Searching

Mesin Learning

By. Gunawansyah

Metode-metode pencarian

- ***Blind* (*un-informed*)**
  - Tanpa informasi
  - Kompleksitas tinggi
- ***Heuristic* (*informed*)**
  - Dengan informasi
  - Kompleksitas relatif rendah

Metode-metode pencarian

Blind (*un-informed*)	*Heuristic* (*informed*)
a) Breadth First Search (BFS) b) Depth First Search (DFS) c) Depth Limited Search (DLS) d) Uniform Cost Search (UCS) e) Iterative Deepening Search (IDS) f) Bi-Directional Search (BDS)	a) Hill Climbing b) Simulated Annealing (SA) c) Greedy Best First Search d) A* e) Iterative Deepening A* (IDA*) f) Simplified Memory-Bounded A* (SMA*) g) Bi-directional A* (BDA*) h) Modified Bi-directional A* (MBDA*) i) Dynamic Weighting A* (DWA*) j) Beam A* (BA*)

Ukuran Performansi

- ***Completeness***

  Apakah metode tersebut **menjamin penemuan solusi** jika solusinya memang ada?

- ***Optimality***

  Apakah metode tersebut menjamin menemukan solusi yang **terbaik** jika terdapat beberapa solusi berbeda?

- ***Time complexity***

  Berapa lama **waktu** yang diperlukan?

- ***Space complexity***

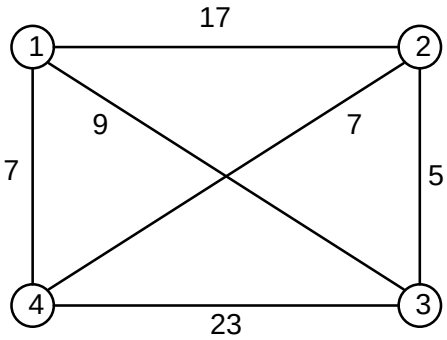
  Berapa banyak **memori** yang diperlukan?

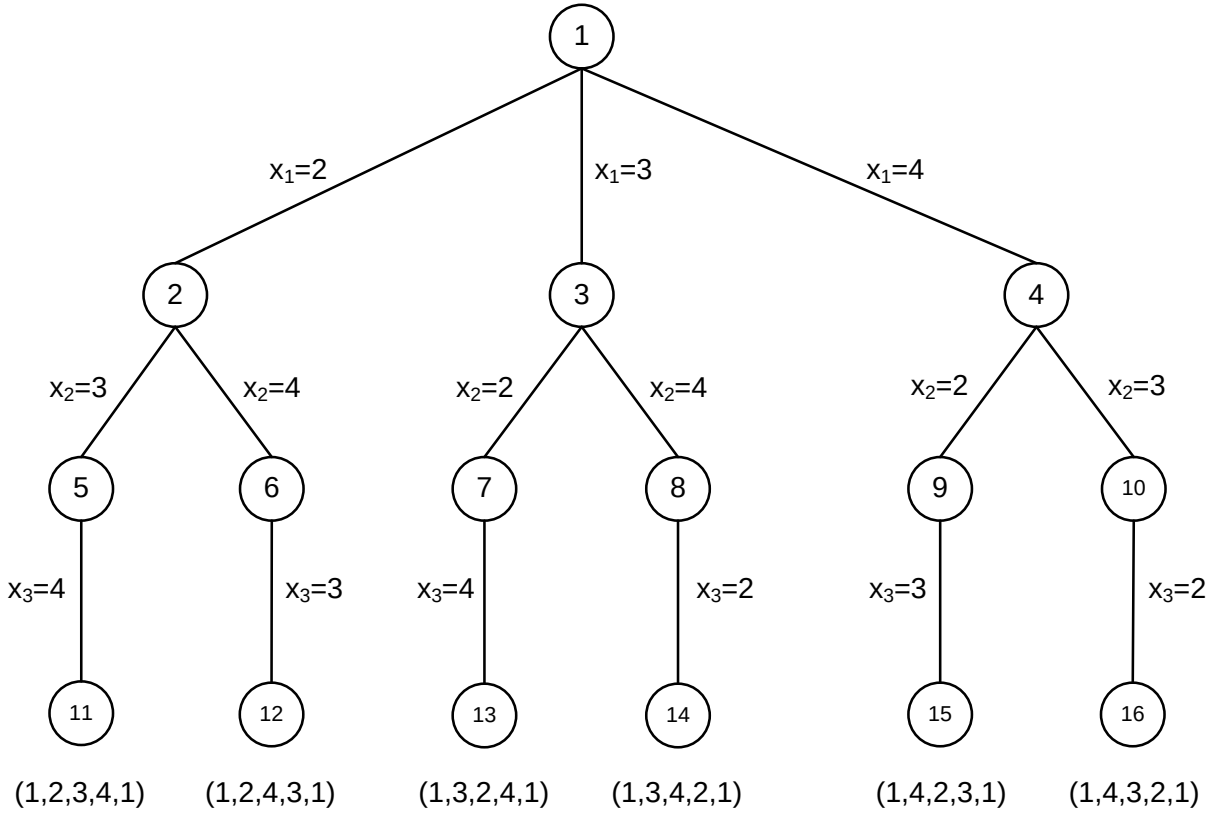
Traveling Salesman Problem

- Pencarian urutan semua lokasi yang harus dikunjungi
- Mulai dari suatu kota dan kembali ke kota tersebut
- Meminimalkan total biaya.
- Setiap kota harus dikunjungi satu kali.

Himpunan Operator TSP

- Aturan untuk memilih lokasi yang belum pernah terpilih satu demi satu sehingga dihasilkan satu rute kunjungan yang lengkap dari lokasi awal kemudian mengunjungi semua lokasi yang lain tepat satu kali dan akhirnya kembali ke lokasi awal.


17
1
2
9
7
7
5
4
3
23


1
x1=2
x1=3
x1=4
2
3
4
x2=3
x2=4
x2=2
x2=4
x2=2
x2=3
5
6
7
8
9
10
x3=4
x3=3
x3=4
x3=2
x3=3
x3=2
11
12
13
14
15
16
(1,2,3,4,1)
(1,2,4,3,1)
(1,3,2,4,1)
(1,3,4,2,1)
(1,4,2,3,1)
(1,4,3,2,1)

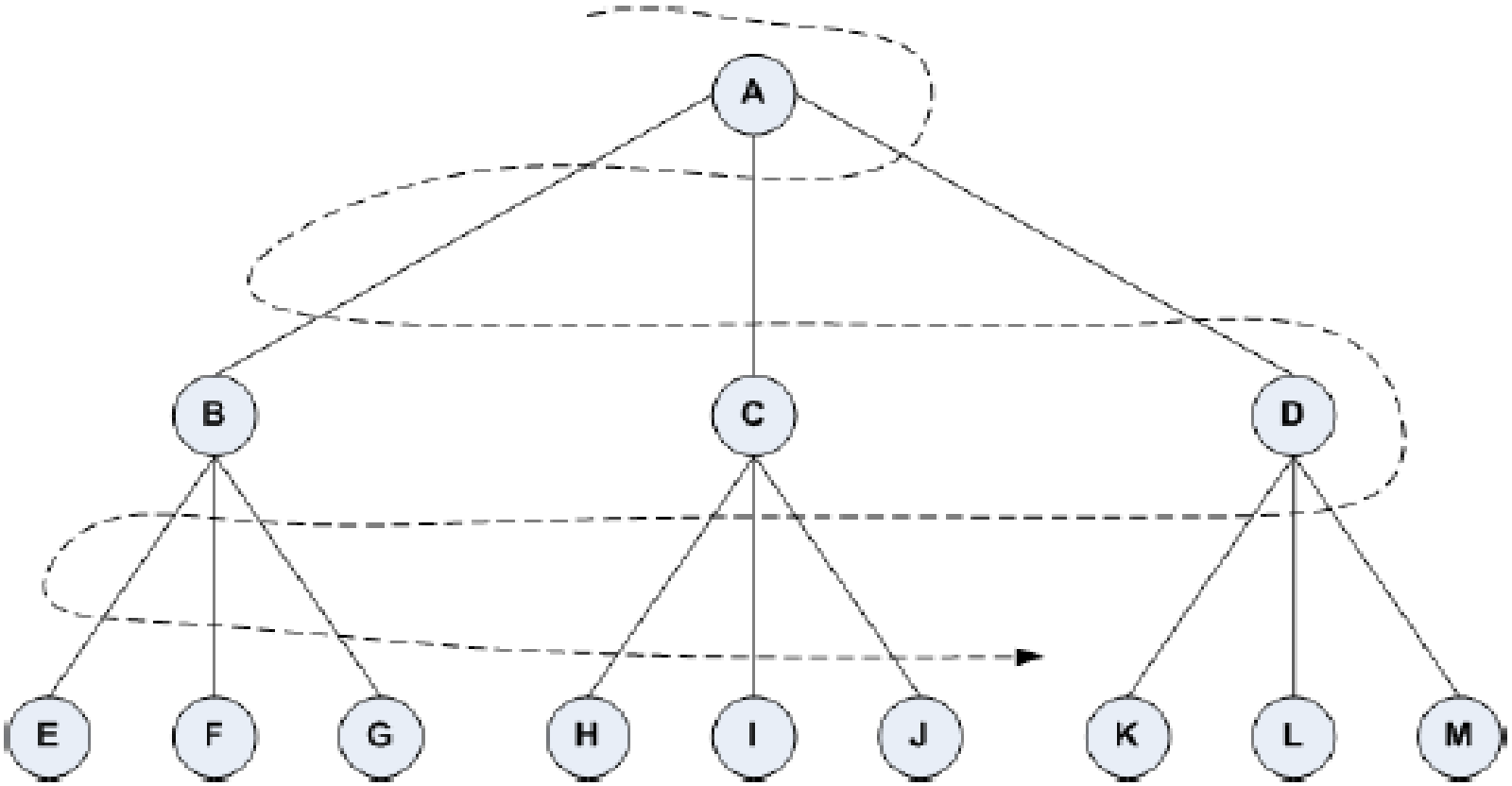


Figure 1.6: Diagram pohon dari BFS.

