Algoritma Branch & Bound

(Bagian 1)

Bahan Kuliah IF2211 Strategi Algoritma

Oleh: Rinaldi Munir, Nur Ulfa Maulidevi, Masayu Leylia Khodra

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika ITB
2021

Overview

- Pembentukan pohon ruang status (*state space tree*) dinamis untuk mencari solusi persoalan
  - BFS
  - DFS
  - DLS
  - IDS
  - Backtracking

Overview

- BFS: solusi dengan minimum step, *exponential space*
- DFS: lebih efisien (1 solusi) , lintasannya dapat terlalu panjang (pohon ruang status tidak berhingga kedalamannya)
- DLS: variasi DFS, solusi bisa tidak ketemu (*depth-limited*)
- IDS: sekuens DLS (*depth* ++)
- *Backtracking*: basis DFS, *expand* simpul jika arahnya benar, fungsi pembatas

Algoritma *Branch & Bound* (B&B)

- Digunakan untuk persoalan optimisasi → meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan (*constraints*) persoalan
- B&B = BFS + *least cost search*
- BFS murni: Simpul berikutnya yang akan diekspansi berdasarkan urutan pembangkitannya (FIFO)
- B&B:
  - Setiap simpul diberi sebuah nilai *cost*:
    `ĉ(i)` = nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i.
  - Simpul berikutnya yang akan di-expand **tidak lagi** berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki *cost* yang paling kecil (*least cost search*) – pada kasus minimasi.

B&B vs Backtracking

- Persamaan:
  - Pencarian solusi dengan pembentukan pohon ruang status
  - ‘Membunuh’ simpul yang tidak ‘mengarah’ ke solusi
- Perbedaan:
  - ‘nature’ persoalan yang bisa diselesaikan:
    - *Backtracking*: Tak ada batasan (optimisasi/non-optimasi), umumnya untuk persoalan non-optimisasi
    - B&B:
      - Persoalan optimisasi
      - Untuk setiap simpul pada pohon ruang-status, diperlukan suatu cara penentuan batas (*bound*) nilai terbaik fungsi objektif pada setiap solusi yang mungkin, dengan menambahkan komponen pada solusi sementara yang direpresentasikan oleh simpul
      - Nilai dari solusi terbaik sejauh ini
  - Pembangkitan simpul: …

B&B vs Backtracking (2)

- Perbedaan:
  - Pembangkitan simpul:
    - Backtracking: umumnya DFS
    - B&B : beberapa ‘aturan’ tertentu → paling umum ‘best-first rule’

“Fungsi Pembatas”

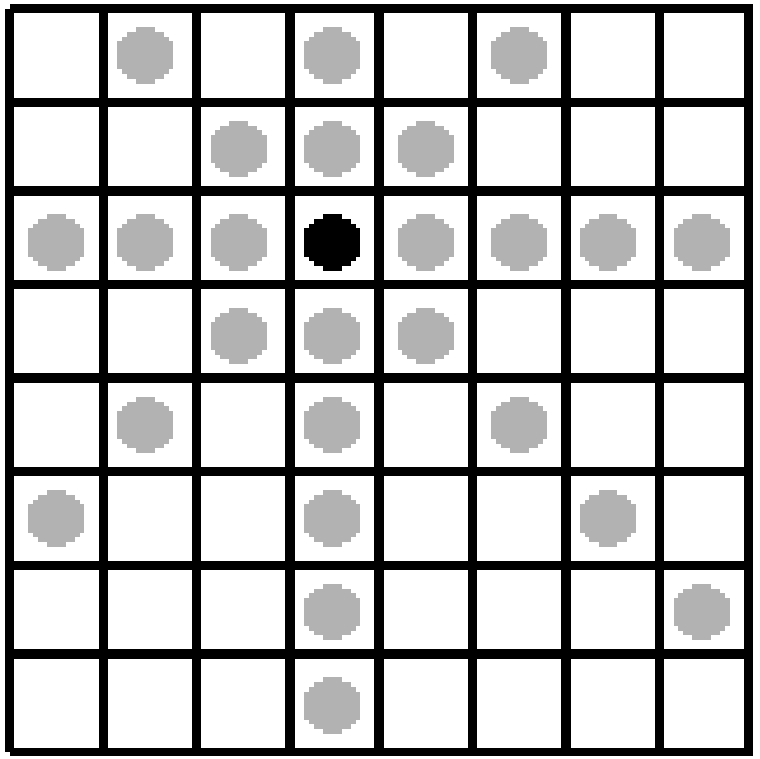
- Algoritma B&B juga menerapkan “pemangkasan” pada jalur yang dianggap tidak lagi mengarah pada solusi.

