TEKNIK INFORMATIKA
FAKULTAS TEKNIK UNIVERSITAS MATARAM

Heuristic Search

Ramaditia D – rama@unram.ac.id

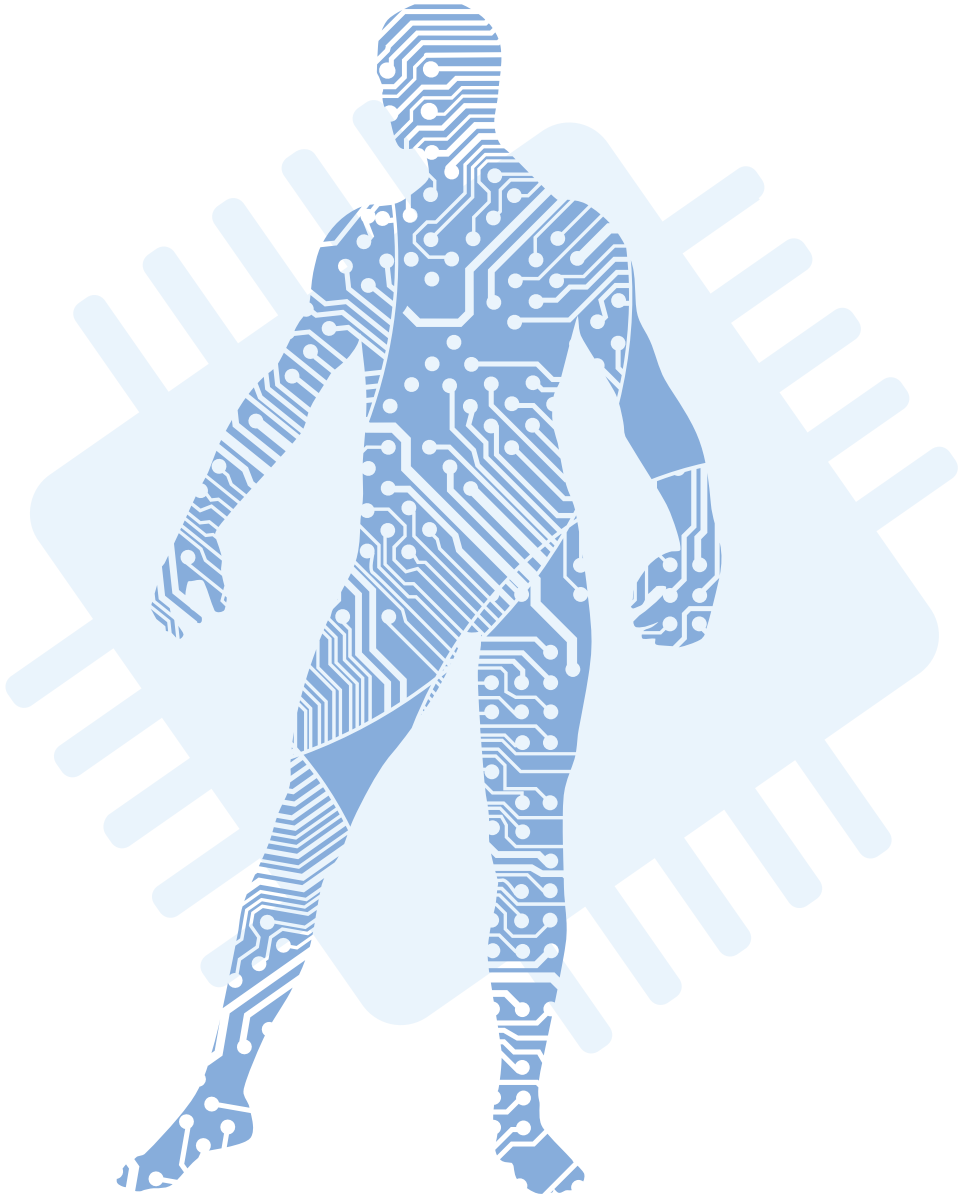
v1.2 update 20 September 2020

Outline

01 **Pendekatan Heuristik**

02 **Metode Best First Search**

03 **Hill Climbing**



Kelemahan Blind Search

- Kelemahan Blind Search :
  - ✓Waktu akses lama
  - ✓Memori yang dibutuhkan besar
  - ✓Ruang masalah besar – tidak cocok – karena keterbasan kecepatan komputer dan memori

- Solusi ➔ Pencarian heuristik

Blind VS Heurisctic Search

Blind Search

Tidak ada alternatif yang dibuang, hanya masalah prioritas

Ada beberapa alternatif yang dibuang.

Heuristik

HEURISTIC / INFORMED SEARCH

- Kata *Heuristic* berasal dari sebuah kata kerja bahasa Yunani, *heuriskein*, yang berarti ‘mencari’ atau menemukan.

- Dalam dunia pemrograman, sebagian orang menggunakan kata heuristik sebagai lawan kata algoritmik, dimana kata heuristik ini diartikan sebagai **suatu proses yang mungkin dapat menyelesaikan suatu masalah, tetapi tidak ada jaminan bahwa solusi yang dicari selalu dapat ditemukan.**

- Heuristik dapat diartikan juga sebagai suatu kaidah yang merupakan metoda/prosedur yang didasarkan kepada pengalaman dan praktek, syarat, trik atau bantuan lainnya yang **membantu mempersempit dan memfokuskan** proses pelacakan kepada suatu tujuan tertentu.

Pendekatan Heuristik

- Teknik pencarian heuristik (*heuristic searching*) merupakan suatu strategi untuk melakukan proses pencarian ruang keadaan (*state space*) suatu problema secara **selektif**, yang memandu proses pencarian di sepanjang jalur yang memiliki kemungkinan sukses paling besar, dan mengesampingkan usaha yang "dianggap bodoh" dan memboroskan waktu.

- Heuristik adalah sebuah teknik yang mengembangkan efisiensi dalam proses pencarian, namun dengan kemungkinan mengorbankan kelengkapan (*completeness*).

- Di dalam mempelajari metode-metode pencarian ini, kata heuristik diartikan sebagai suatu fungsi yang memberikan suatu nilai berupa biaya perkiraan (estimasi) dari suatu solusi.

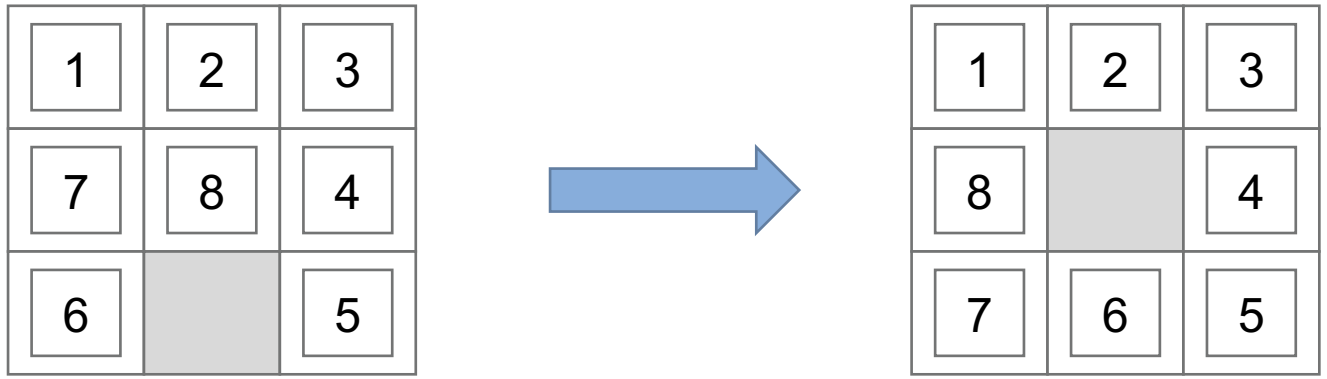
Kecerdasan Buatan menggunakan heuristik dalam 2 situasi dasar:

1. Permasalahan yang memiliki solusi pasti, tetapi biaya komputasinya untuk mendapatkan solusi tersebut sangat tinggi.

2. Permasalahan yang mungkin tidak mempunyai solusi yang pasti disebabkan oleh ambiguitas (keraguan/ketidakpastian). Contohnya: diagnosa kedokteran.

Heuristik hanyalah sebuah cara menerka langkah berikutnya yang harus diambil dalam memecahkan suatu masalah berdasarkan informasi yang tersedia.

Contoh Permasalahan Pencarian Heuristik : Kasus 8-Puzzle



Initial State **Goal State**

Heuristic function h = ?

Beberapa Pendekatan Kasus 8-Puzzle

- Ada 4 action / aturan :
  - Ubin kosong digeser ke kiri
  - Ubin kosong digeser ke kanan
  - Ubin kosong digeser ke bawah
  - Ubin kosong digeser ke atas

Fungsi heuristik yang mungkin:

- $h_1(n)$ = jumlah ubin yang menempati posisi yang **benar**
- $h_2(n)$ = jumlah ubin yang menempati posisi yang **salah**
- $h_3(n)$ = jumlah gerakan untuk mencapai tujuan

Pendekatan Heuristik 1: Kasus 8-Puzzle

Misalkan fungsi heuristik untuk kasus ini didefinisikan sebagai: banyaknya ubin yang menempati posisi yang **benar**, dimana jumlah yang lebih tinggi adalah yang lebih diharapkan (lebih baik).