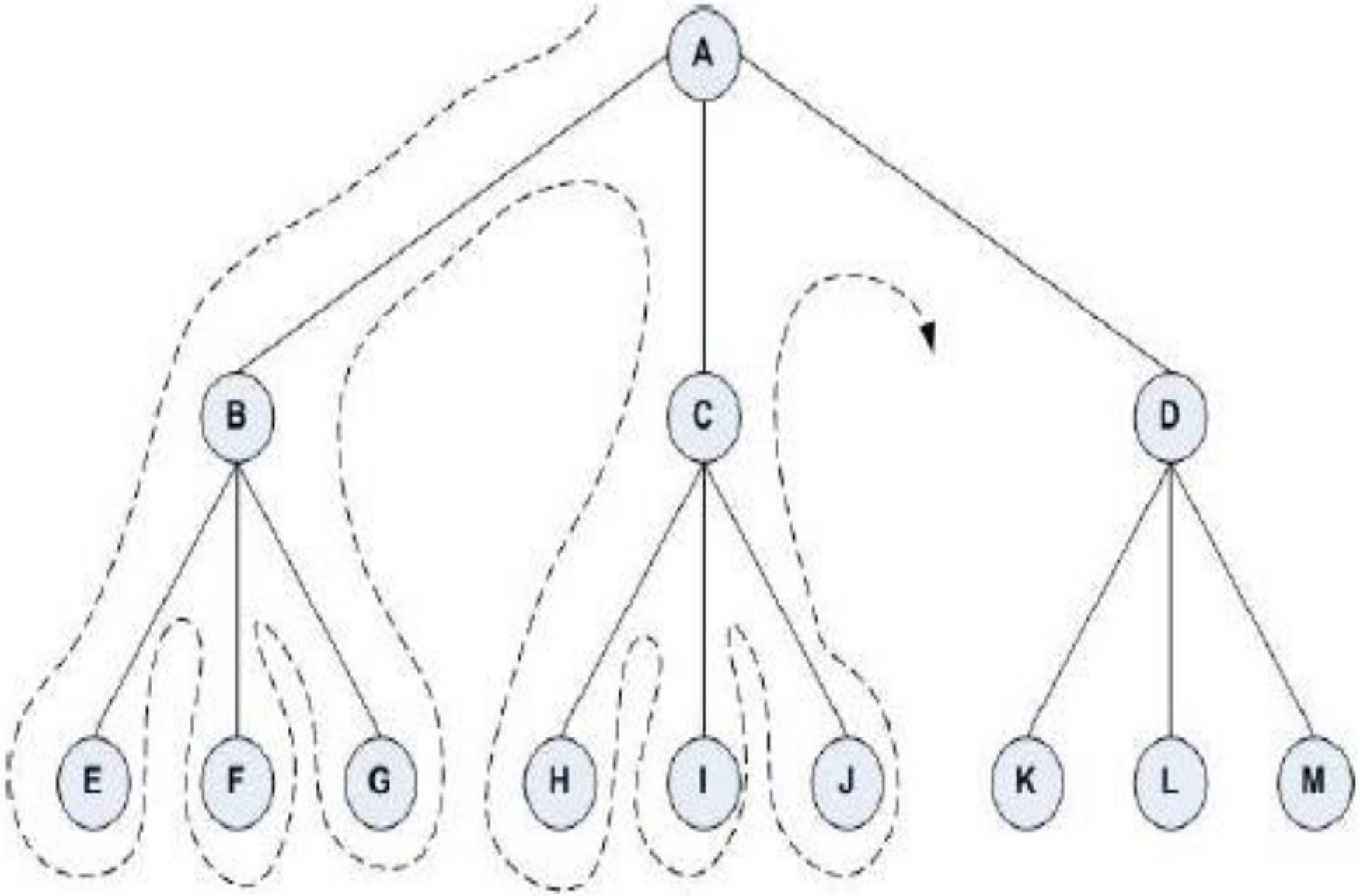
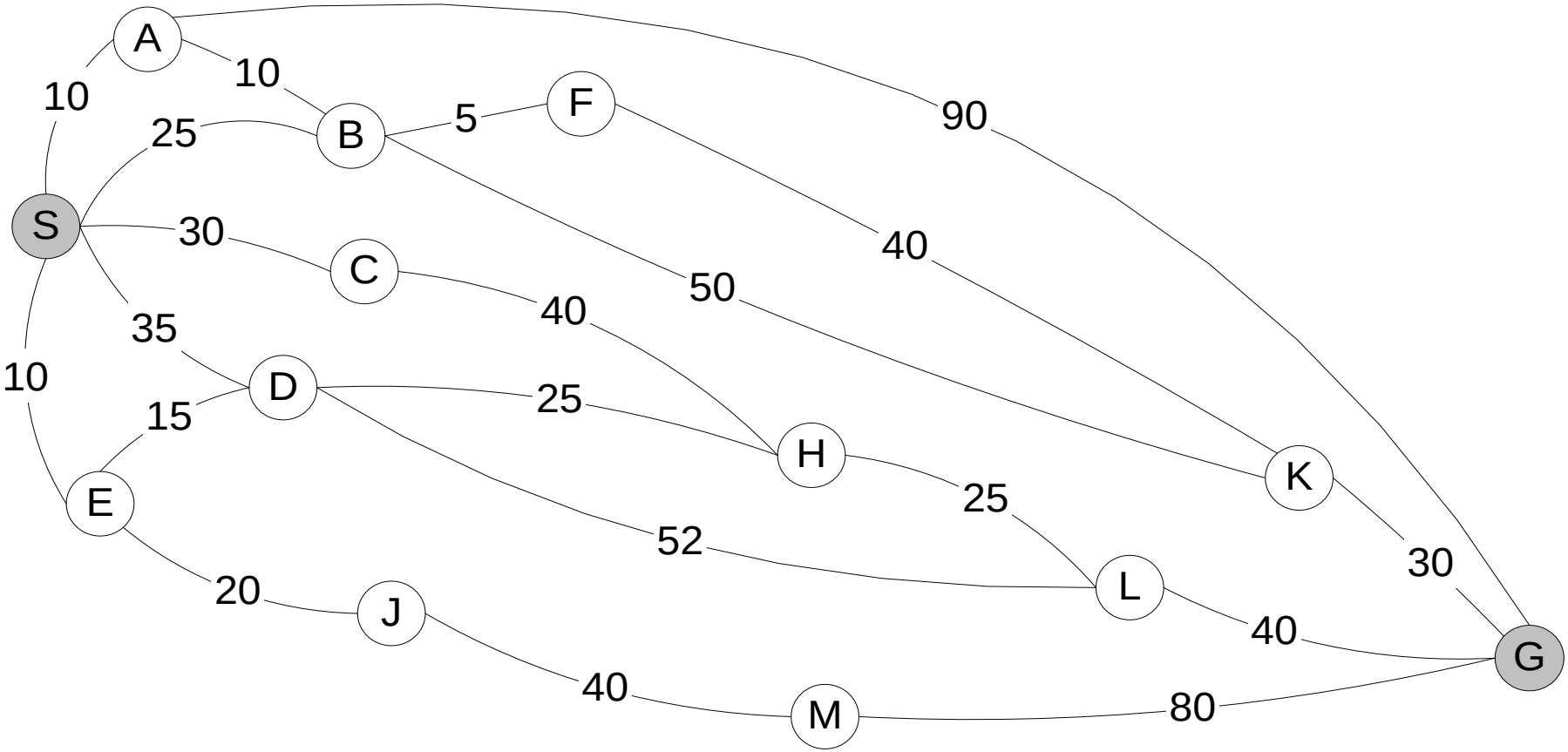


Figure 1.5: Diagram pohon dari DFS.