- Kriteria pemangkasan secara umum:
  - Nilai simpul tidak lebih baik dari nilai terbaik sejauh ini (*the best solution so far*)
  - Simpul tidak merepresentasikan solusi yang ‘feasible’ karena ada batasan yang dilanggar
  - Solusi pada simpul tersebut hanya terdiri atas satu titik → tidak ada pilihan lain; bandingkan nilai fungsi obyektif dengan solusi terbaik saat ini, yang terbaik yang diambil

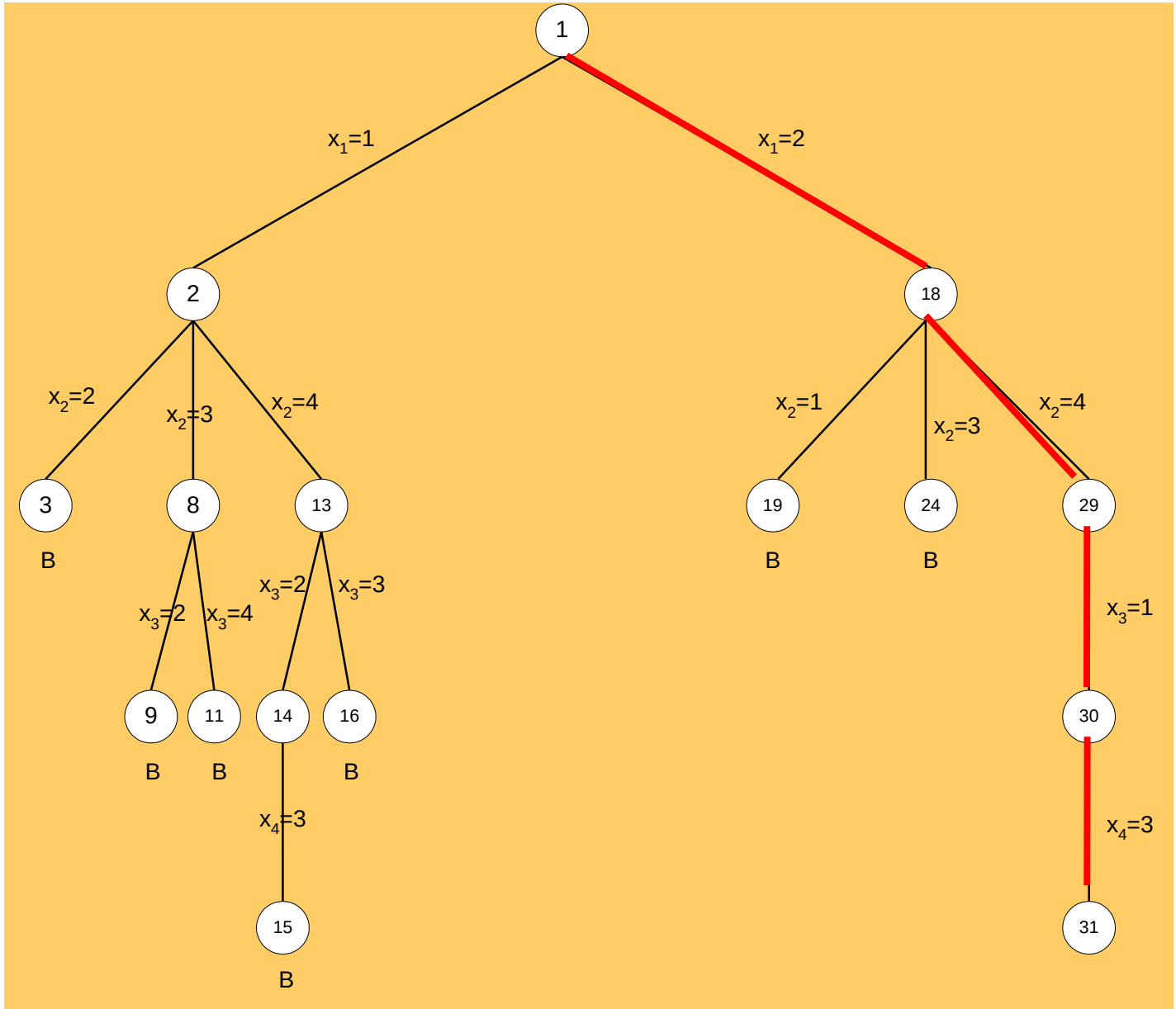
Persoalan N-Ratu (*The N-Queens Problem*)