Initial State

1	2	3
7	8	4
6		5

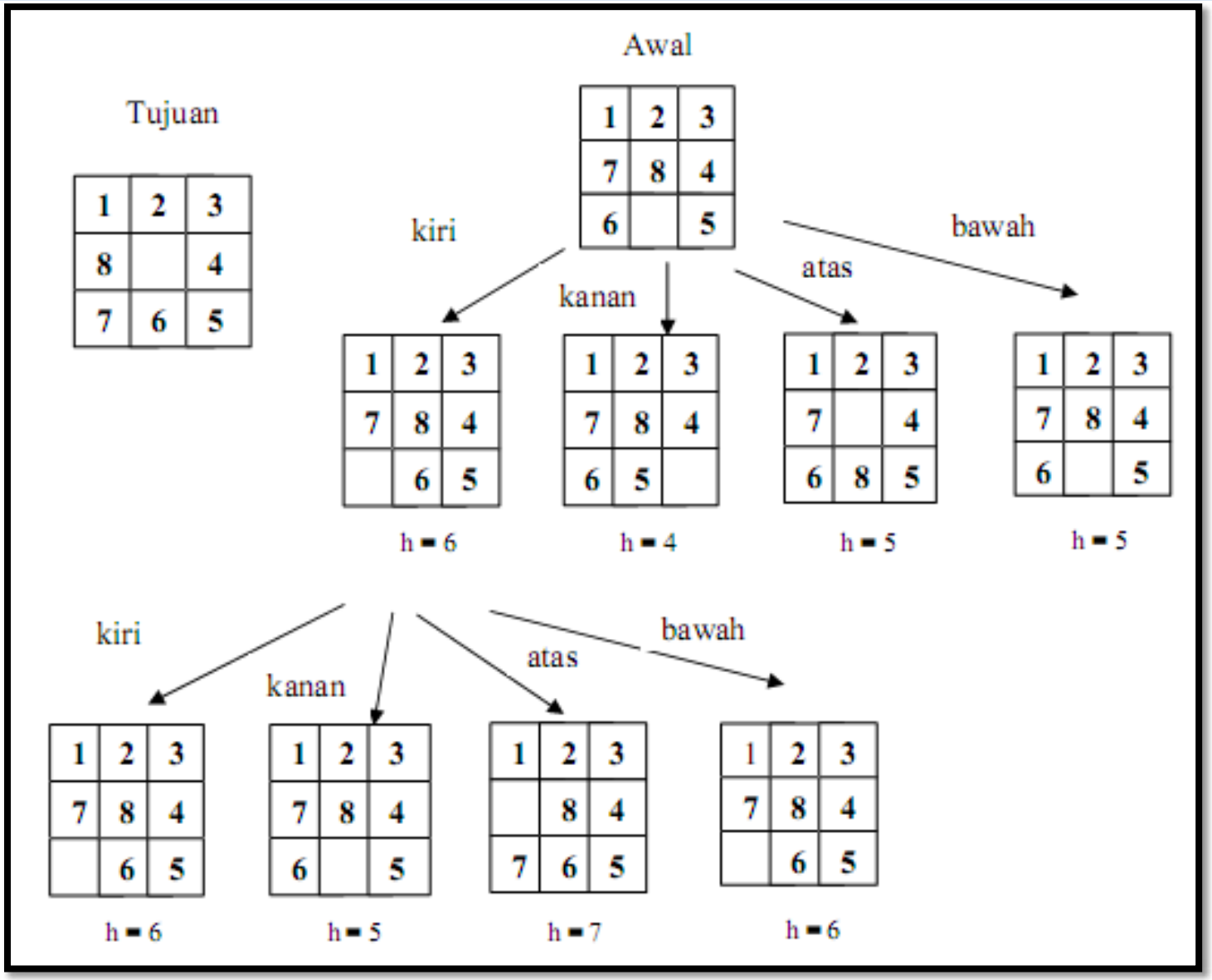
h = 5

Goal State

1	2	3
8		4
7	6	5

h = 8

Pengerjaan Pendekatan Heuristik 1



Pendekatan Heuristik 2: Kasus 8-Puzzle

Misalkan fungsi heuristik untuk kasus ini didefinisikan sebagai: banyaknya ubin yang menempati posisi yang **salah**, dimana jumlah yang lebih rendah adalah yang lebih diharapkan (lebih baik).

