, y_k)$ merupakan pembangkitan untuk nilai y_{k+1}, tetapi jika tidak mengarah pada vektor solusi maka $(y_1, y_2, \ldots, y_k)$ dibuang dan tidak dipertimbangkan dalam pencarian solusi.

III. Metode Penelitian

Data yang digunakan dalam penelitian ini merupakan data yang digunakan dalam penelitian Arinta (2007) yaitu diberikan papan

Vol. 01,No. 01 (2019),pp. xx-xx
p-ISSN 2656-6303
URL:http://journal.unirow.ac.id/index.php/mv

catur sebagai tempat meletakkan benteng yang direpresentasikan sebagai matriks berukuran 4×4 dengan ada beberapa kotak yang tidak boleh ditempati.

Tabel I
Representative PAPAN CATUR

$(1,1)$	$(2,1)$	$(3,1)$	$(4,1)$
$(1,2)$	$(2,2)$	$(3,2)$	$(4,2)$
$(1,3)$	$(2,3)$	$(3,3)$	$(4,3)$
$(1,4)$	$(2,4)$	$(3,4)$	$(4,4)$

Unsur matriks yang bernilai 0 merupakan kotak dalam papan catur yang tidak boleh ditempati, sedangkan kotak yang boleh ditempati bernilai 1.

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Tahapan propses perhitungan *n-rook problem* dijelaskan oleh gambar 1.

Gambar 1. Alur penelitian

IV. HASIL DAN PEMBAHASAN

A. *perhitungan SDR*

Tahap awal perhitungan *n-rook problem* menggunakan SDR yaitu dengan merubah data yang ada kedalam bentuk matriks.

Tabel II
PELETAKAN *rook*

	1	2	3	4
A_1		$\times$	$\times$	
A_2	$\times$			
A_3	$\times$	$\times$		$\times$
A_4			$\times$	$\times$

Unsur matriks yang bernilai 0 merupakan kotak dalam papan catur yang tidak boleh ditempati, sedangkan kotak yang boleh ditempati diberikan nilai 1.

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Berdasarkan tabel 2, terdapat $A = \{A_1, A_2, A_3, A_4\}$ yang merupakan subset dari $Y = \{1,2,3,4\}$ dimana A_i adalah himpunan yang elemennya merupakan kotak yang boleh ditempati oleh *rook* pada baris ke-i.
$A_1 = \{1,4\}$
$A_2 = \{2,3,4\}$
$A_3 = \{3\}$
$A_4 = \{1,2\}$

Sehingga dengan perhitungan SDR diperoleh dua solusi peletakan *rook* atau dua SDR yaitu:
$(SDR)_1 = \{4,2,3,1\}$
$(SDR)_2 = \{1,4,3,2\}$

B. *perhitungan Backtracking*

Berdasarkan penelitian yang dilakukan oleh Arinta (2007) yaitu diberikan papan catur berukuran 4×4 sebagai tempat peletakan *rook* yang direpresentasikan sebagai matriks (n,m)

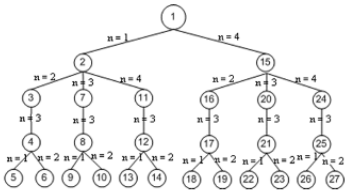
dimana n = kolom dan m = baris (sesuai gambar 2).

(1,1)	(2,1)	(3,1)	(4,1)
(1,2)	(2,2)	(3,2)	(4,2)
(1,3)	(2,3)	(3,3)	(4,3)
(1,4)	(2,4)	(3,4)	(4,4)

Gambar 2. *Representative* papan catur

Penyelesaian permasalahan menggunakan Backtracking dibantu oleh pohon ruang status sebagaimana gambar 3.

Setiap *rook* akan diletakkan dari baris ke-1



Gambar 3. *pohon ruang status*
sumber: makalah IF2251 strategi algoritmatik tahun 2007

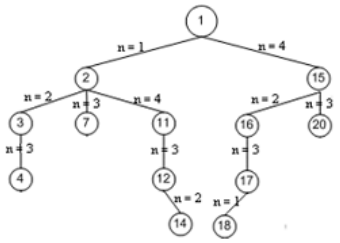
sampai baris ke-4. Pada level ke-1 hanya terdapat dua kemungkinan n yaitu $n = 1$ atau $n = 4$, dikarenakan terdapat beberapa kotak yang tidak boleh ditempati oleh *rook*. Hal tersebut berlaku untuk baris ke-2 sampai baris ke-4.

$S_1 = \{1,4\}$
$S_2 = \{2,3,4\}$
$S_3 = \{3\}$
$S_4 = \{1,2\}$

Pohon dinamis yang terbentuk dalam pencarian solusi pada persoalan tersebut digambarkan oleh gambar 4.

sehingga diperoleh dua himpunan solusi yaitu Y_1 dan Y_2.

$$Y_1 = \{1,4,3,2\}$$
$$Y_2 = \{4,2,3,1\}$$



Gambar 4. *pohon dinamis pencarian permasalahan*
sumber: makalah IF2251 strategi algoritmatik tahun 2007

C. analisis perbandingan SDR dengan Backtracking

Hasil perhitungan menggunakan *Backtracking* maupun SDR memberikan solusi yang sama yaitu terdapat dua himpunan solusi yang elemen-elemennya juga sama yaitu:

Hasil SDR
$(SDR)_1 = \{4,2,3,1\}$
$(SDR)_2 = \{1,4,3,2\}$

Hasil *Backtracking*
$Y_1 = \{1,4,3,2\}$
$Y_2 = \{4,2,3,1\}$

Tetapi karena dalam perhitungan *Backtracking* diperlukan pohon ruang status dalam proses pencarian solusi maka perhitungan *Backtracking* membutuhkan waktu yang lama daripada menggunakan SDR.

V. KESIMPULAN

Berdasarkan hasil dan pembahasan, maka dapat disimpulkan bahwa perhitungan penempatan *rook* dalam permainan catur menggunakan *Backtracking* maupun SDR memberikan hasil yang sama, tetapi perhitungan SDR lebih efisien daripada *Backtracking*. Hal tersebut dikarenakan perhitungan menggunakan *Backtracking* membutuhkan waktu yang lebih lama daripada SDR.

REFERENSI

[1] A. P. Auza, Penggunaan Algoritma Backtracking dalam Pencarian Koefisien *Rook Polynomial*, dalam Makalah IF2251 Strategi Algoritmik, Bandung (2007).

[2] H. F. Hidayat, Deskripsi Langkah Benteng pada Papan Catur n x n dengan Menggunakan Algoritma Runut Balik, Skripsi Universitas Islam Negeri Maulana Malik Ibrahim, Malang (2011).

[3] R. A. Genadi, Penyelesaian n-Queens Problem dengan Metode *Recursive Backtracking*, dalam Makalah IF2120 Matematika Diskrit, Bandung (2018).

[4] R. A. Brualdi, Introductory Combinatorics, Hongkong: Pearson Education (2009).

[5] S. Slamet dan H. Makaliwe, Matematika Kombinatorika, Jakarta: PT Elex Media Komputindo, Gramedia (1991).