- Tinjau kembali persoalan N-Ratu

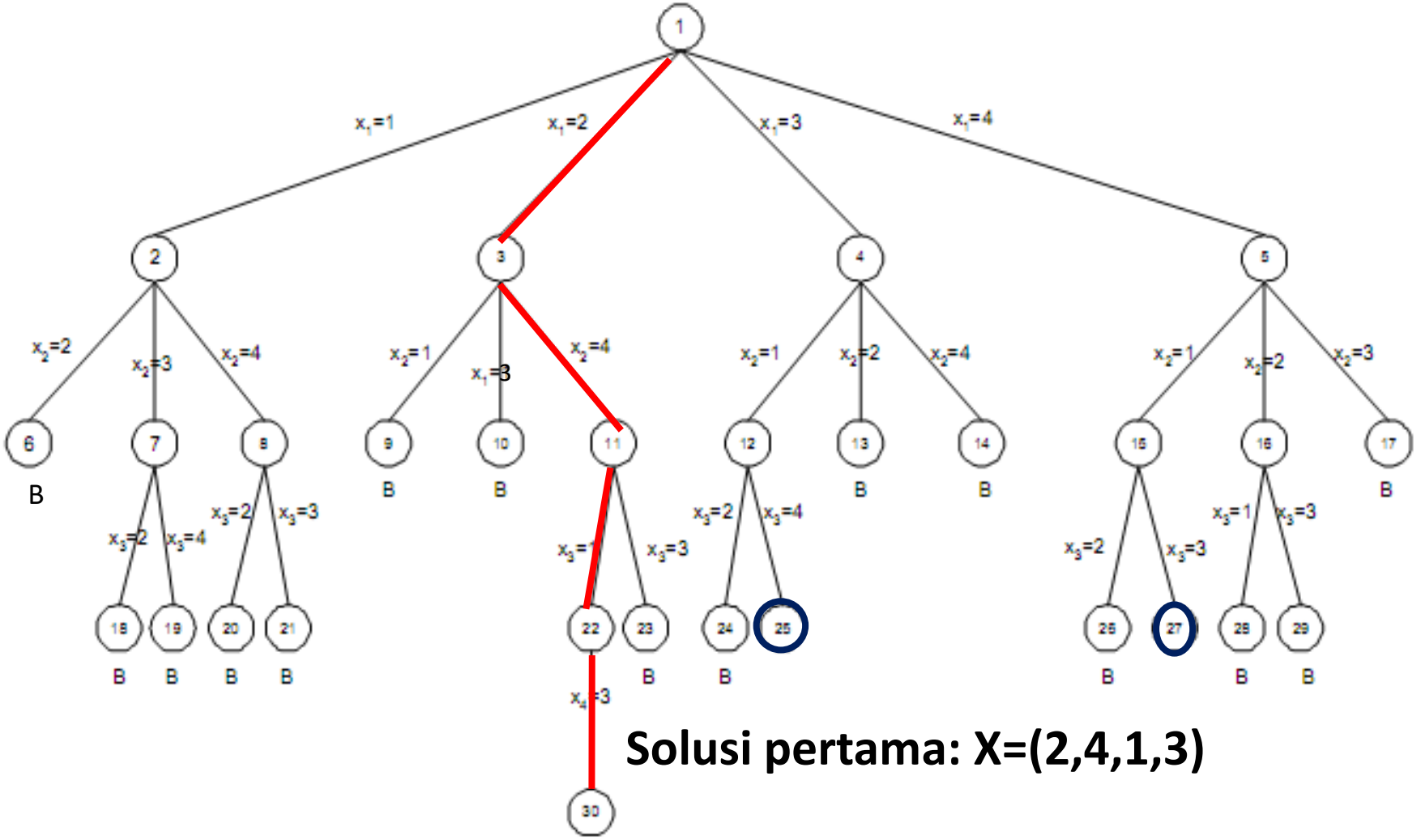
- Diberikan sebuah papan permainan yang berukuran $N \times N$ dan N buah ratu. Bagaimanakah menempatkan N buah ratu (Q) itu pada petak-petak papan permainan sedemikian sehingga tidak ada dua ratu atau lebih yang terletak pada satu baris yang sama, atau pada satu kolom yang sama, atau pada satu diagonal yang sama.



Pohon ruang status persoalan 4-Ratu: Algoritma *Backtracking*



Solusi 4-Ratu dengan BFS-dengan Pembatasan (*FIFO-Branch and Bound*)