Initial State

1	2	3
7	8	4
6		5

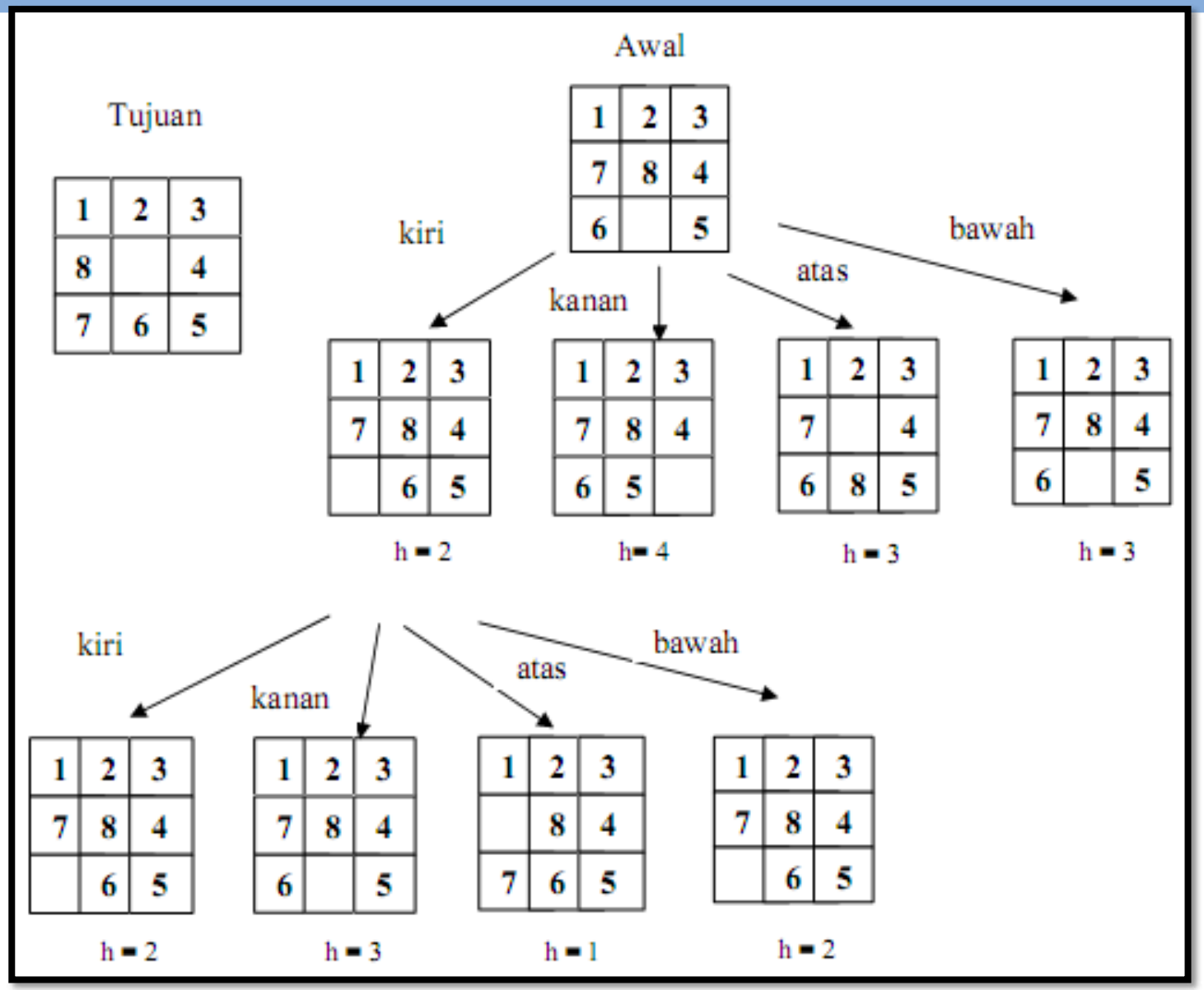
h = 3

Goal State

1	2	3
8		4
7	6	5

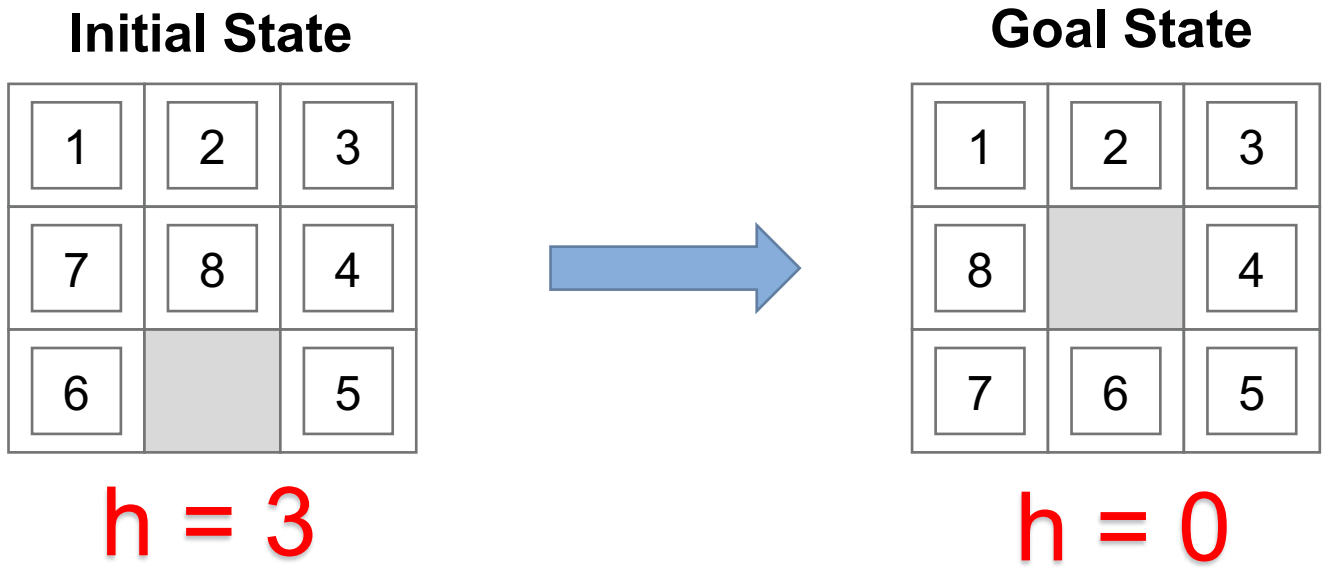
h = 0

Pengerjaan Pendekatan Heuristik 2

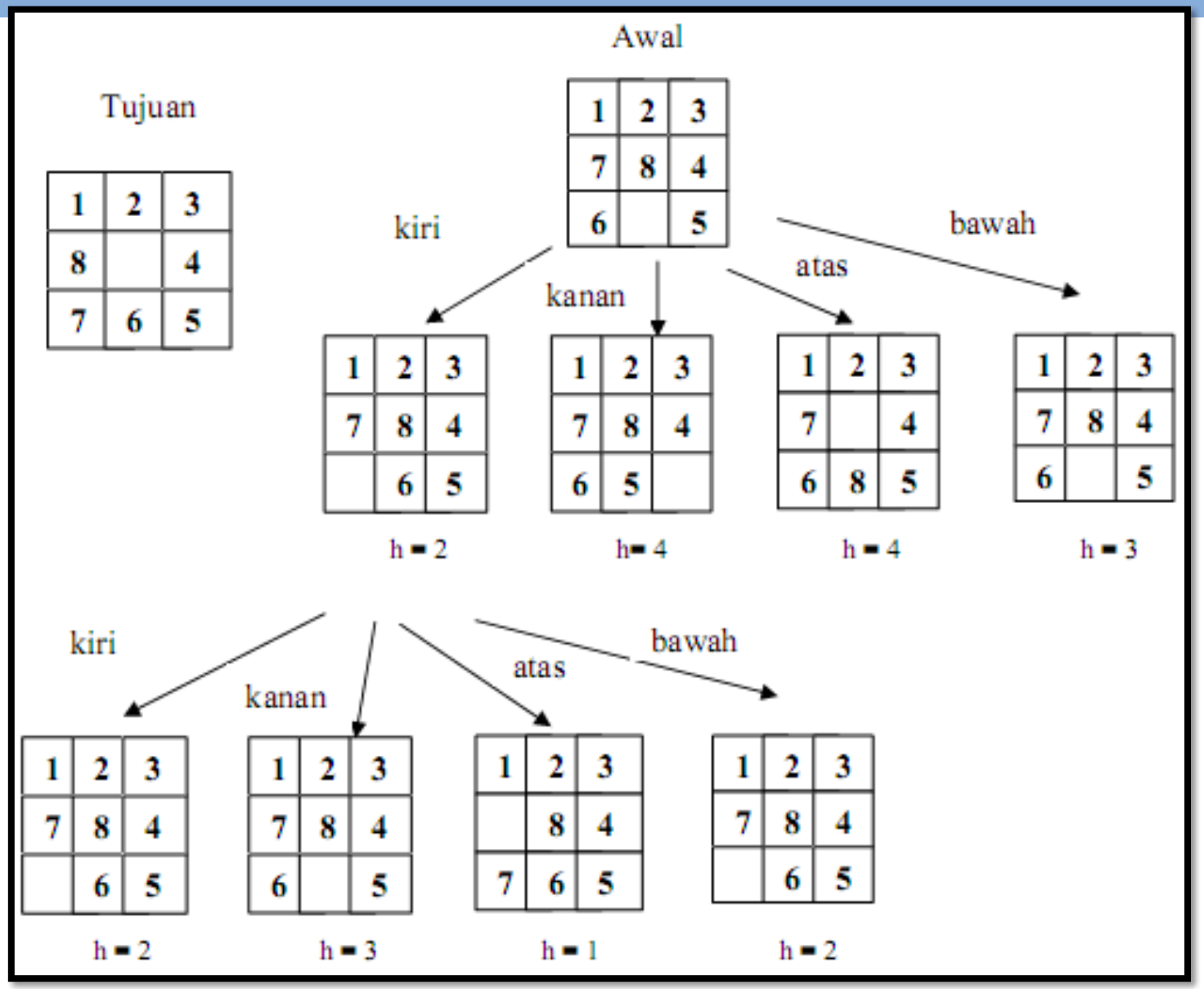


Pendekatan Heuristik 3: Kasus 8-Puzzle

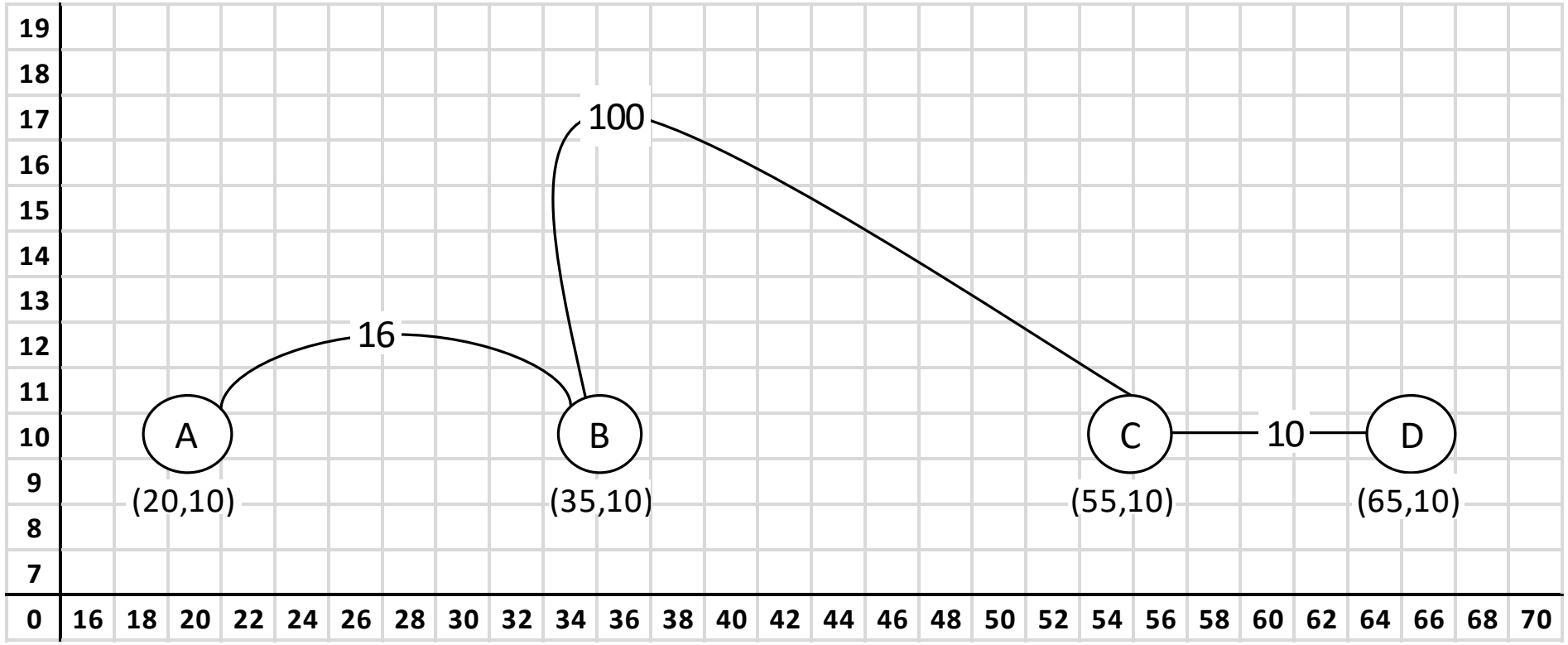
Misalkan fungsi heuristik untuk kasus ini didefinisikan sebagai: total gerakan yang diperlukan untuk mencapai tujuan, dimana jumlah yang lebih kecil adalah yang lebih diharapkan (lebih baik).



Pengerjaan Pendekatan Heuristik 3



Contoh Permasalahan Pencarian Heuristik : Kasus Jalur Terpendek



Heuristic function h = ?

Contoh Permasalahan Pencarian Heuristik : Kasus Jalur Terpendek

Fungsi heuristik yang mungkin:

- Jarak garis lurus antar kota.

Dengan fungsi heuristik h adalah Euclidean distance:

$$d_{ab} = \sqrt{(y_b - y_a)^2 + (x_b - x_a)^2}$$

Fungsi Heuristik

- Fungsi heuristik diterima jika estimasi biaya yang dihasilkan tidak melebihi biaya sebenarnya.
- Fungsi heuristik yang terlalu tinggi dapat membuat proses pencarian hilang atau mencapai hasil yang tidak optimal.
- Fungsi heuristik yang baik adalah fungsi yang memberikan perkiraan biaya yang mendekati biaya sebenarnya.

Metode Best First Search

- Metode *Best First Search* merupakan metode pencarian yang menitik-beratkan pengembangan suatu lokasi pencarian di sebuah node yang dinilai paling menguntungkan berdasarkan suatu nilai estimasi tertentu.
- Umumnya *Best-First Search* melakukan proses searching dengan cara memberikan estimasi berapa jauh node asal dari solusi yang diinginkan.
- Dengan metode ini, proses dilakukan dengan melakukan ekspansi terhadap setiap node yang memiliki estimasi terpendek.
- Varian dari Best First Search adalah Greedy Best First Search dan A* (baca : A-Star).

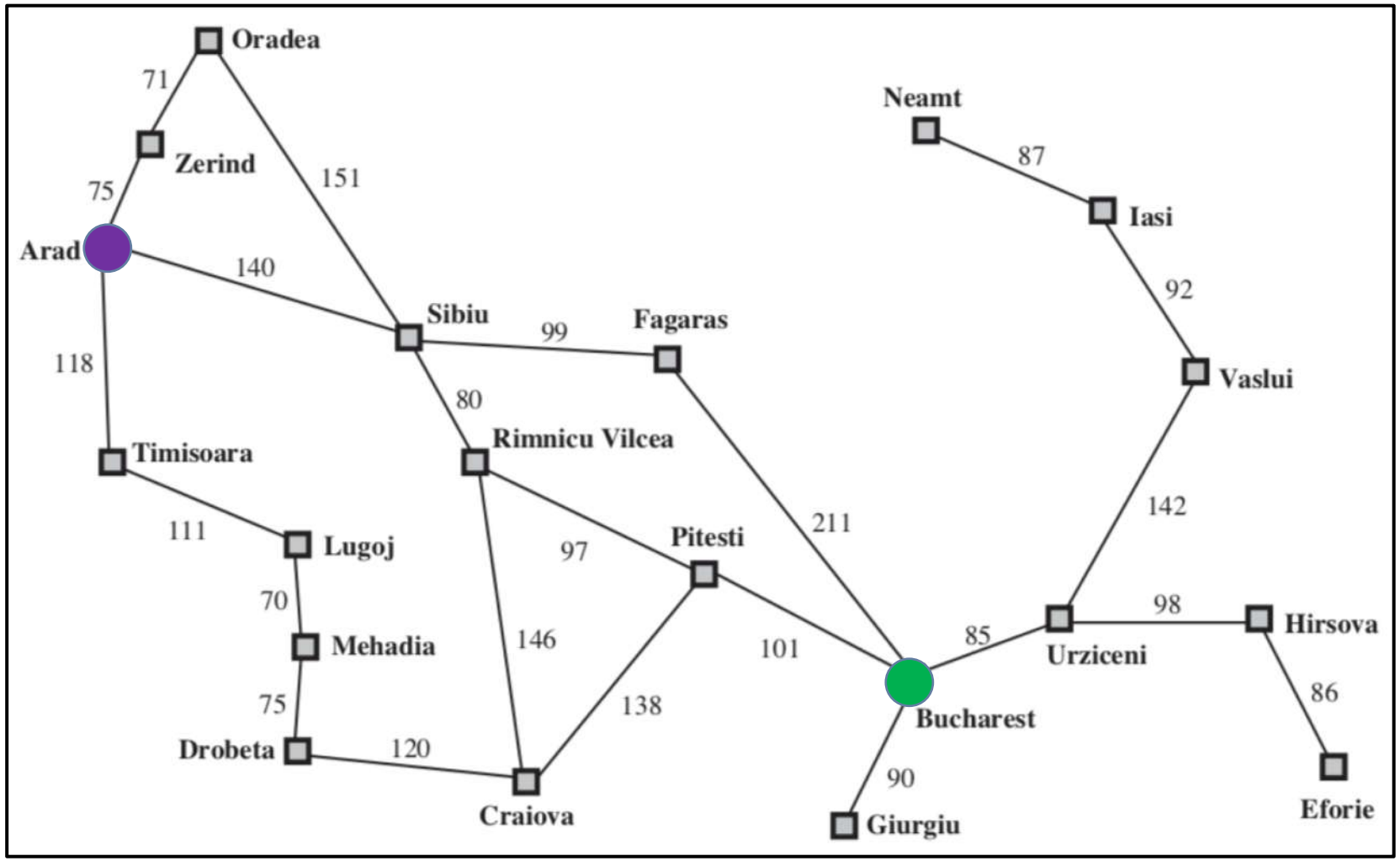
Metode Greedy Best First Search

- Pada metode ini, fungsi evaluasi ***f(n)*** adalah sama dengan fungsi heuristik ***h(n)***, dimana *h(n)* merupakan estimasi biaya dari suatu lokasi ke tujuan.

- Fungsi heuristic metode ini, dinotasikan sebagai $\mathbf{h_{SLD}(n)}$, yaitu harga **Straight-Line-Distance** dari satu lokasi ke tujuan.

- Greedy Best First Search melakukan ekspansi terhadap node yang terlihat paling mendekati ke tujuan.