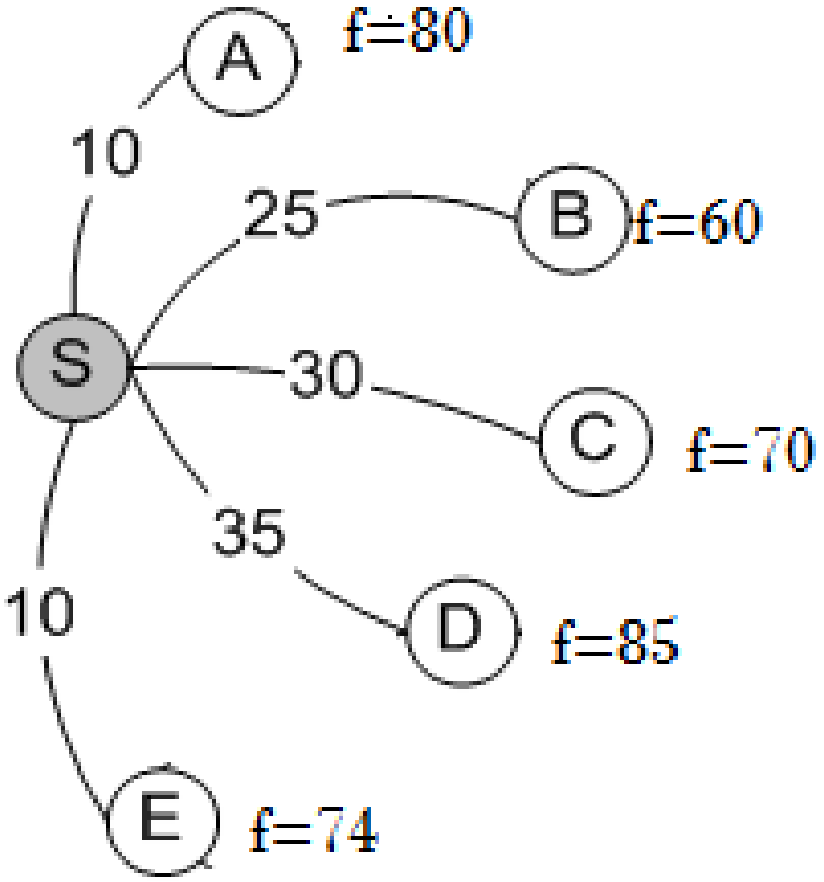
Greedy Best First Search

Greedy Best First Search



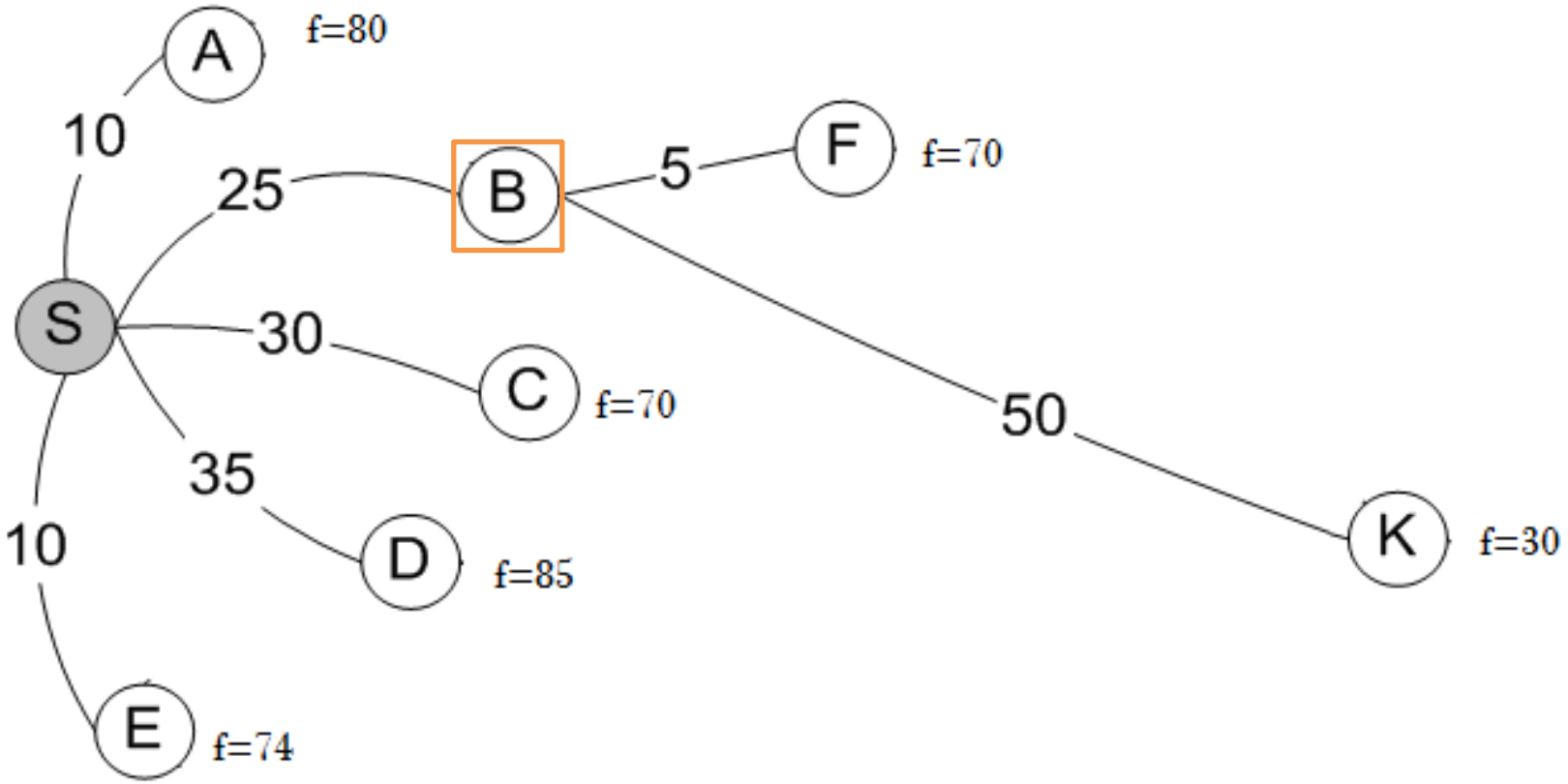
n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

Langkah 1



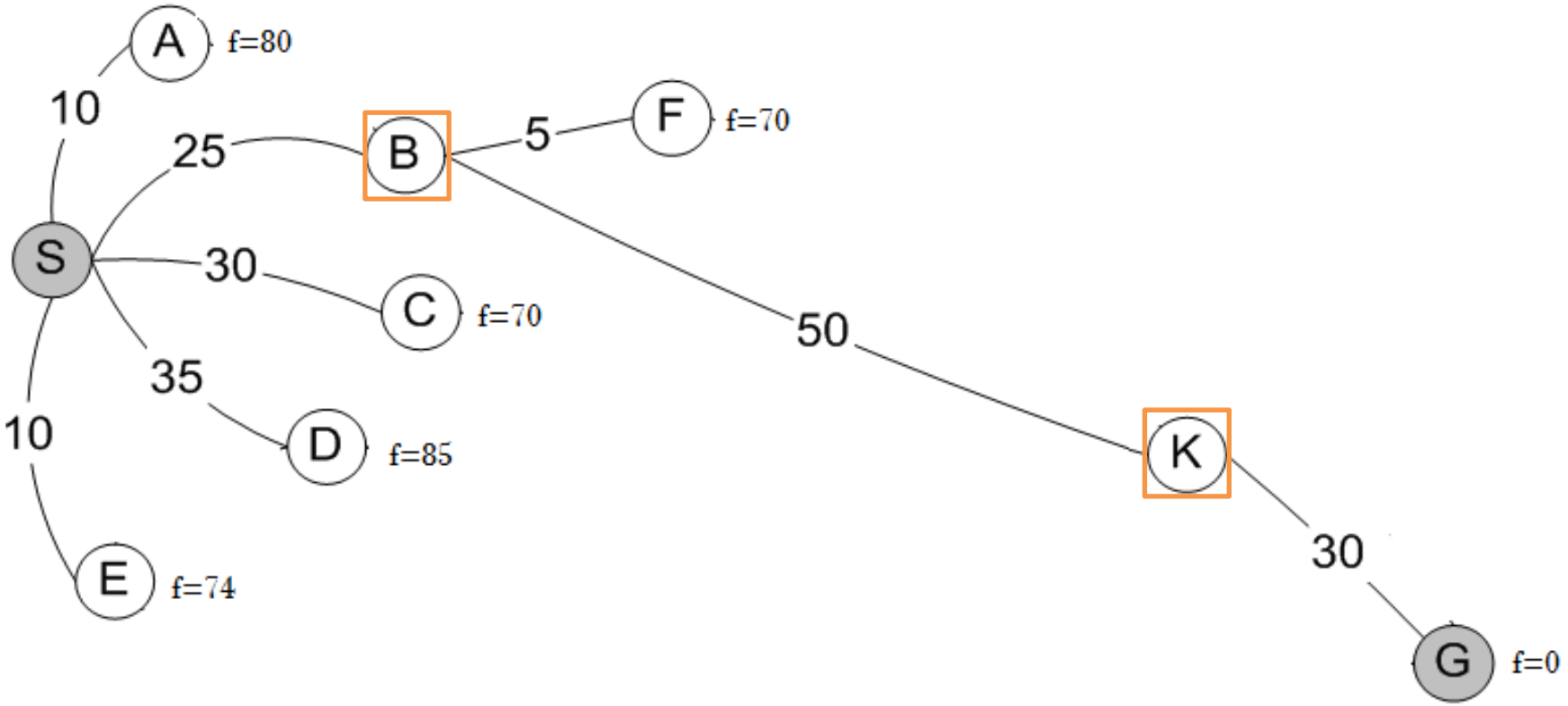
n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