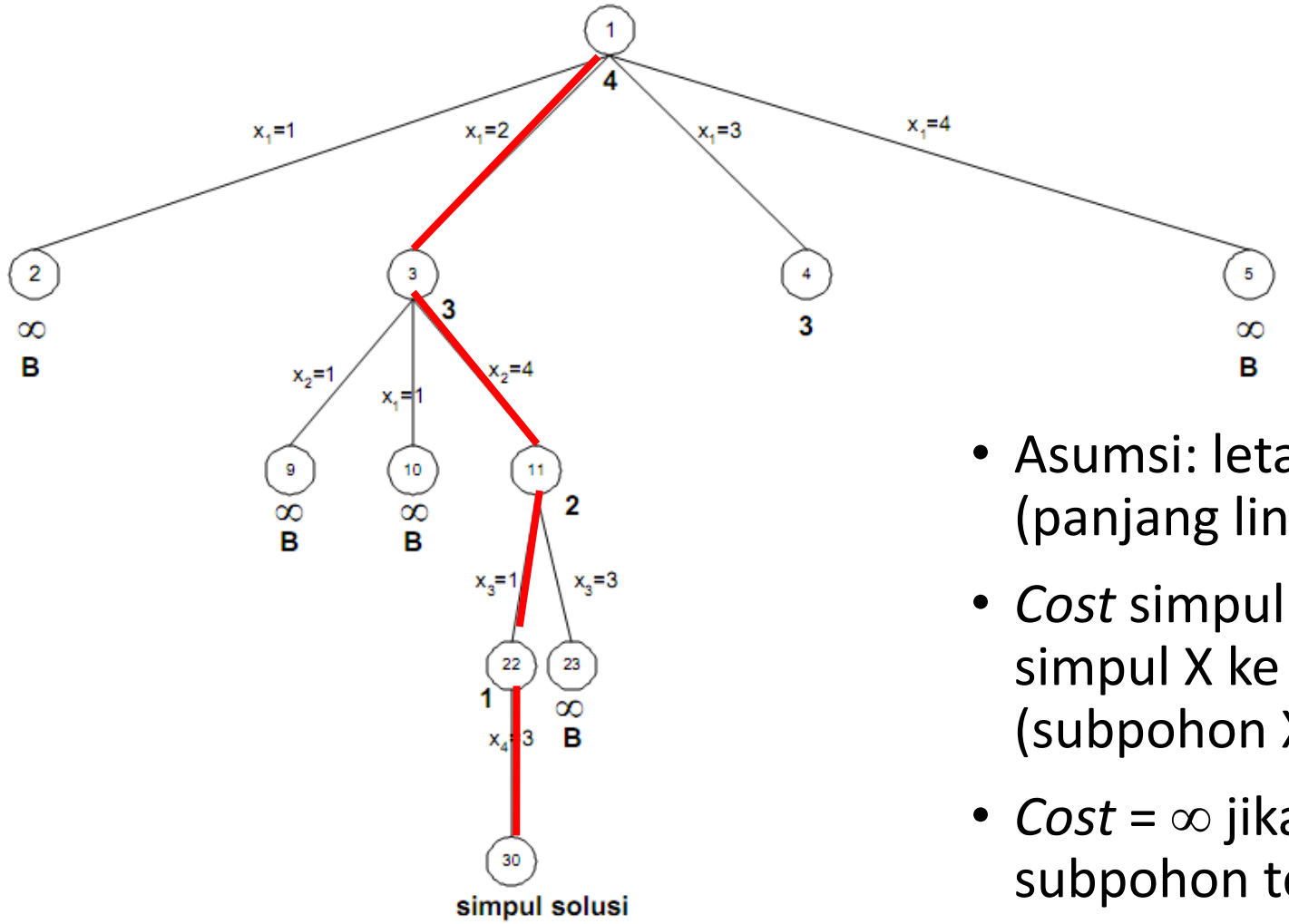


Strategi Pencarian Solusi 4-Ratu menggunakan Algoritma *Branch and Bound*

- Simpul hidup yang menjadi simpul-E(xpand) ialah simpul yang mempunyai nilai *cost* terkecil (*least cost search*)→ salah satu jenis aturan

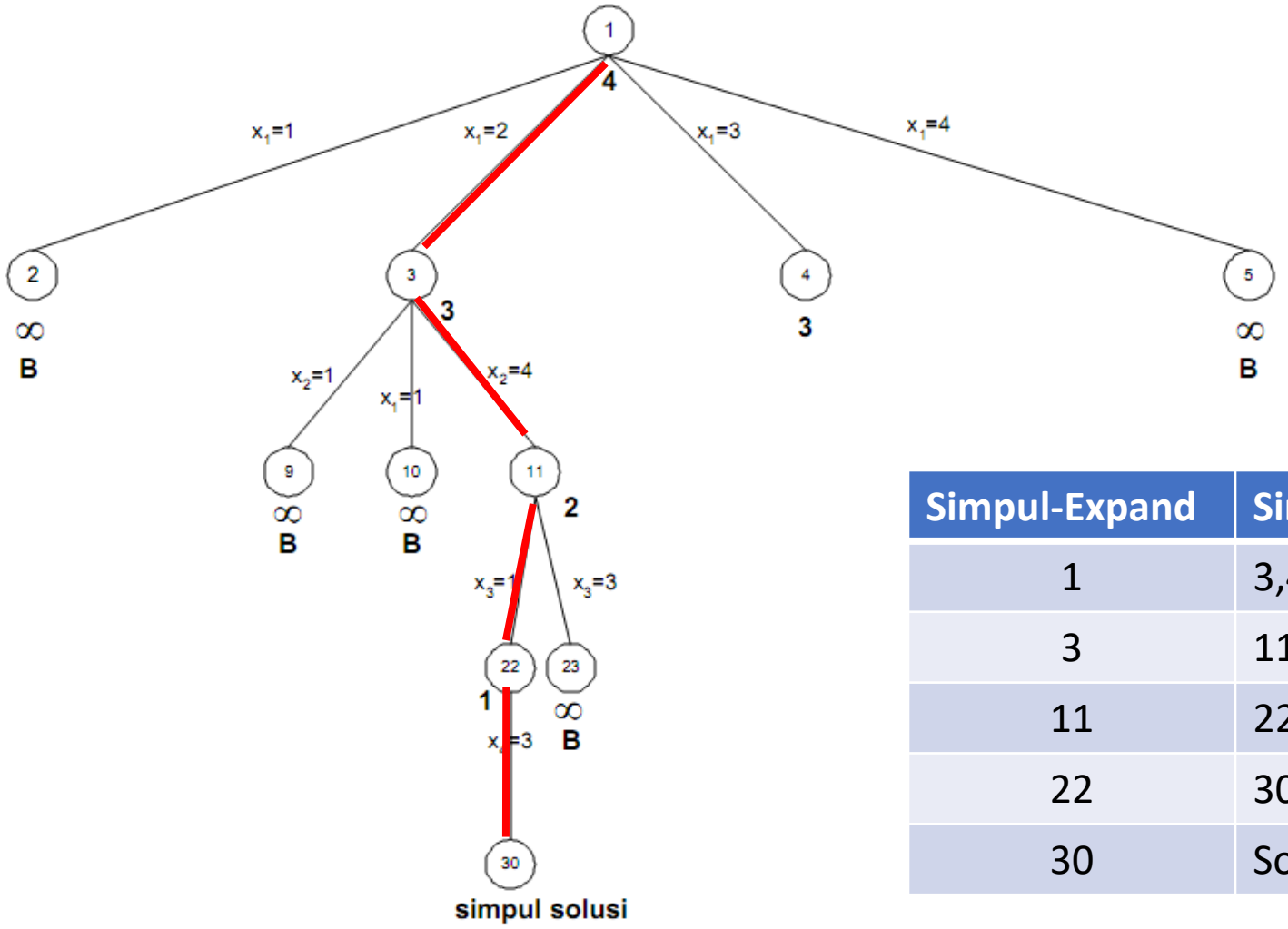
- Untuk setiap simpul X, nilai cost ini dapat berupa:
  1. jumlah simpul dalam upapohon X yang perlu dibangkitkan sebelum simpul solusi ditemukan
  2. panjang lintasan dari simpul X ke simpul solusi terdekat (dalam upapohon X ybs) → misal ini yang dipilih untuk persoalan 4-ratu

Solusi 4-Ratu dengan Algoritma *Branch and Bound*



- Asumsi: letak simpul solusi diketahui (panjang lintasan solusi = 4)
- *Cost* simpul X: panjang lintasan dari simpul X ke simpul solusi terdekat (subpohon X).
- *Cost* = ∞ jika tidak ada simpul solusi di subpohon tersebut.