Contoh Penerapan Metode Greedy Best First Search (GBFS)

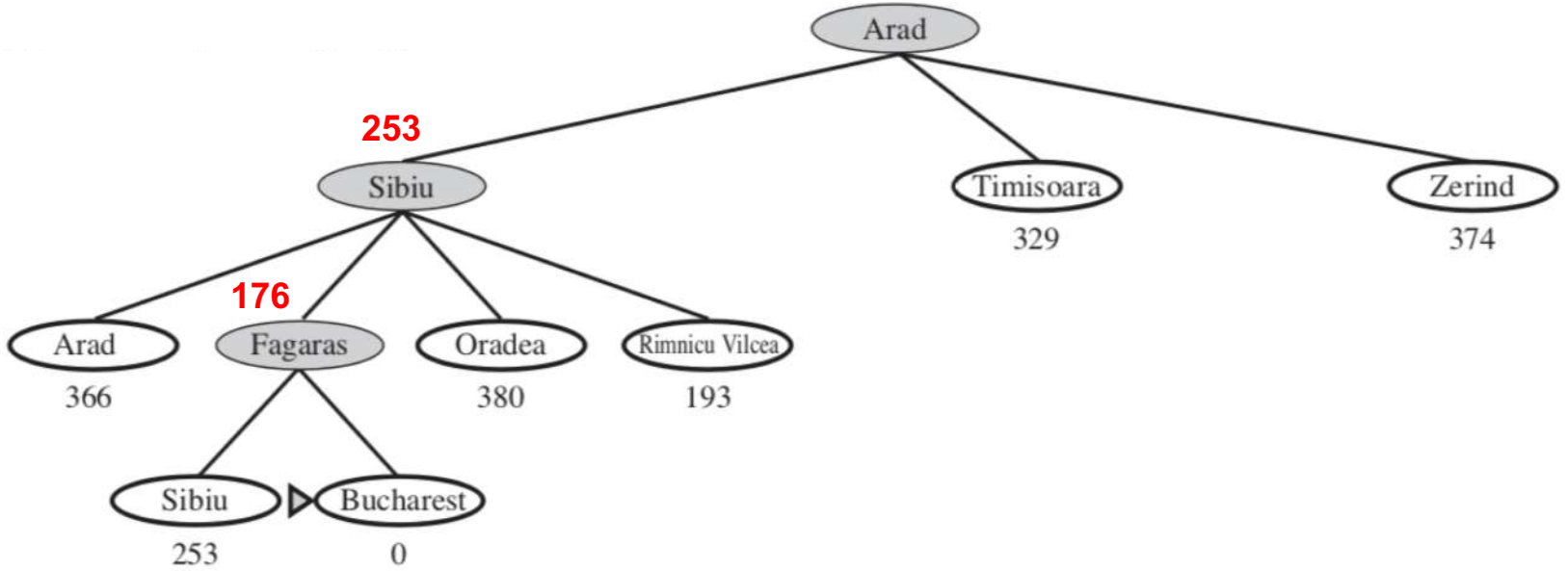


S = Arad

G = Bucharest

Kota	SLD	Kota	SLD
Mehadia	241	Arad	366
Neamt	234	Bucharest	0
Oradea	380	Craiova	160
Pitesti	100	Dobreta	242
Rimnicu Vilcea	193	Eforie	161
Sibiu	253	Fagaras	176
Timisoara	329	Giurgiu	77
Urziceni	85	Hirsova	151
Vaslui	199	Iasi	226
Zerind	374	Lugoj	244

Langkah Pengerjaan Metode GBFS



Kota	SLD
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244

Kota	SLD
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	85
Vaslui	199
Zerind	374

Contoh algoritme GBFS

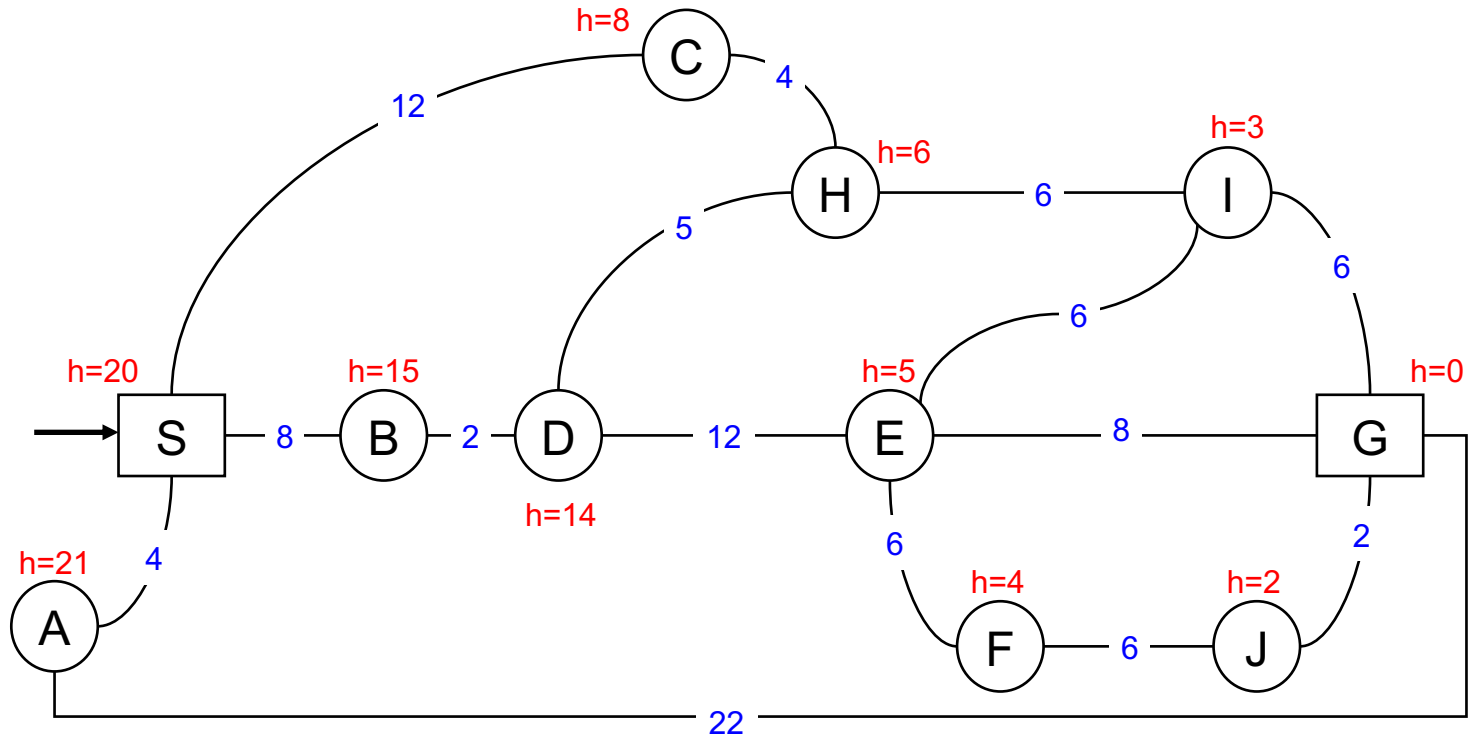
Algorithm 9.2 Best-first Search(Input: **S**, **Goal**)

```
1: Open = [S]
2: Closed = []
3: repeat
4:    if Open.Head = Goal then
5:       return success
6:    end if
7:    generate children's set C of Open.Head
8:    if n ∈ C already exists in OPEN and new n is reachable by shorter path then
9:       remove the old n
10:   end if
11:   if n ∈ C already exists in Closed and reachable by shorter path then
12:      replace n ∈ C by the same node from Closed, along with shorter distance from root
13:   end if
14:   remove Open.Head and insert into Closed
15:   update distance from root for all C nodes
16:   add all C to either side of Open and record their parents
17:   sort Open by path length so that least cost path node is at front
18: until Open = nil
19: return fail
```

Source: Chowdhary, Fundamentals of Artificial Intelligence. 2020

Latihan 1

Gunakan metode greedy best first search untuk menentukan rute dari S menuju G, dimana biaya estimasi heuristik ditentukan oleh jarak garis lurus suatu lokasi ke G.



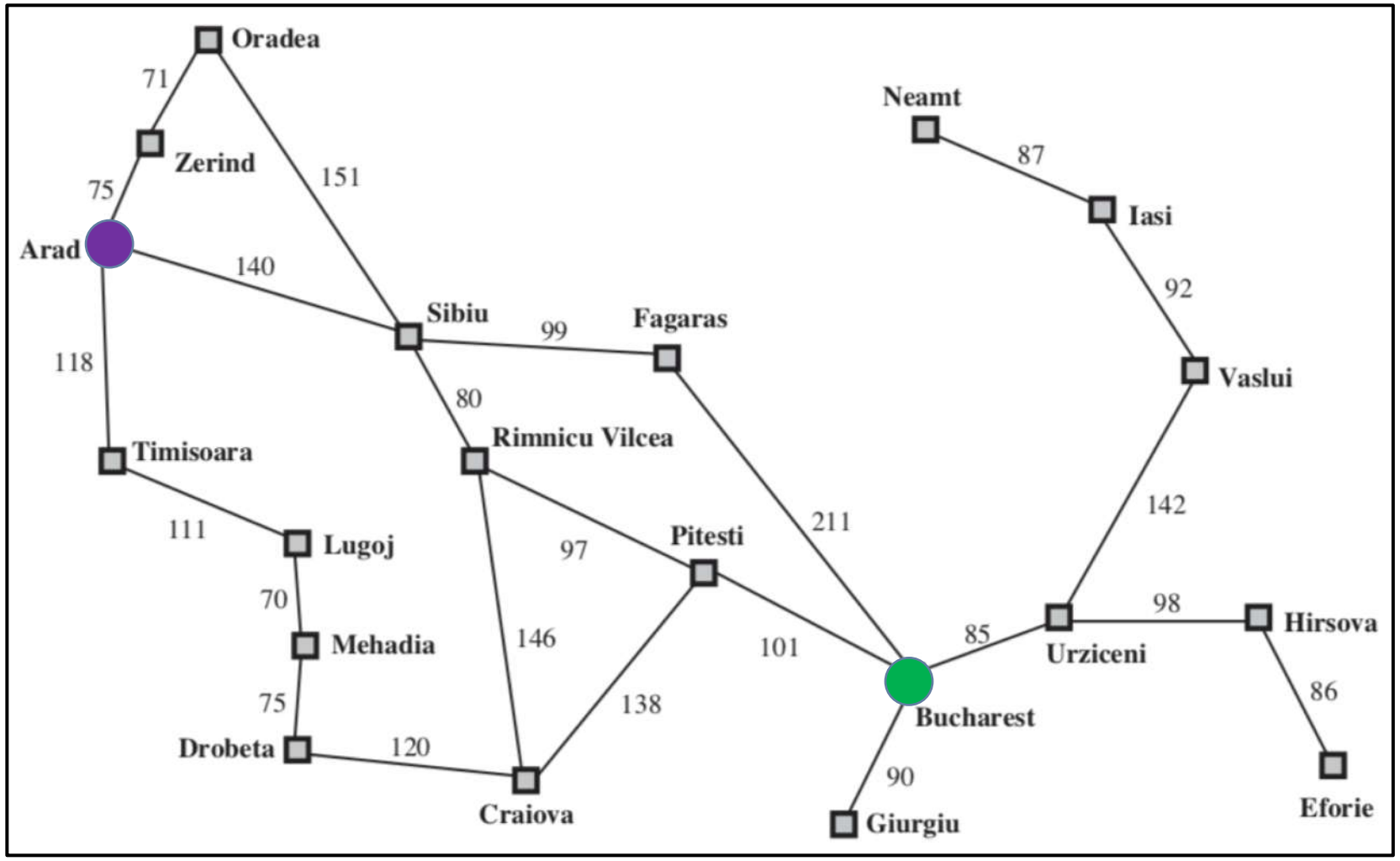
Metode Pencarian A*

- Metode ini dirancang untuk memperbaiki kelemahan metode Greedy dengan cara menghindari ekspansi terhadap rute yang sudah tergolong mahal.

- Fungsi evaluasi f(n) = g(n) + h(n),
- Dimana g(n) adalah biaya yang sudah dikeluarkan hingga suatu node, dan h(n) adalah estimasi biaya yang perlu dikeluarkan dari suatu node hingga node tujuan.

- Dengan kata lain, A* menghitung total biaya yang sudah dikeluarkan ditambah estimasi biaya yang perlu dikeluarkan. A* melakukan ekspansi terhadap node dengan f(n) paling rendah.