Langkah 2



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

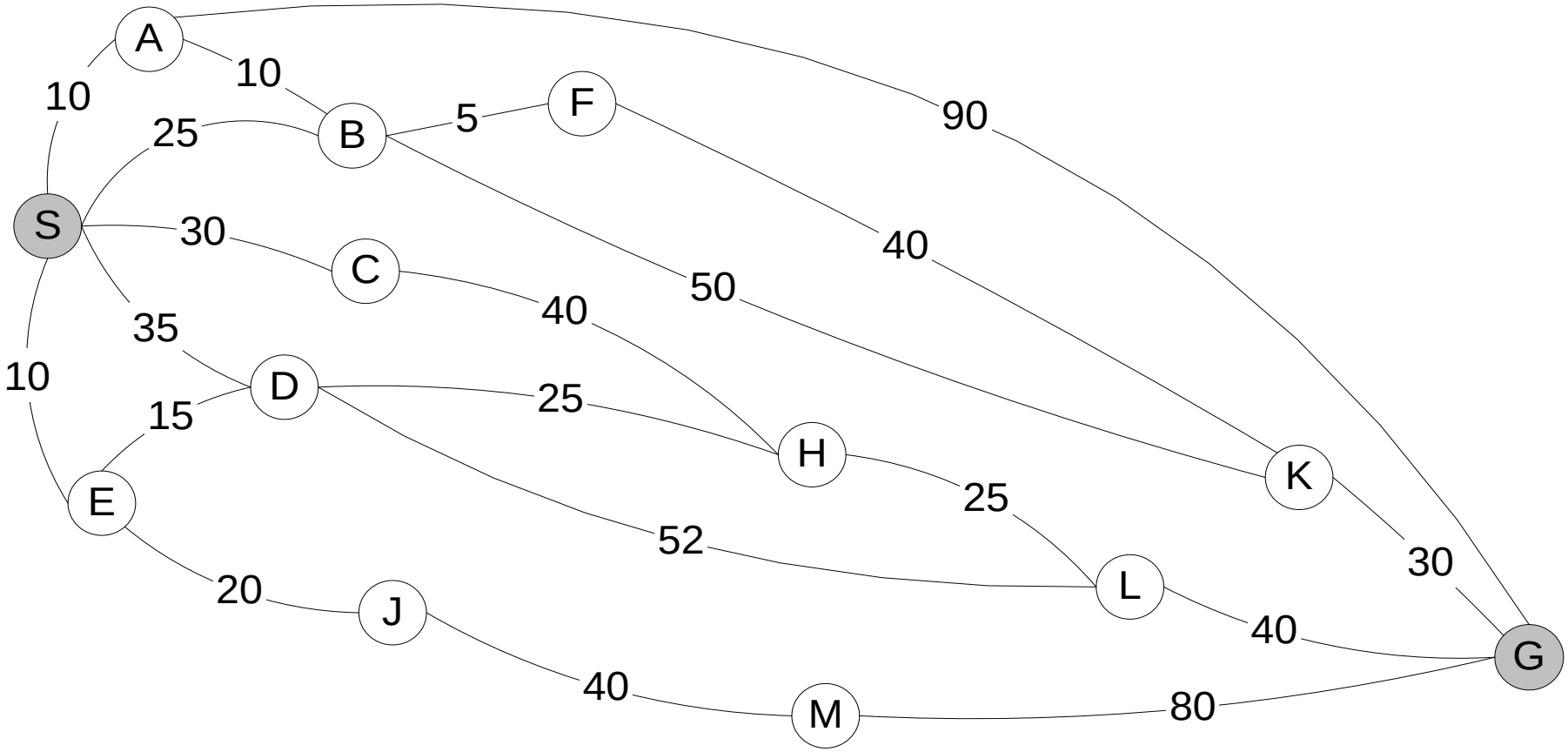
Langkah 3



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

A*