Pembentukan Pohon Ruang Status 4-Ratu dengan Algoritma *Branch & Bound*



Simpul-Expand	Simpul Hidup
1	3,4,2,5
3	11,4,2,5,9,10
11	22,4,2,5,9,10,23
22	30,4,2,5,9,10,23
30	Solusi ketemu

Priority queue

Cost dari Simpul Hidup

- Pada umumnya, untuk kebanyakan persoalan, letak simpul solusi tidak diketahui.
  - Persoalan N-Ratu: persoalan yg ideal (letak simpul solusi diketahui)

- Letak simpul solusi diketahui?
  - *knapsack problem*,
  - *graph colouring*,
  - permainan 8-*puzzle*,
  - TSP

- Oleh karena itu, nilai *cost* (atau *bound*) simpul *i* merupakan estimasi ongkos termurah lintasan dari simpul *i* ke simpul solusi (yang tidak diketahui letaknya), dilambangkan dengan $\hat{c}(i)$. Nilai $\hat{c}(i)$ dihitung secara heuristik.
- Dengan kata lain, $\hat{c}(i)$ menyatakan **batas bawah** (*lower bound*) dari ongkos pencarian solusi dari status *i*.

Algoritma *Branch & Bound*

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (*goal node*), maka solusi telah ditemukan. Jika hanya satu solusi yang diinginkan, maka stop.
2. Jika Q kosong, Stop.
3. Jika Q tidak kosong, pilih dari antrian Q simpul *i* yang mempunyai nilai 'cost' $\hat{c}(i)$ paling kecil. Jika terdapat beberapa simpul *i* yang memenuhi, pilih satu secara sembarang.
4. Jika simpul *i* adalah simpul solusi, berarti solusi sudah ditemukan. Jika satu solusi yang diinginkan, maka stop.

   Pada persoalan optimasi dengan pendekatan *least cost search*, periksa *cost* semua simpul hidup. Jika *cost* nya lebih besar dari *cost* simpul solusi, maka matikan simpul tersebut.
5. Jika simpul *i* bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika *i* tidak mempunyai anak, kembali ke langkah 2.
6. Untuk setiap anak *j* dari simpul *i*, hitung $\hat{c}(j)$, dan masukkan semua anak-anak tersebut ke dalam Q.
7. Kembali ke langkah 2.

Permainan 15-Puzzle

1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

(a) Susunan awal

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(b) Susunan akhir

- State berdasarkan ubin kosong (*blank*)
- Aksi: up, down, left, right

Reachable Goal?

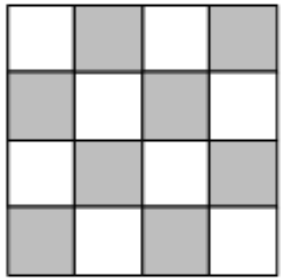
- Terdapat 16! ≈ 20,9 x 10^{12} susunan ubin yang berbeda, dan hanya setengah yang dapat dicapai dari state awal sembarang.
- Teorema : Status tujuan hanya dapat dicapai dari status awal jika $\sum_{i=1}^{16} KURANG(i) + X$ bernilai genap.
- X=1 jika sel kosong pada posisi awal ada pada sel yg diarsir

1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

State awal

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

State akhir



http://www.cs.umsl.edu/~sanjiv/classes/cs5130/lectures/bb.pdf

Reachable Goal : KURANG(i)

i	Kurang (i)
1	0
2	0
3	1
4	1
5	0
6	0
7	1
8	0
9	0
10	0
11	3
12	6
13	0
14	4
15	11
16	10

- KURANG(*i*) = banyaknya ubin bernomor *j* sedemikian sehingga *j* < *i* dan POSISI(*j*) > POSISI(*i*). POSISI(*i*) = posisi ubin bernomor *i* pada susunan yang diperiksa.
- KURANG (4) = 1 : terdapat 1 ubin (2)
- Kesimpulan: status tujuan tidak dapat dicapai.

1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

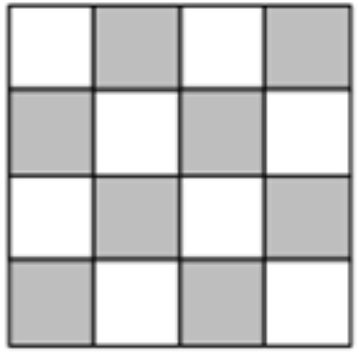
$$\sum_{i=1}^{16} Kurang\,(i) + X = 37 + 0 = 37$$

Reachable Goal ?

1	2	3	4
5	6		8
9	10	7	11
13	14	15	12

?