Contoh Penerapan Metode A*



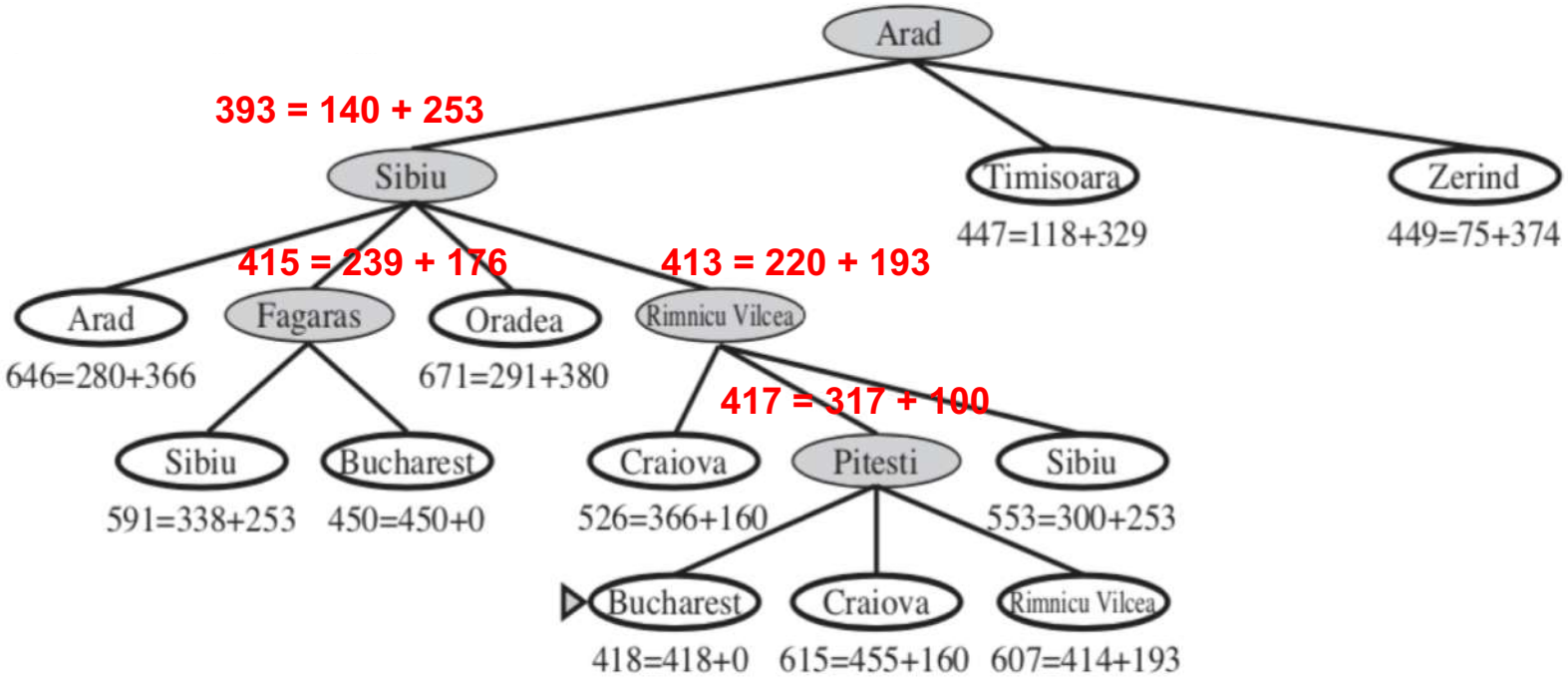
S = Arad

G = Bucharest

Kota	SLD
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	85
Vaslui	199
Zerind	374

Kota	SLD
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244

Langkah Pengerjaan Metode A*



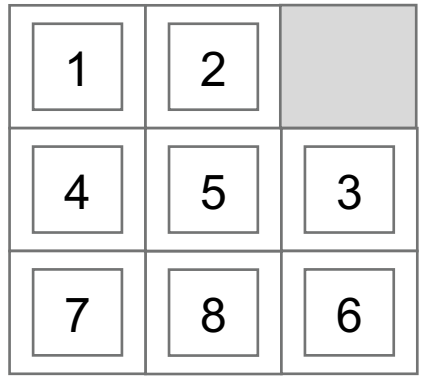
Kota	SLD
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244

Kota	SLD
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	85
Vaslui	199
Zerind	374

Contoh Lain Penggunaan A*

- Diketahui sebuah puzzle berukuran 3X3 yang berisi angka.
- Permasalahan adalah angka-angka dalam puzzle tersebut belum teratur.

Initial State

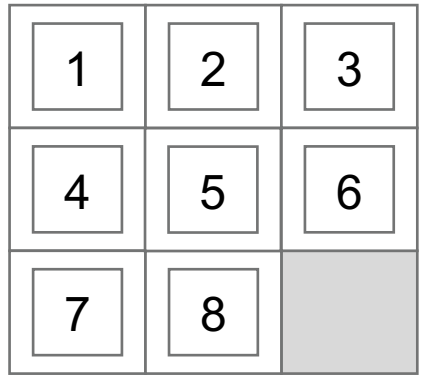


Nilai awal = {1,2,blank,4,5,3,7,8,6}

Goal = {1,2,3,4,5,6,7,8,blank}

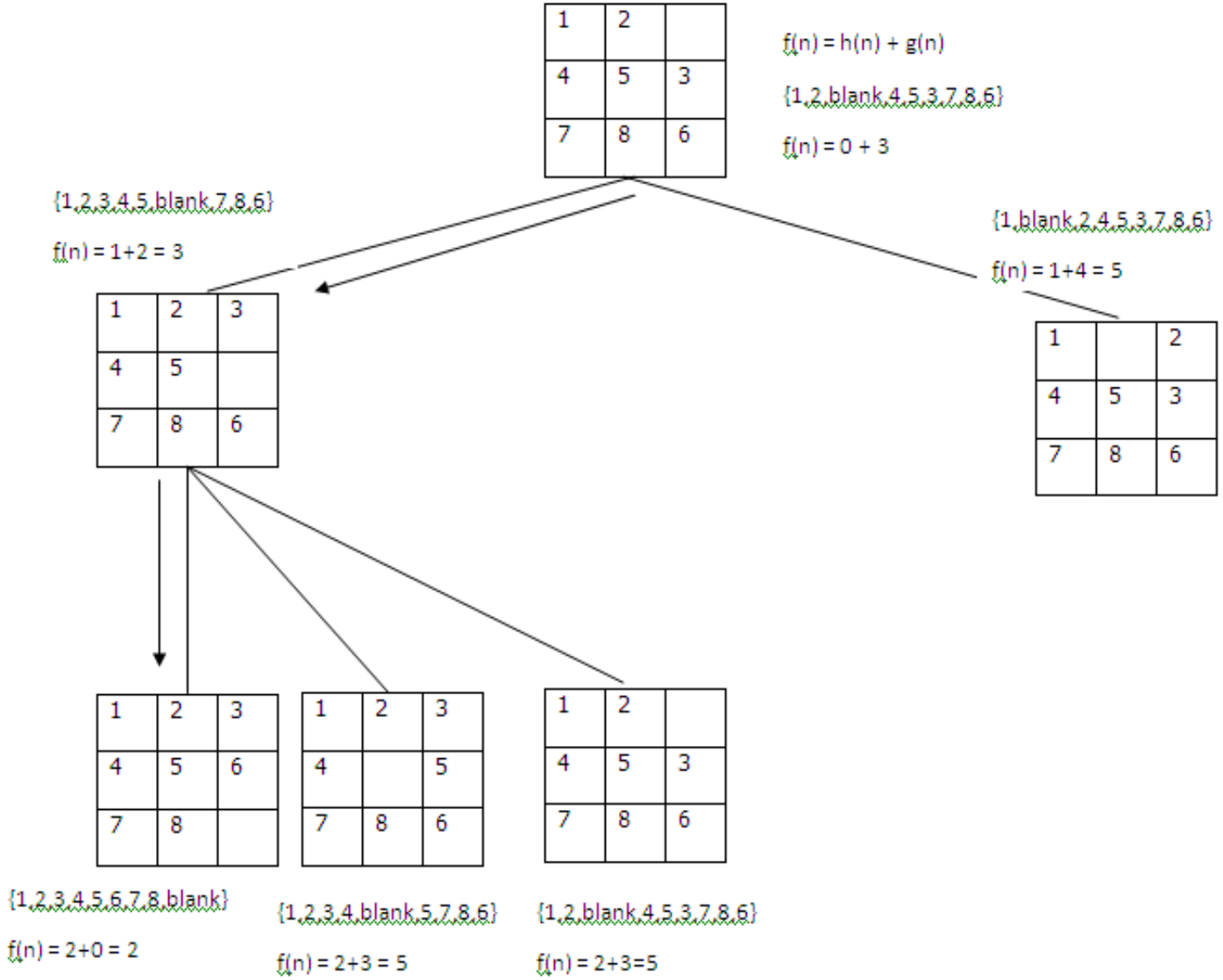
Nilai Evaluasi = f(n) = g(n) + h(n)

- g(n) = kedalaman dari pohon
- h(n) = jumlah angka yang masih salah posisi



Goal State

Langkah Penyelesaian A* Terhadap Puzzle 3x3



Contoh Algoritme A*

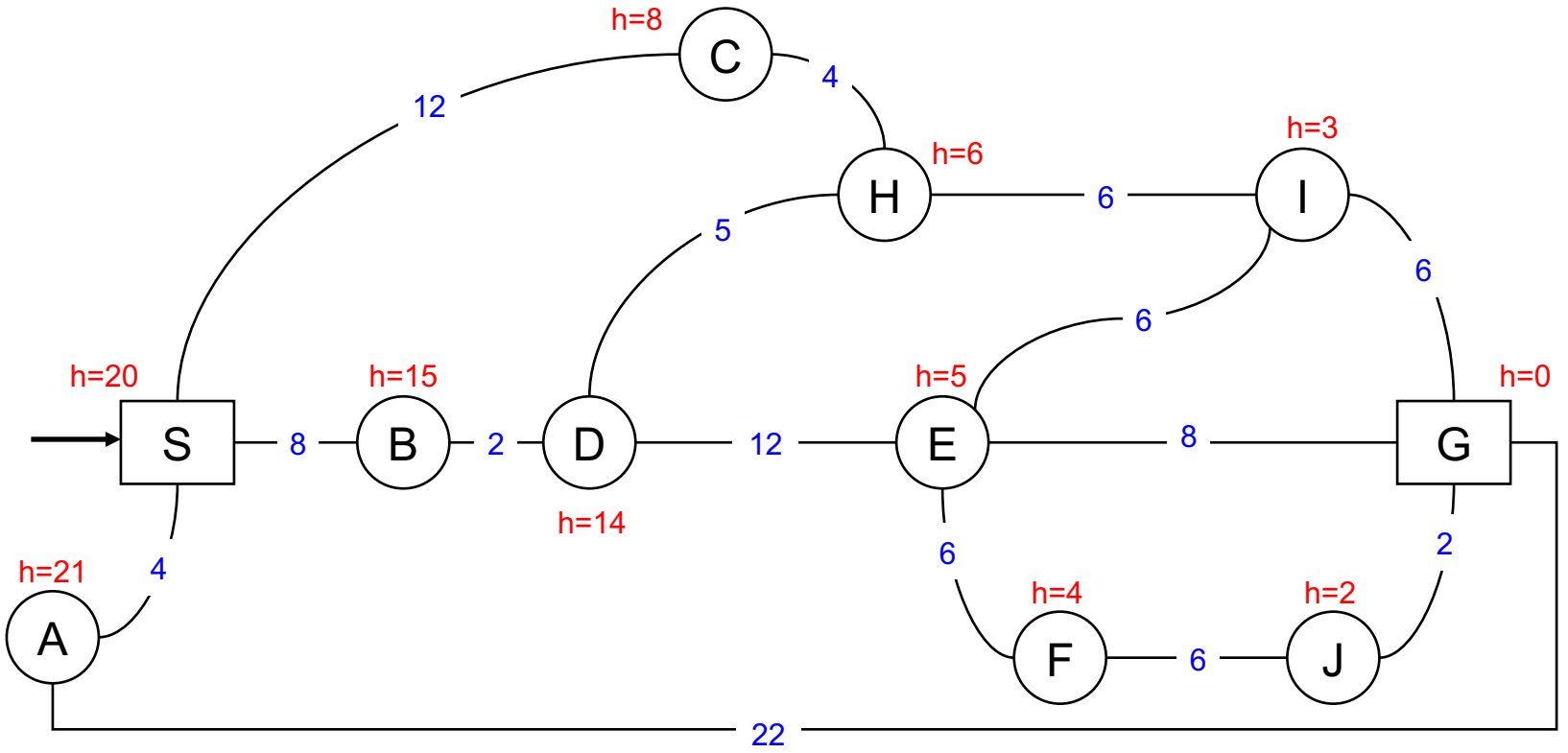
Algorithm 9.3 Admissible search A^*(Input: **G**, **S**, **Goal**)