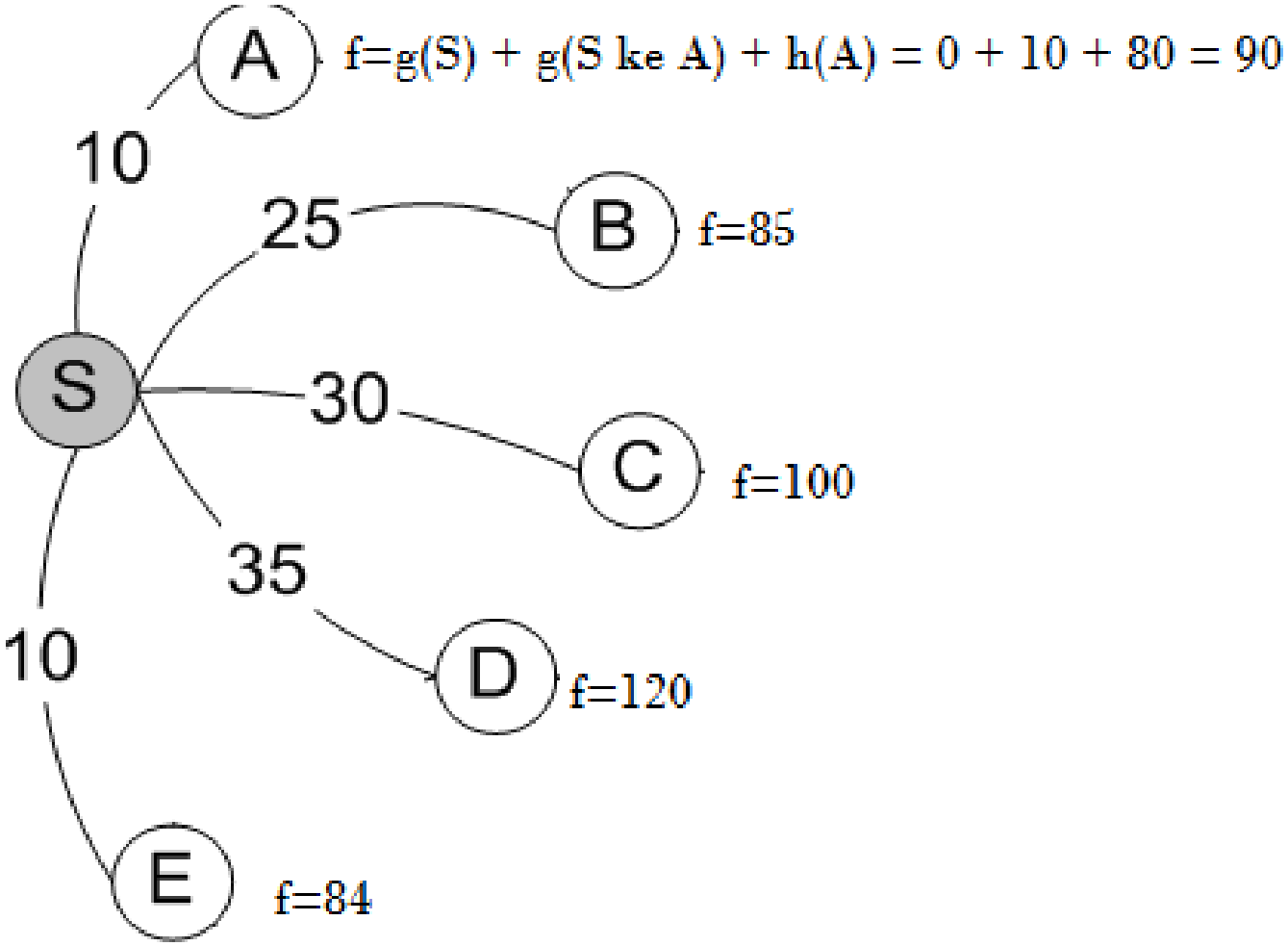
A*



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

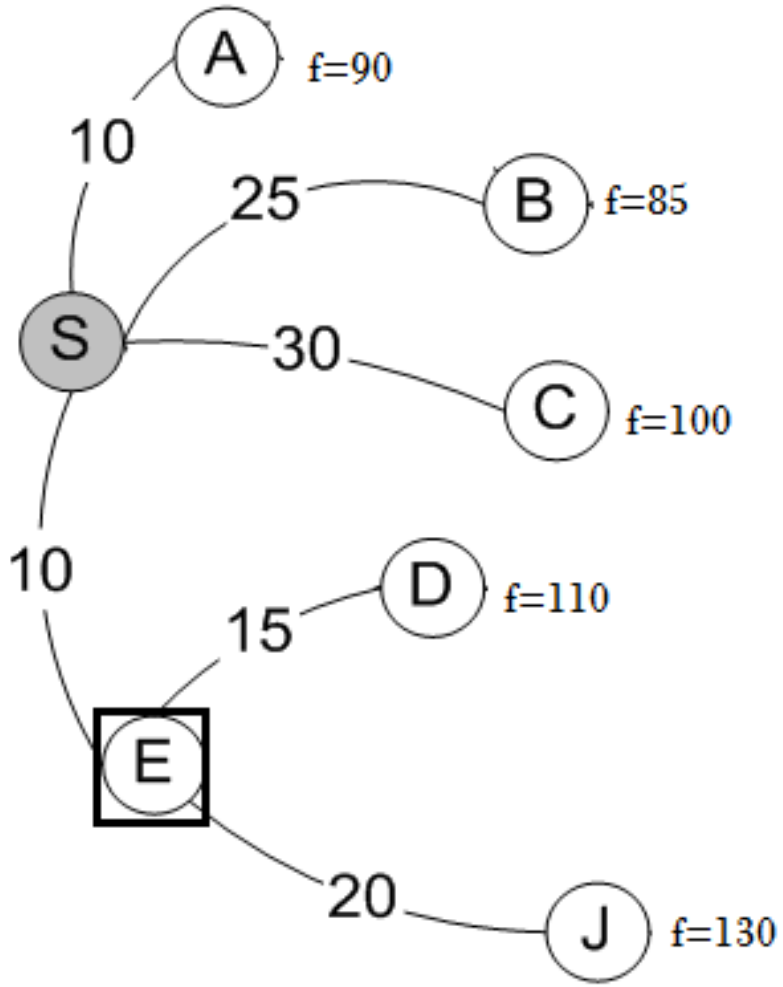
A*

Langkah 1



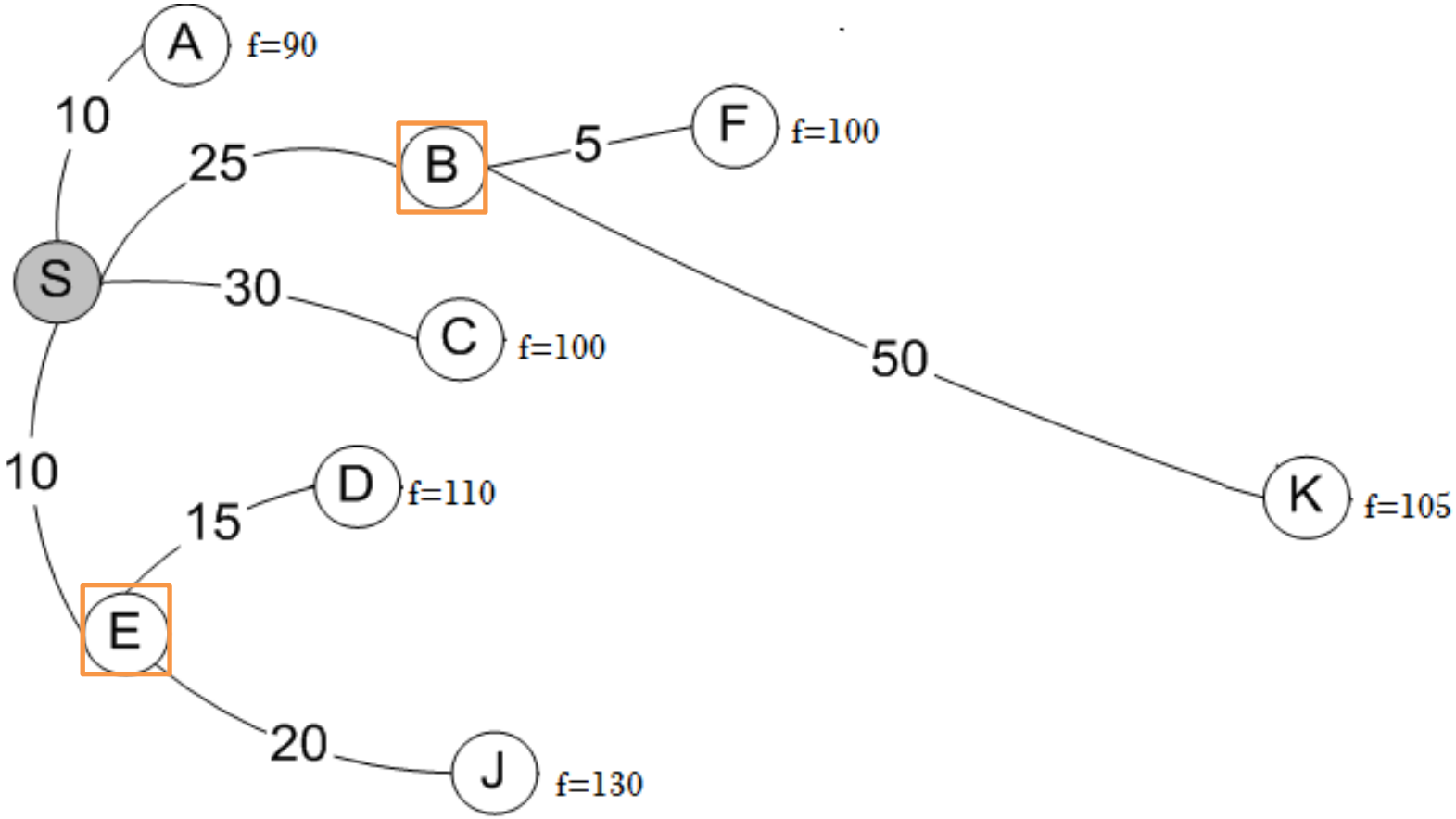
A*

Langkah 2