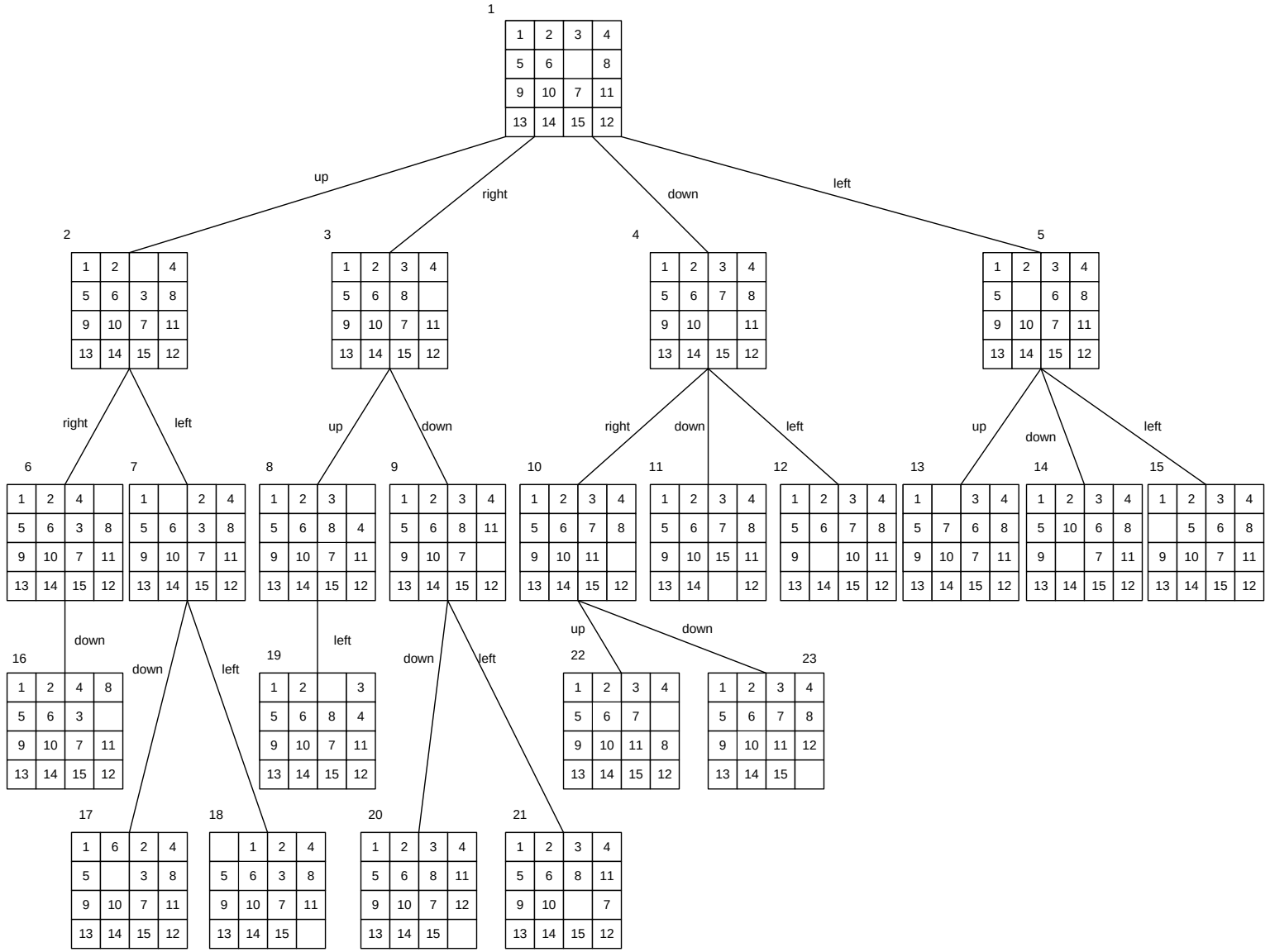
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	