1: **Open** $= [S]$
2: **Closed** $= []$
3: compute $f^*(n)$ for all $n \in$ **Open**
4: **repeat**
5: select the open node n whose $f^*(n)$ is smallest
6: resolve ties arbitrarily, but always in favor of any node $n \in$ **Goal**
7: **if** $n \in$ **Goal then**
8: move n to **Closed**
9: terminate algorithm
10: **else**
11: move n to **Closed**
12: apply successor operator to n
13: calculate f^* for each successor n_i of n
14: move all $n_i \notin$ **Closed** to **Open**
15: move to **Open** any $n_i \in$ **Closed** and for which $f^*(n_i)$ is smaller now than it was when n_i was in **Closed**
16: **end if**
17: **until Open** $= nil$
18: return *fail*

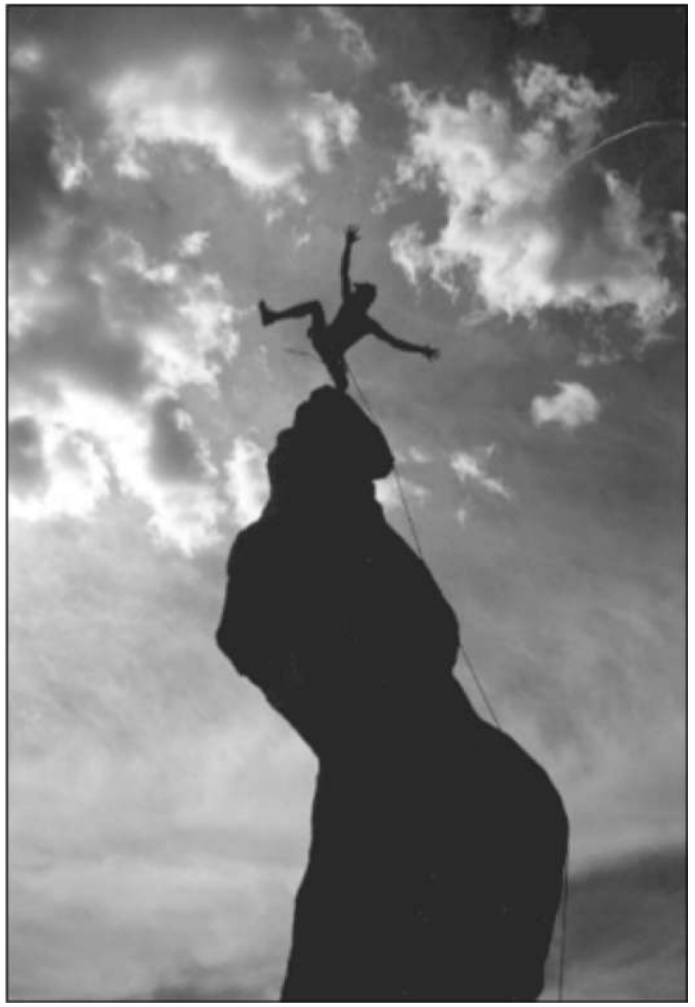
Source: Chowdhary, Fundamentals of Artificial Intelligence. 2020

Latihan 2

Gunakan metode A* untuk menentukan rute harus dilalui dari S ke G, dimana biaya estimasi heuristik ditentukan oleh jarak garis lurus suatu lokasi ke G.



Hill Climbing (HC)



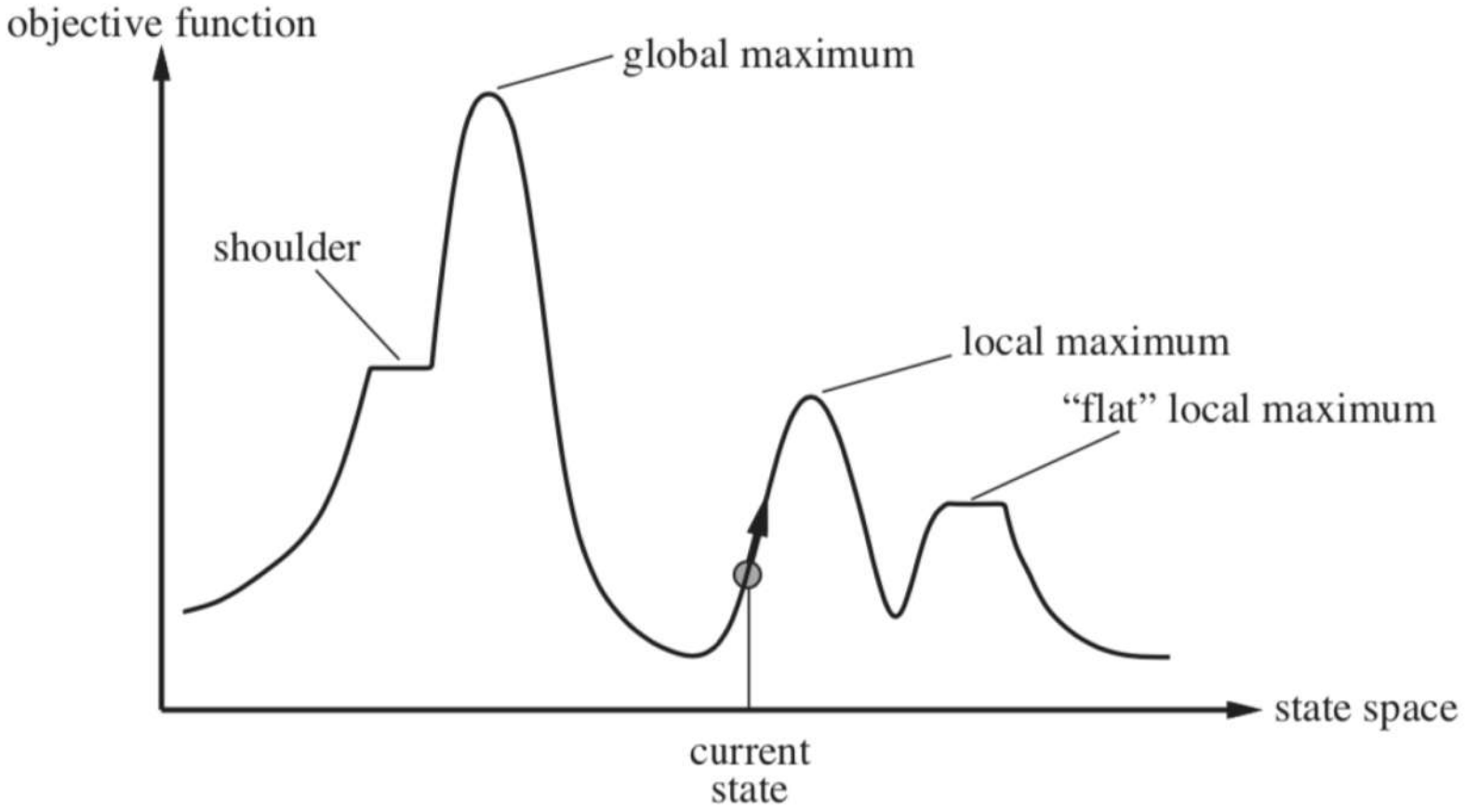
Metode Hill Climbing (HC)

- Metode Hill-climbing merupakan variasi dari ***depth-first search***.
- Dengan metode ini, eksplorasi terhadap keputusan dilakukan dengan cara *depth-first search* dengan mencari path yang bertujuan **menurunkan *cost*** untuk menuju kepada goal/keputusan ➔ Mirip seperti UCS, hanya saja dilakukan untuk node anak paling awal yang lebih murah dibandingkan dengan node orang-tua.

- Analogi HC adalah seperti mencoba menaiki Gunung Himalaya di tengah kabut tebal dan sedang terkena amnesia.

- Dengan cara demikian sebetulnya kita **berasumsi** bahwa secara umum arah tertentu semakin dekat ke puncak gunung.

Visualisasi Metode HC



Algoritma Simple Hill Climbing

- Evaluasi state awal, jika state awal sama dengan tujuan, maka proses berhenti. Jika tidak sama dengan tujuan maka lanjutkan proses dengan membuat state awal sebagai state sekarang.

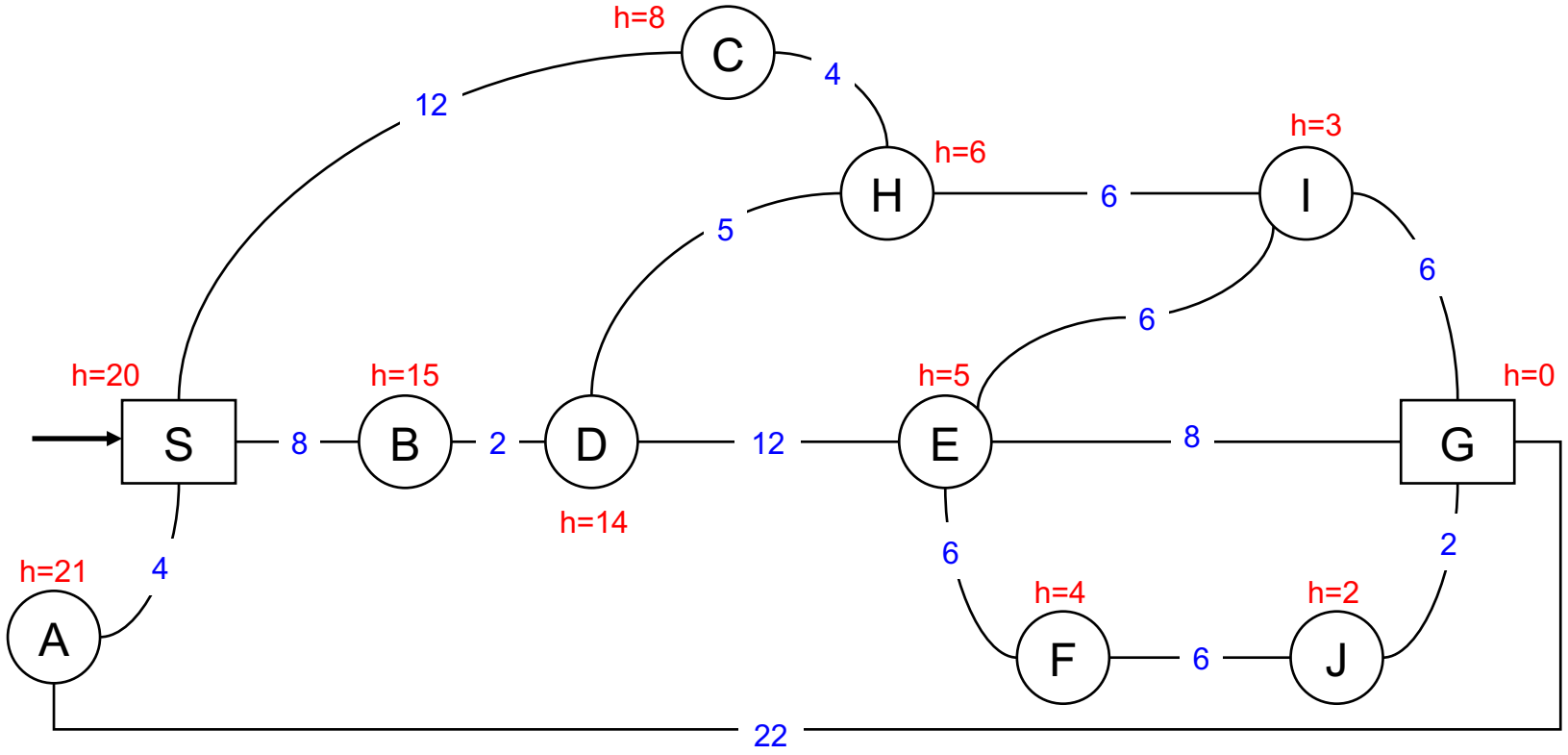
- Kerjakan langkah berikut sampai solusi ditemukan atau sampai tidak ada lagi operator baru yang dapat digunakan dalam state sekarang:
  - Cari sebuah operator yang belum pernah digunakan dalam state sekarang dan gunakan operator tersebut untuk membentuk state baru.
  - Evaluasi state baru.
    - Jika **state baru adalah tujuan**, maka proses **berhenti**
    - Jika state baru tersebut **bukan tujuan** tetapi state baru **lebih baik daripada state sekarang**, maka buat **state baru menjadi state sekarang**.
    - Jika **state baru tidak lebih baik daripada state sekarang**, maka lanjutkan iterasi.