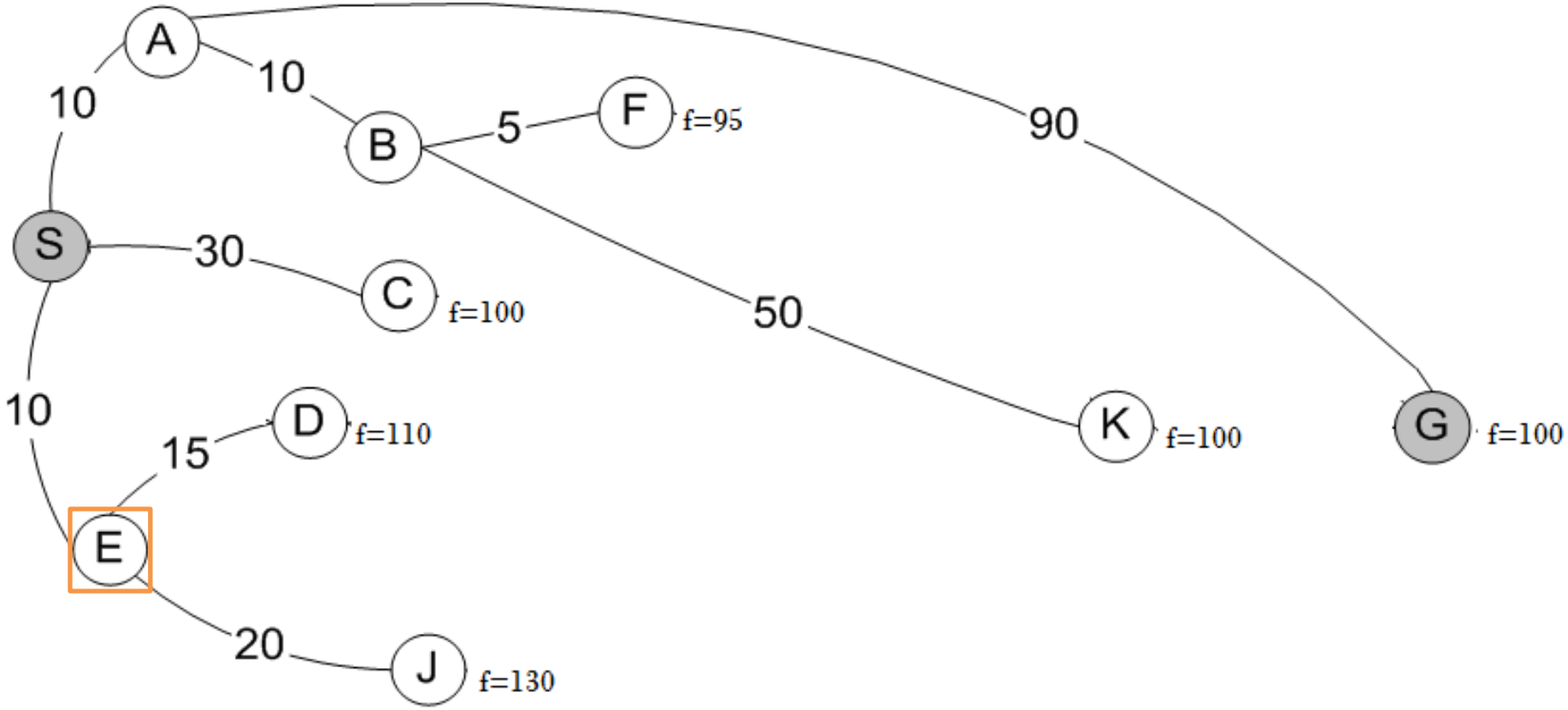
A*

Langkah 3



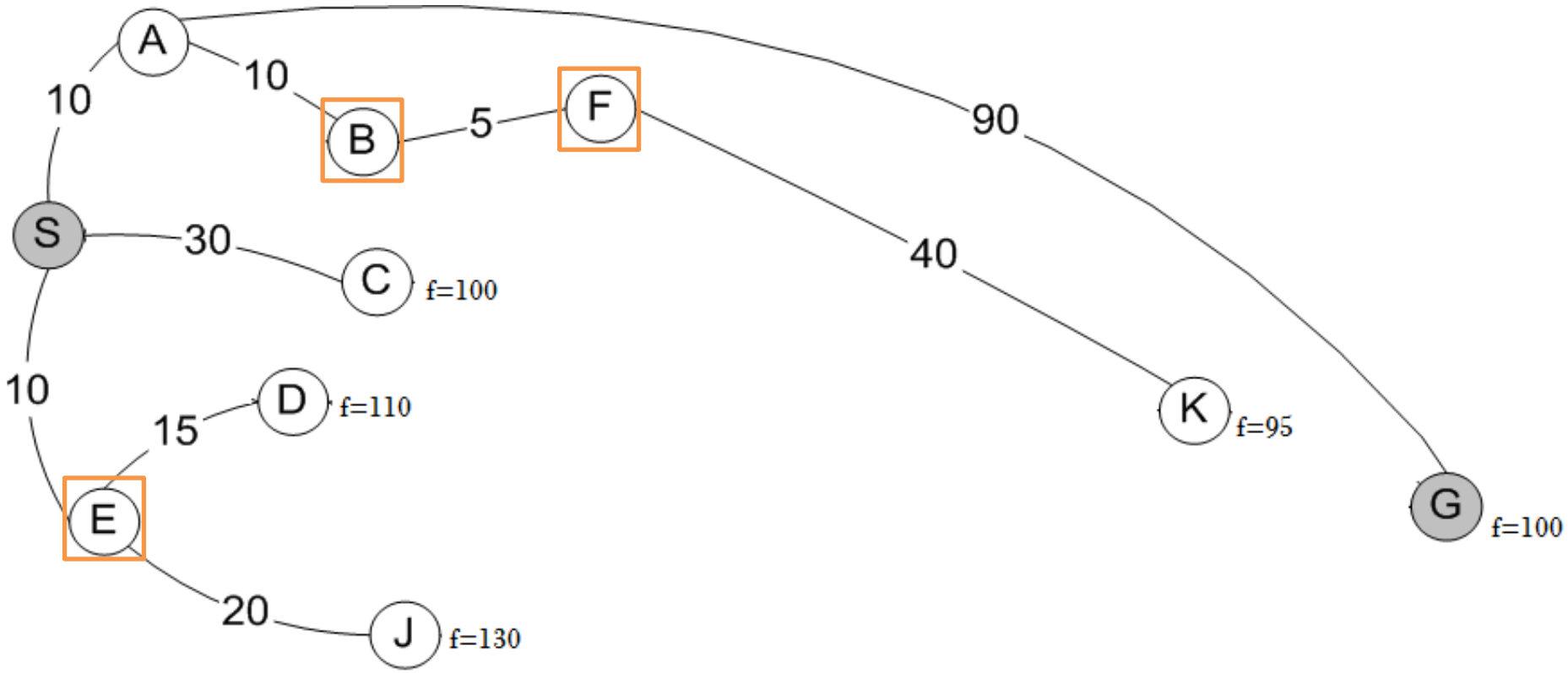
A*

Langkah 4


A
10
10
B
5
F
f=95
90
S
30
C
f=100
50
10
D
f=110
15
K
f=100
G
f=100
E
20
J