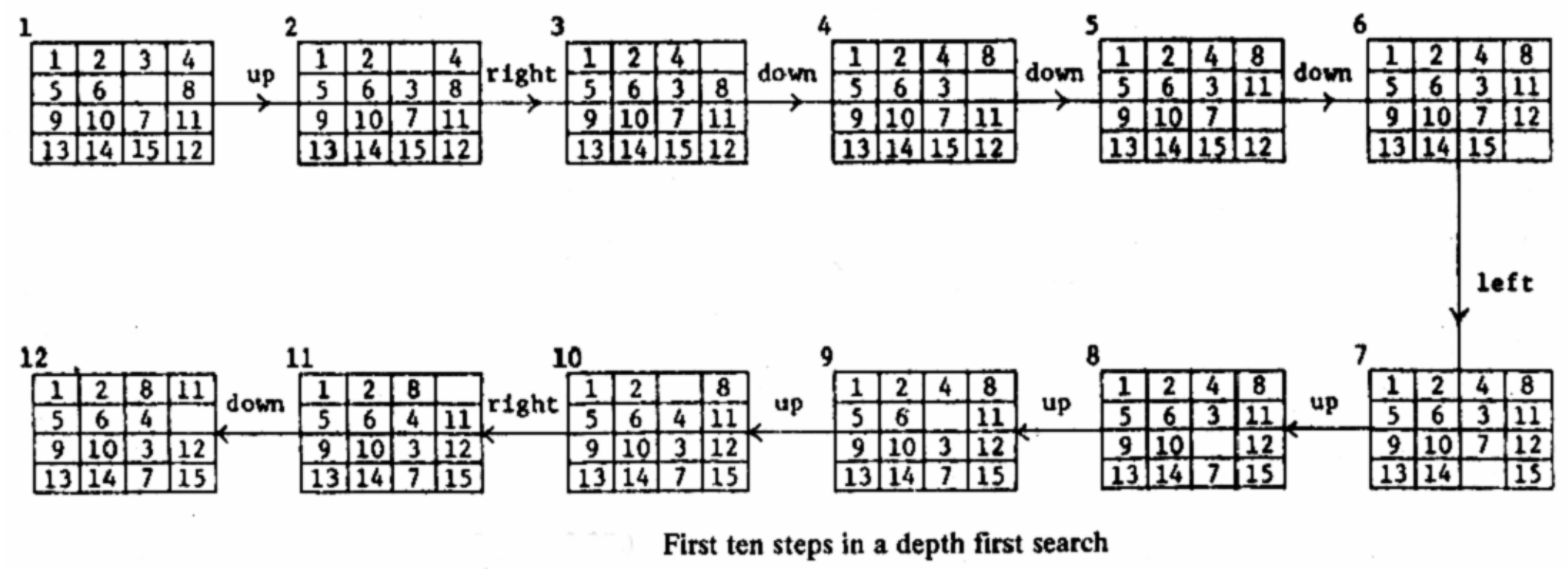
$$\sum_{i=1}^{16} Kurang(i) + X = 15 + 1 = 16$$

i	Kurang (i)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	1
10	1
11	0
12	0
13	1
14	1
15	1
16	9

Sebagian pohon ruang status untuk 15-Puzzle secara BFS



Pohon Pencarian untuk *DFS*



First ten steps in a depth first search

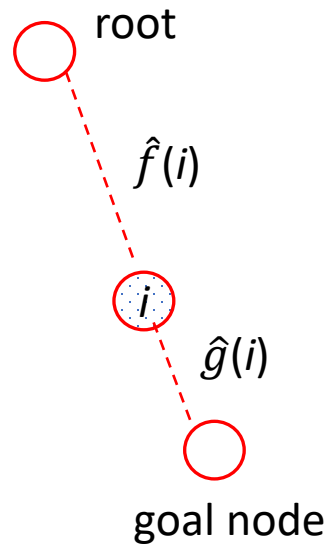
http://chern.ie.nthu.edu.tw/alg2003/alg-2009-chap-7.pdf

Cost dari Simpul Hidup (2)

- Pada umumnya, untuk kebanyakan persoalan, letak simpul solusi (tujuan) tidak diketahui.
- *Cost* setiap simpul umumnya berupa taksiran.

$$\hat{c}(i) = \hat{f}(i) + \hat{g}(i)$$

$\hat{c}(i)$ = ongkos untuk simpul i
$\hat{f}(i)$ = ongkos mencapai simpul i dari akar
$\hat{g}(i)$ = ongkos mencapai simpul tujuan dari simpul i



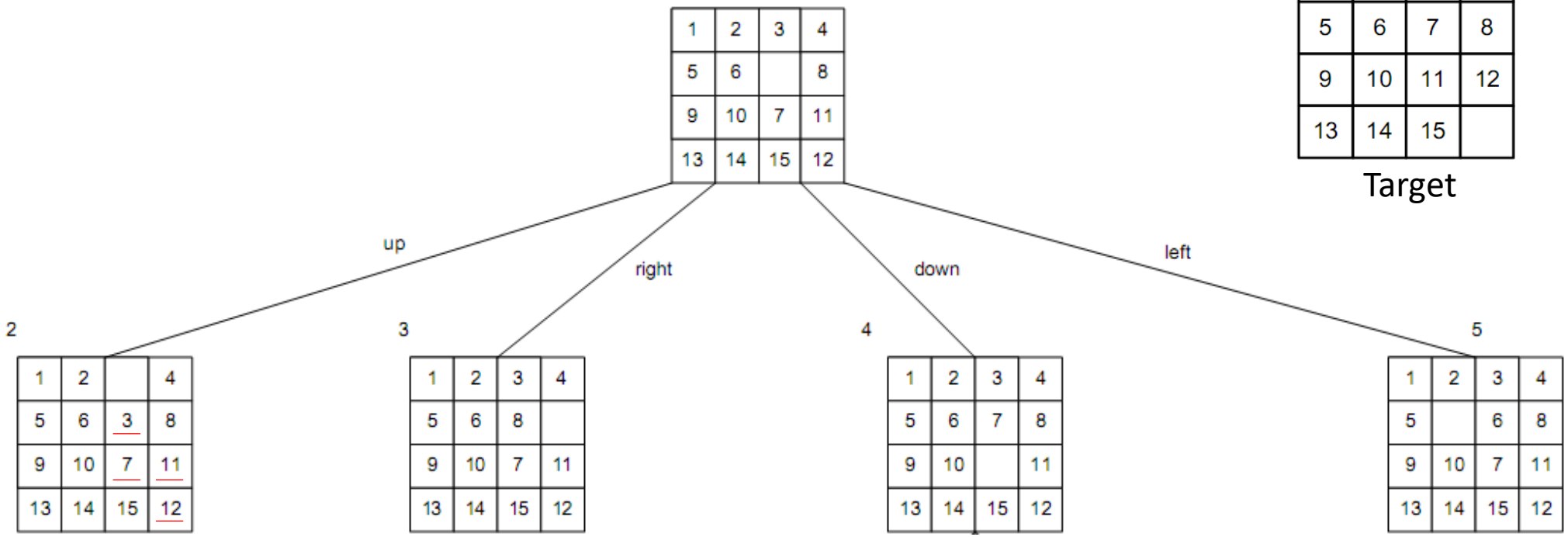
- Cost simpul *P* pada 15-puzzle: $\hat{c}(P) = f(P) + \hat{g}(P)$