3 Masalah Pada Simple HC

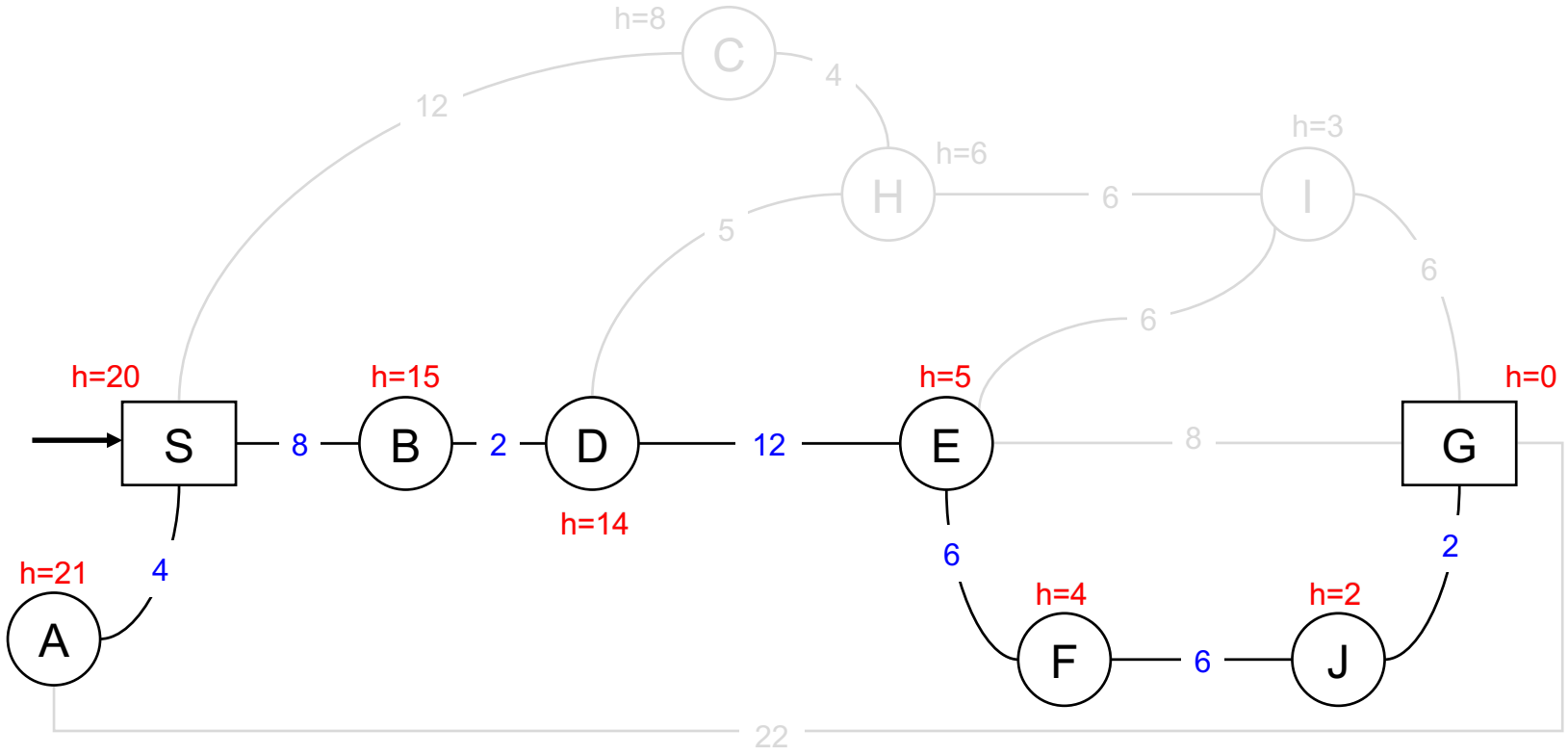
- Algoritma akan berhenti kalau mencapai nilai optimum local
- Urutan penggunaan operator akan sangat berpengaruh pada penemuan solusi
- Tidak diijinkan untuk melihat satupun langkah selanjutnya.

Contoh Penggunaan Simple HC pada jalur

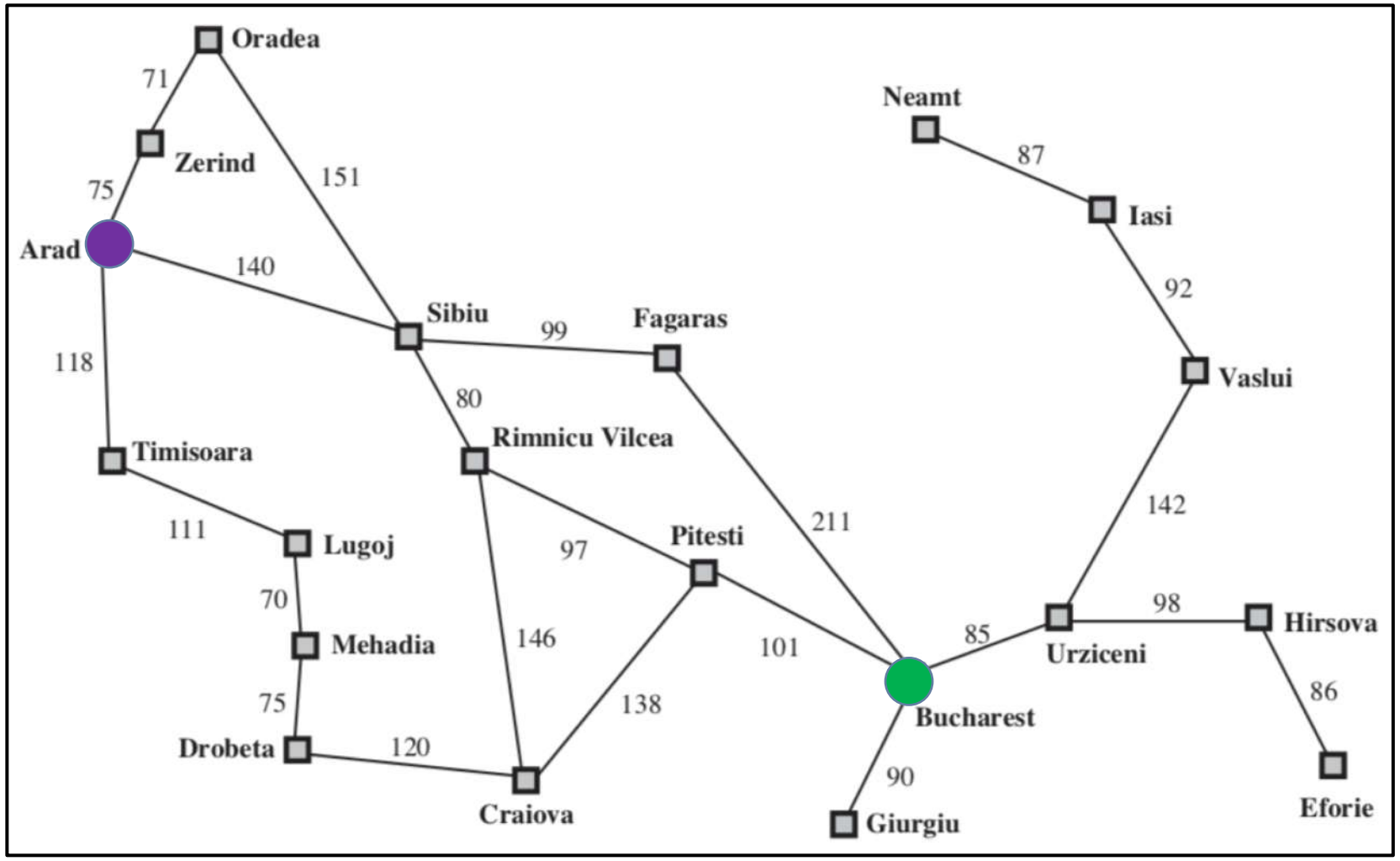
Gunakan metode Simple Hill Climbing untuk menentukan rute harus dilalui dari S ke G, dimana biaya estimasi heuristik ditentukan oleh jarak garis lurus suatu lokasi ke G.



Solusi


h=8
C
12
4
h=6
H
6
h=3
I
5
6
6
h=20
S
8
h=15
B
2
D
12
h=5
E
8
h=0
G
h=14
h=21
4
6
2
h=4
h=2
A
F
6
J
22

Latihan 3



S = Arad

G = Bucharest

Kota	SLD
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	85
Vaslui	199
Zerind	374

Kota	SLD
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244

Metode Steepest (Steepest-Ascent) Hill Climbing (St-HC)