f=130

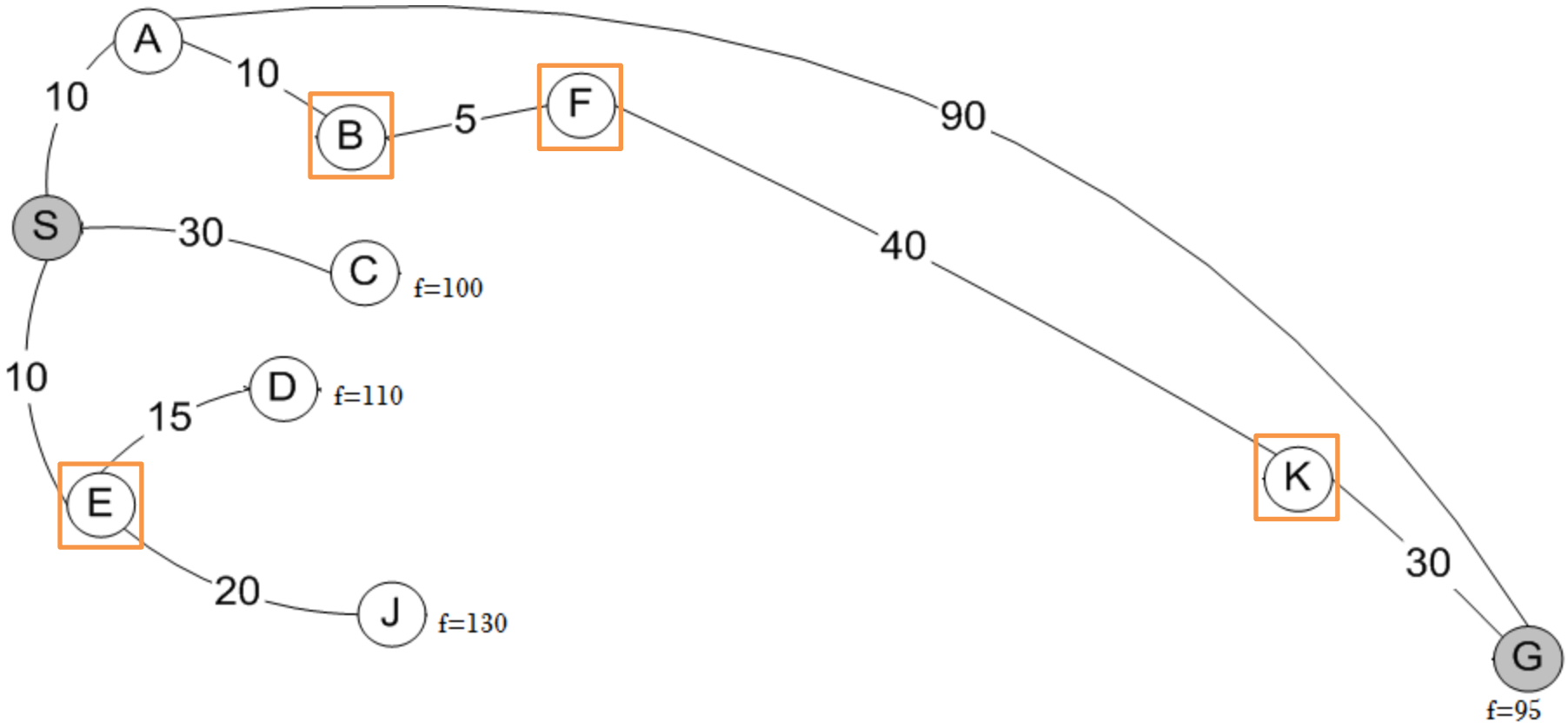
A*

Langkah 5



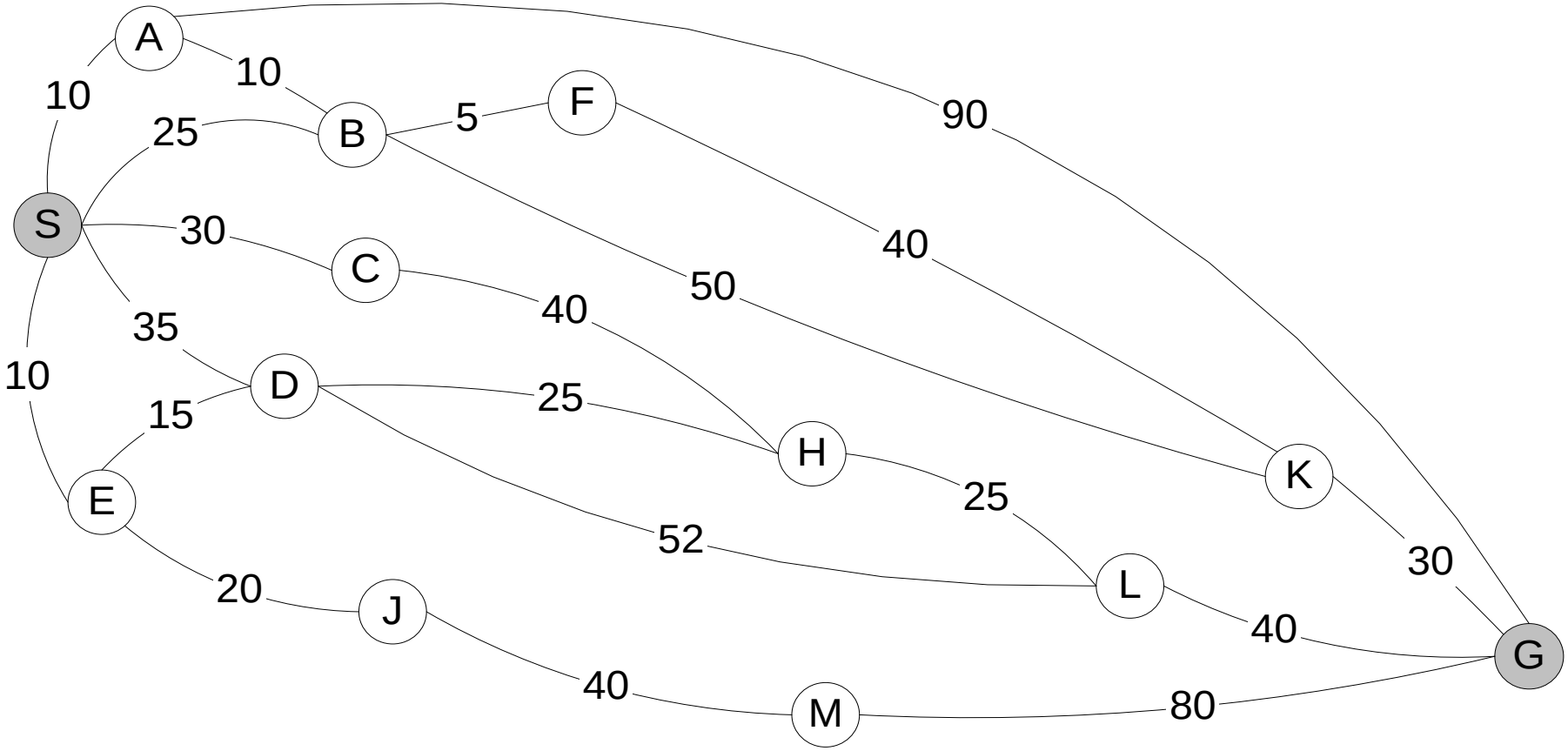
A*

Langkah 6



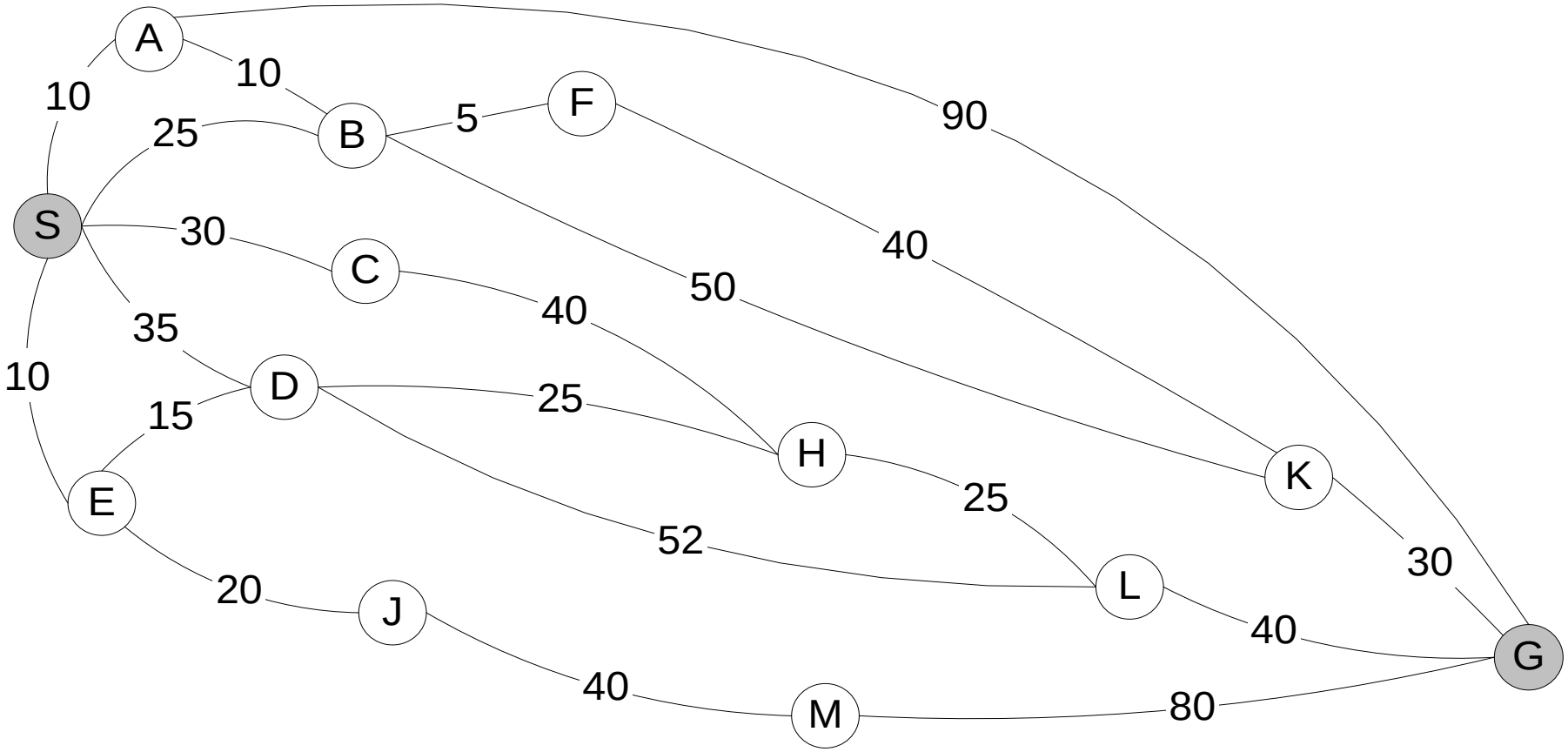
*Iterative Deepening A**
(IDA*)