$f(P)$ = adalah panjang lintasan dari simpul akar ke P

$\hat{g}(P)$ = taksiran panjang lintasan terpendek dari P ke simpul solusi pada upapohon yang akarnya P.

Cost dari Simpul Hidup 15-Puzzle

$$\hat{g}(P) = \text{jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir}$$



Target

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

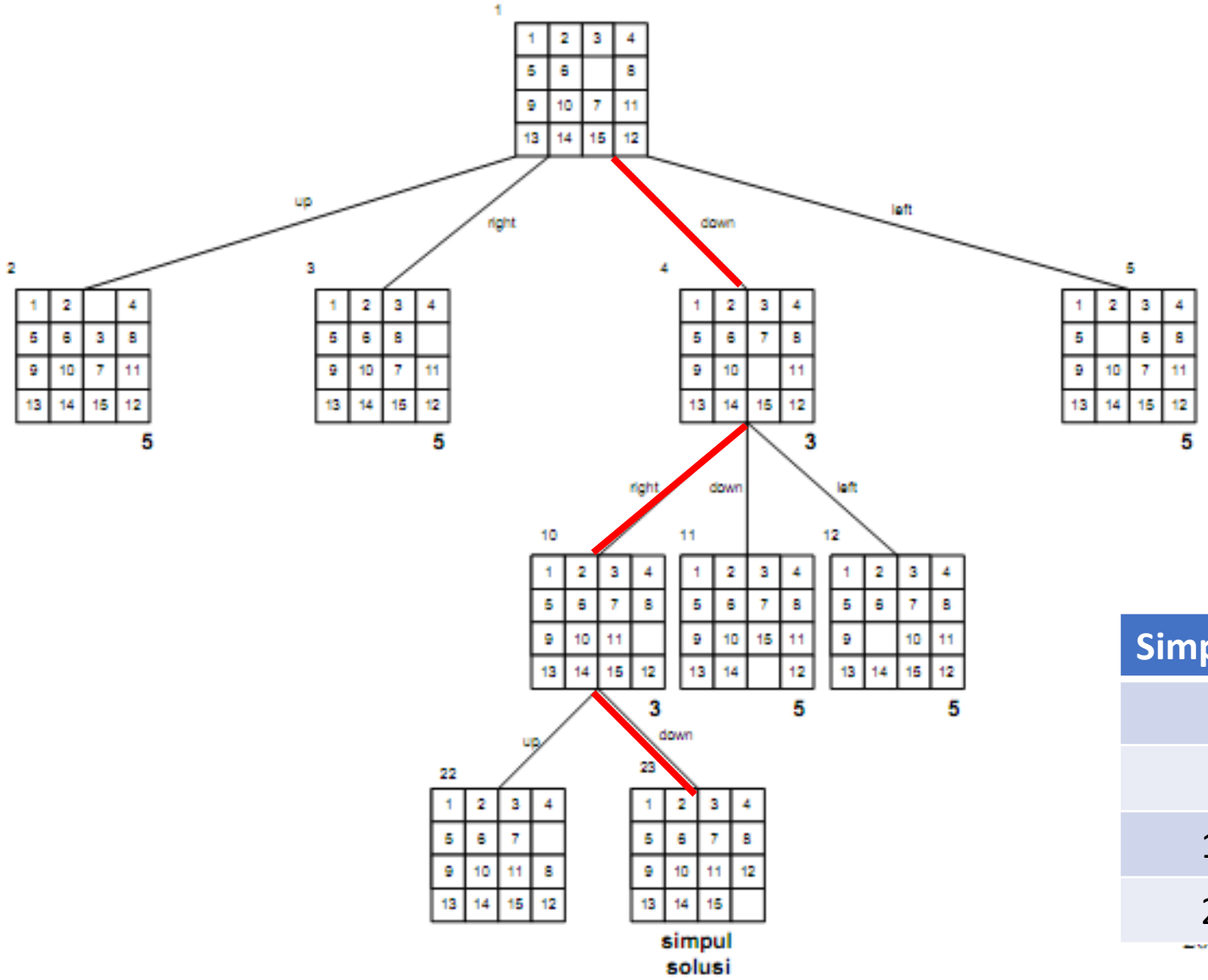
C(2)=f(2)+g(2)
C(2)=1+4=5

C(3)=f(3)+g(3)
C(3)=1+4=5

C(4)=f(4)+g(4)
C(4)=1+2=3

C(5)=f(5)+g(5)
C(5)=1+4=5

Pembentukan Pohon Ruang Status 15-Puzzle dengan Algoritma *Branch & Bound*



Simpul-E	Simpul Hidup
1	4,2,3,5
4	10,2,3,5,11,12
10	23,2,3,5,11,12,22
23	Solusi ketemu

BERSAMBUNG