- Metode ini mirip dengan simple hill climbing

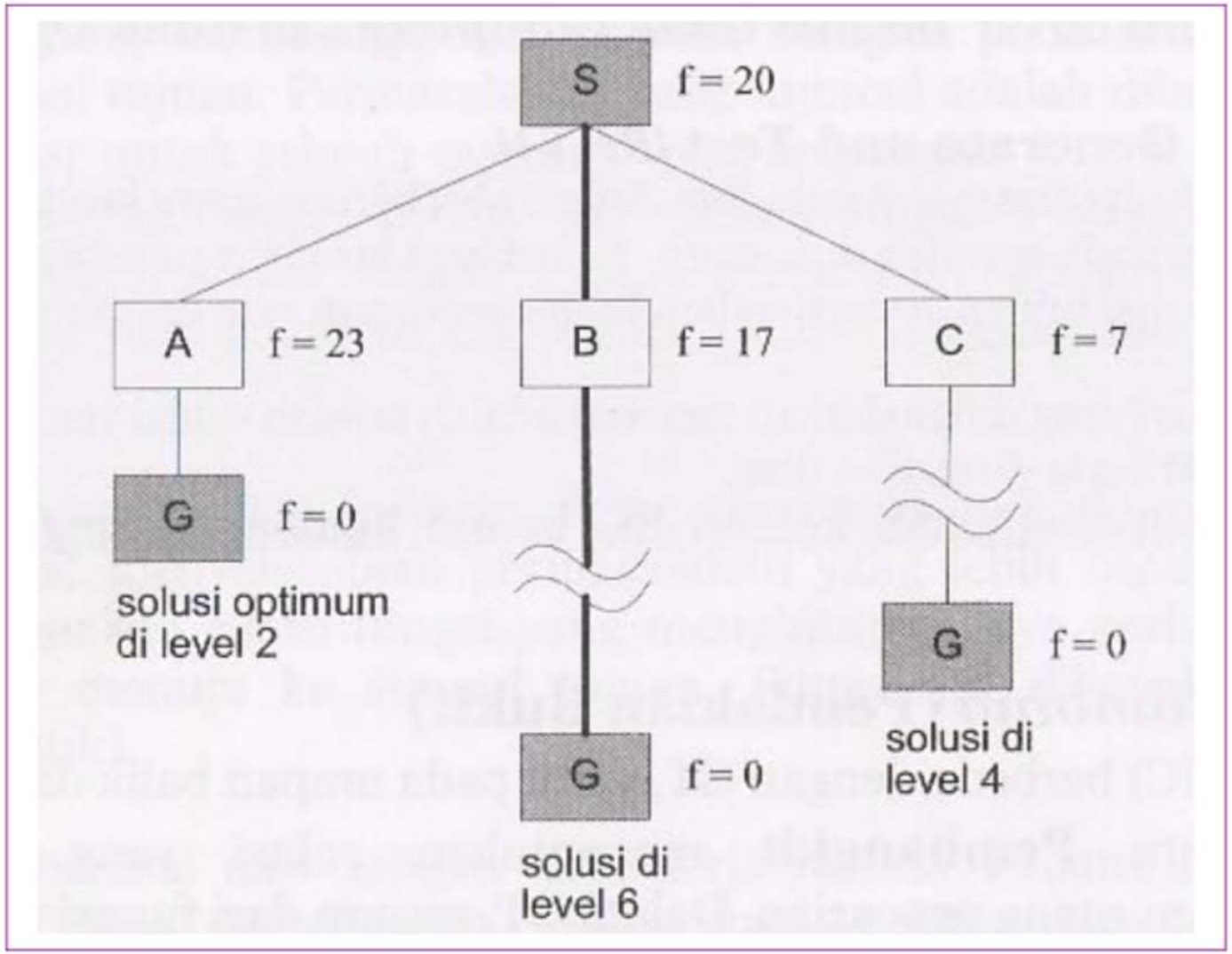
- Perbedaannya dengan simple hill climbing :
  - Semua suksesor dibandingkan, dan yang paling dekat dengan solusi yang dipilih
  - Pada simple hill climbing, node pertama yang jaraknya terdekat dengan solusi yang dipilih, sedangkan pada St-HC, node anak termurah yang dipilih

Algoritma Steepest Hill Climbing

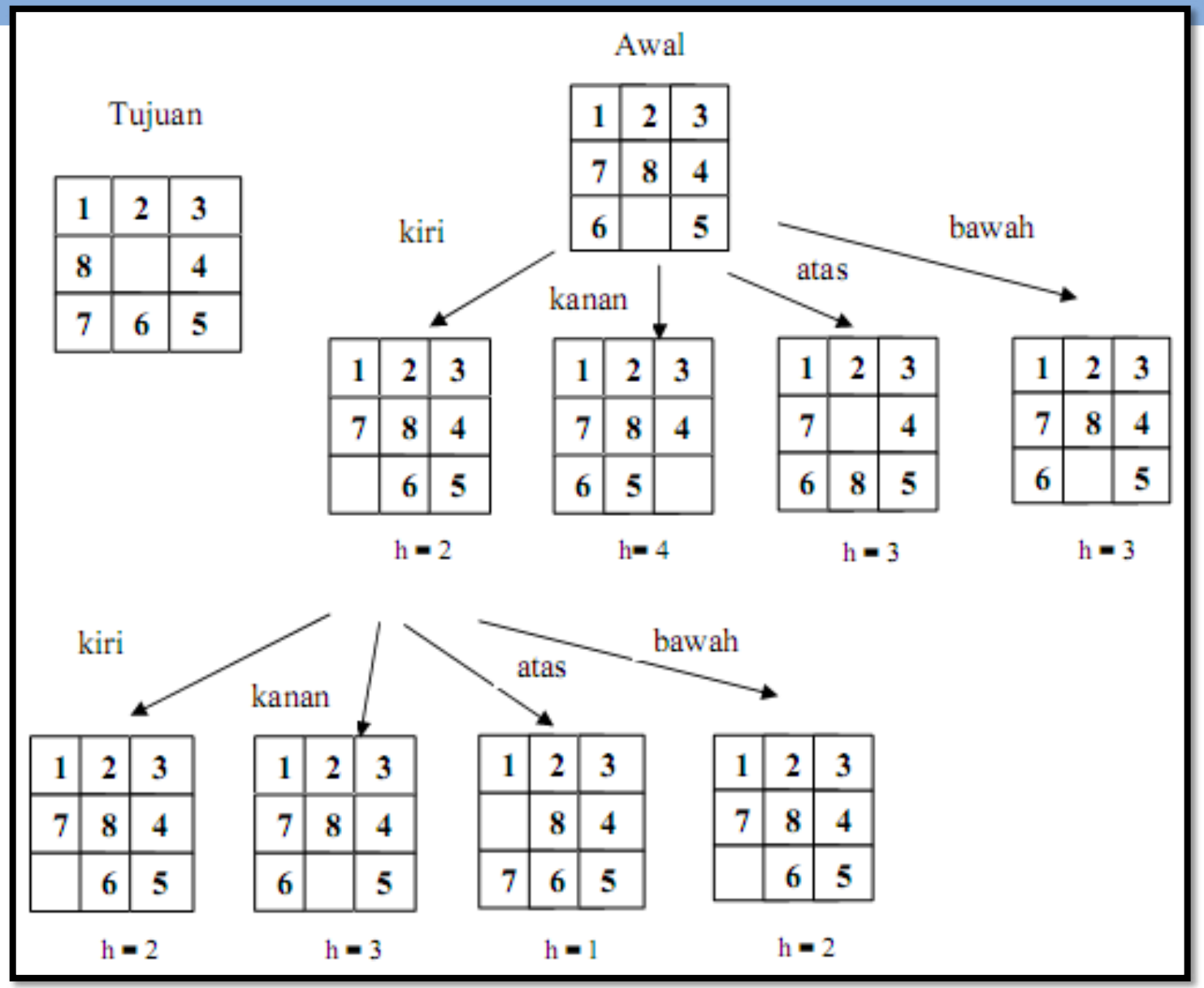
- Evaluasi keadaan awal (Initial State). Jika keadaan awal sama dengan tujuan (Goal state) maka kembali pada initial state dan berhenti berproses. Jika tidak maka initial state tersebut jadikan sebagai current state.

- Mulai dengan current state = initial state.
- Dapatkan semua pewaris (successor) yang dapat dijadikan next state pada current statenya dan evaluasi successor tersebut dengan fungsi evaluasi dan beri nilai pada setiap successor tersebut.
- Jika salah satu dari successor tersebut mempunyai nilai yang lebih baik dari current state maka jadikan successor dengan nilai yang paling baik tersebut sebagai new current state.
- Lakukan operasi ini terus menerus hingga tercapai current state = goal state atau tidak ada perubahan pada current statenya.

Contoh


S
f = 20
A
f = 23
B
f = 17
C
f = 7
G
f = 0
solusi optimum
di level 2
G
f = 0
solusi di
level 6
G
f = 0
solusi di
level 4

Contoh Penggunaan HC



Contoh St-HC pada Problem 8-Puzzle

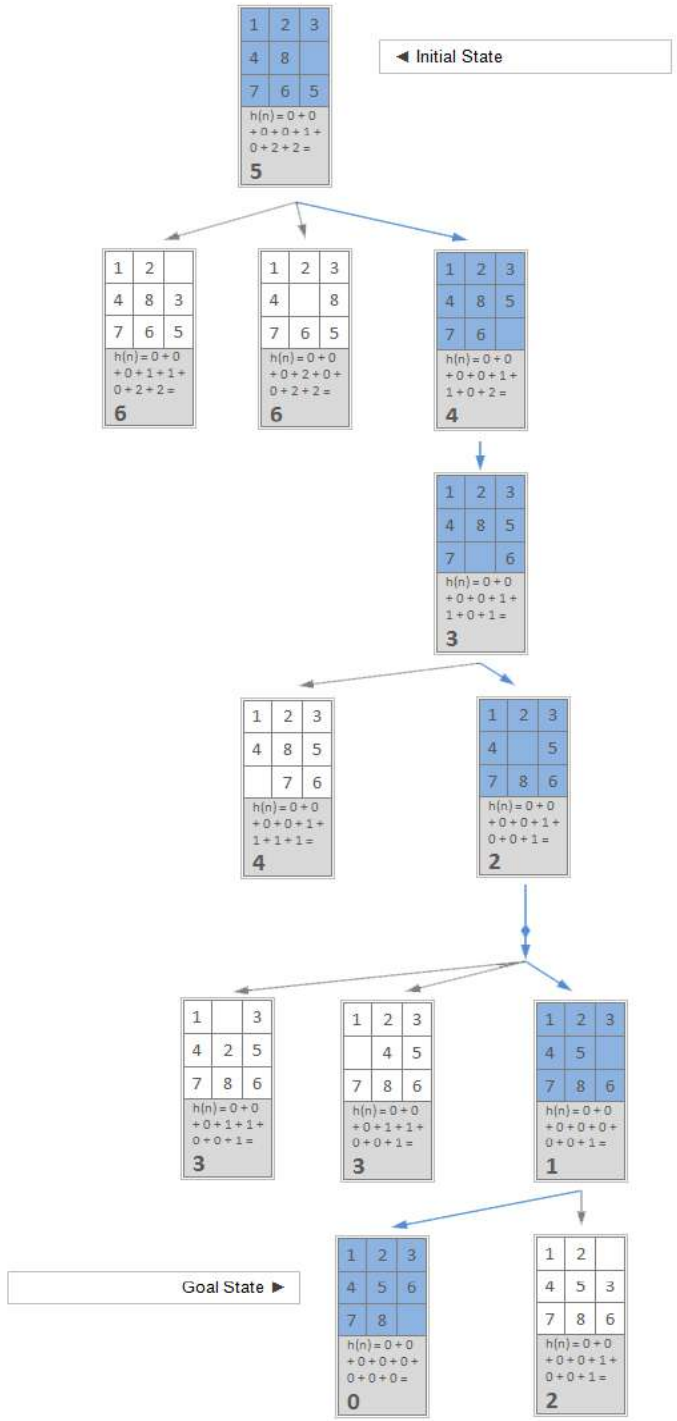
1	2	3
4	8	
7	6	5

Start

1	2	3
4	5	6
7	8	

Goal

Bagaimana Langkah HC-nya?



Contoh algoritme

Algorithm 9.1 Hill-Climb(Input: **G**, **S**, **Goal**)

```
1: Open = [S]
2: Closed = nil
3: if Open = nil then
4:    return fail
5: end if
6: repeat
7:    if Open.Head = Goal then
8:       return success
9:    end if
10:   expand Open.Head and generate children's set, call it C
11:   reject all paths in C having loops
12:   delete Open.Head and insert it into Closed
13:   sort C in order of heuristic, with best heuristic node in the front
14:   insert C at the front of List
15: until Open = nil
16: Return fail
```

Source: Chowdhary, Fundamentals of Artificial Intelligence. 2020

Latihan 4

Gunakan metode Steepest Hill Climbing untuk menentukan rute harus dilalui dari S ke G, dimana biaya estimasi heuristik ditentukan oleh jarak garis lurus suatu lokasi ke G.