IDA*



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

*IDA**



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

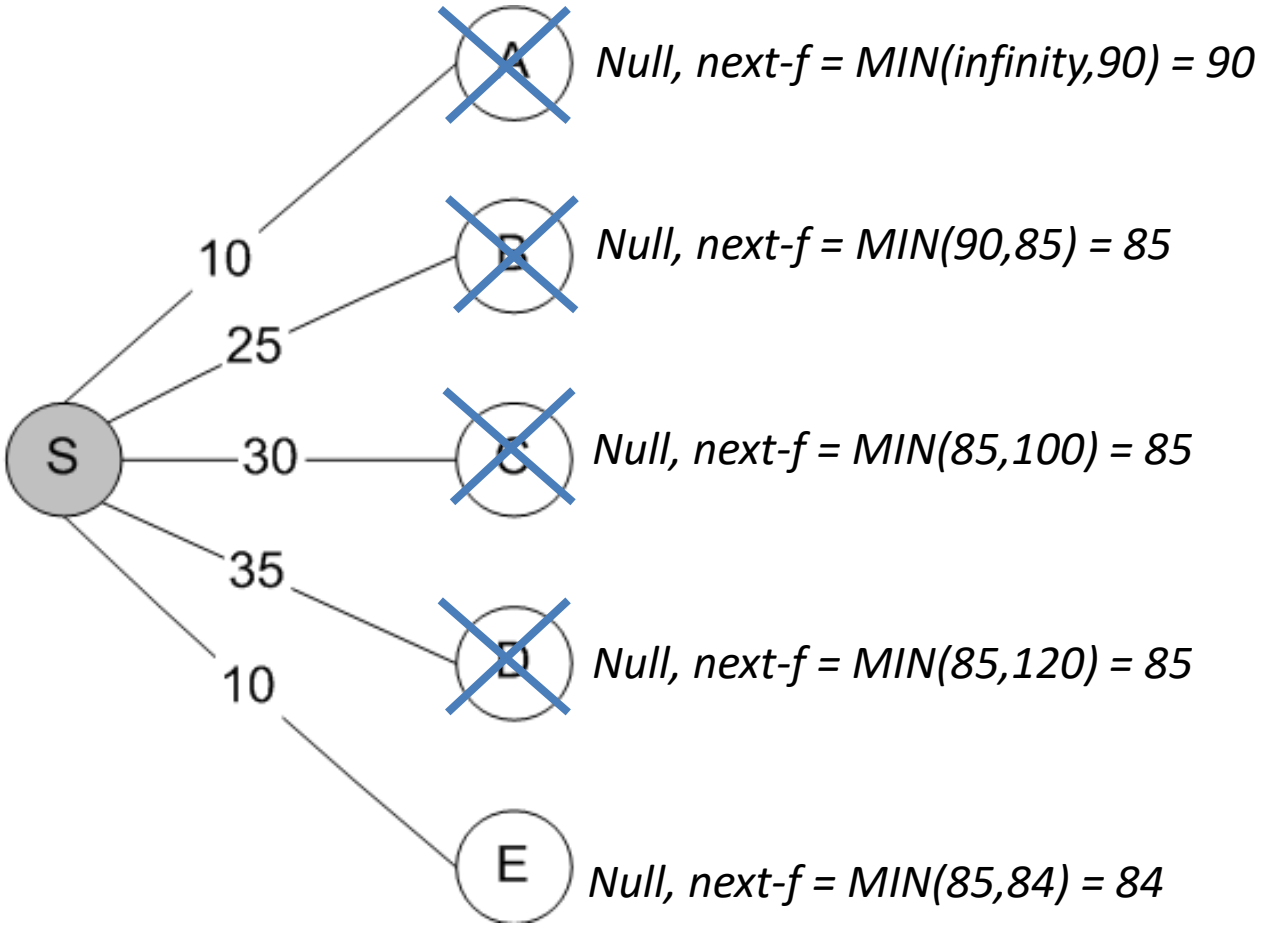
IDA*

n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

Langkah 1

Inputs :
- *Node = S*
- *F-limit = 80*

Returns :
- *Null*
- *Next-f = 84*



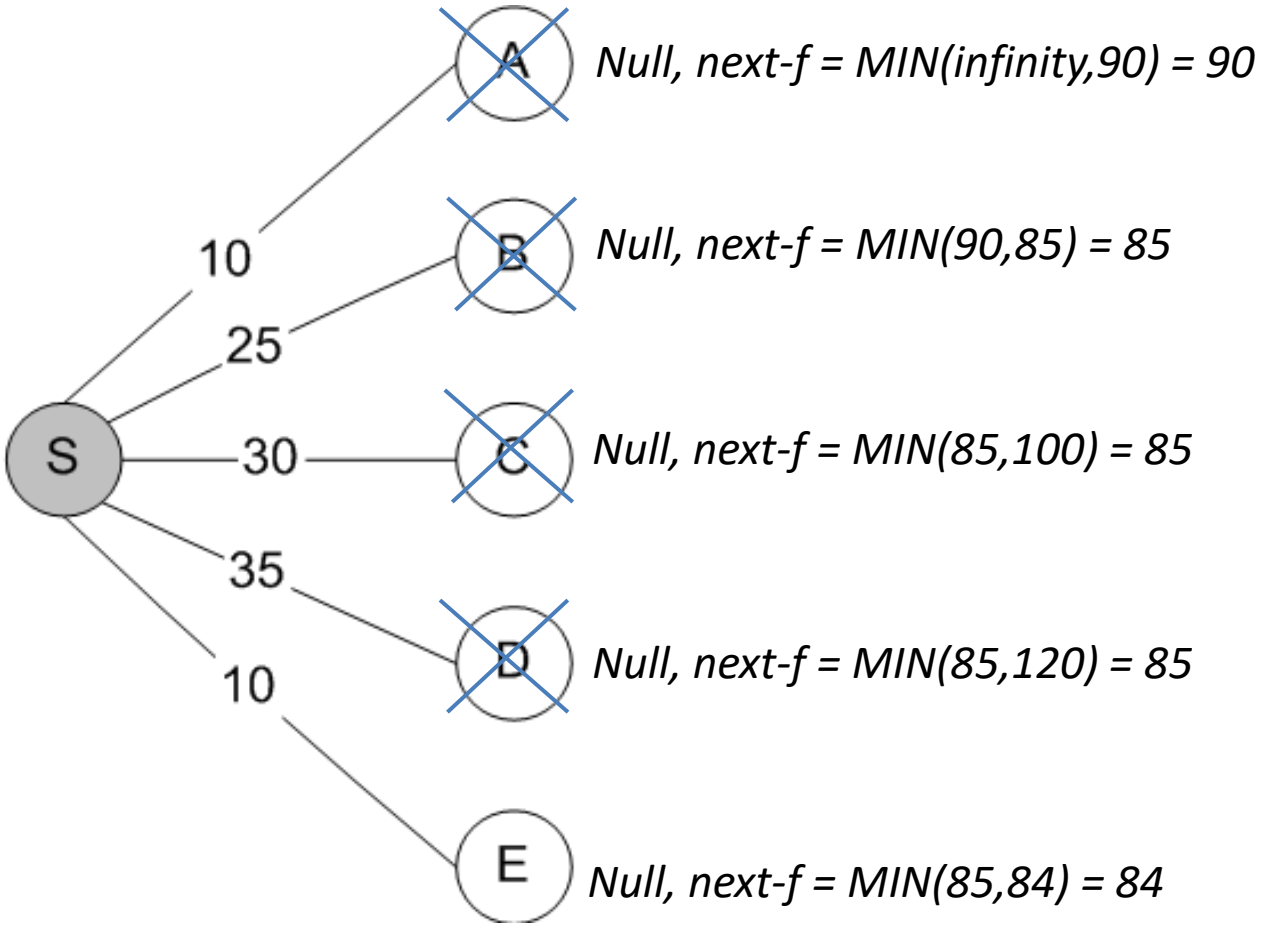
n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

DA*

Langkah 1

Inputs :
- *Node = S*
- *F-limit = 80*

Returns :
- *Null*
- *Next-f = 84*



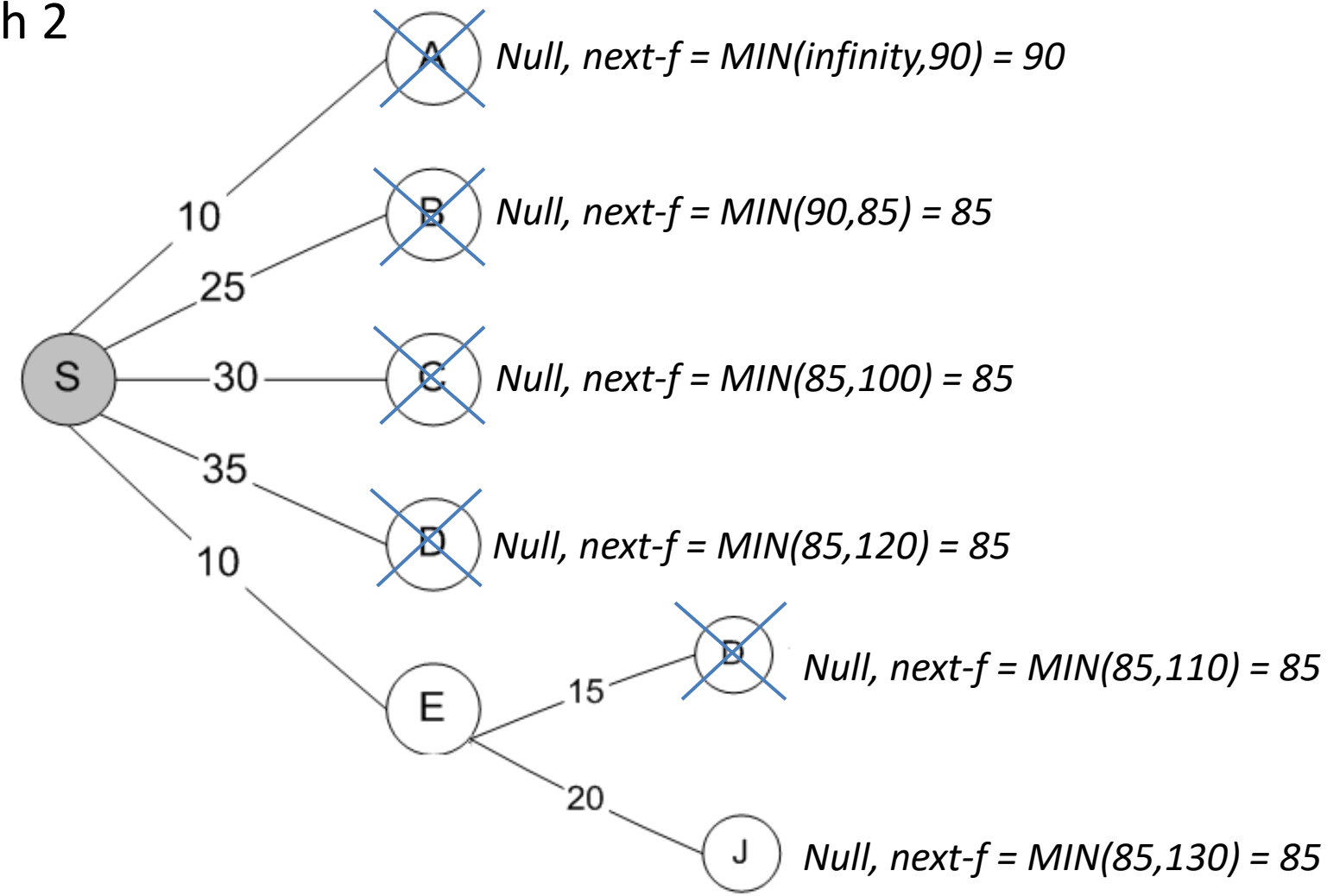
n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

IDA*

Langkah 2

Inputs :
- *Node = S*
- *F-limit = 84*

Returns :
- *Null*
- *Next-f = 85*



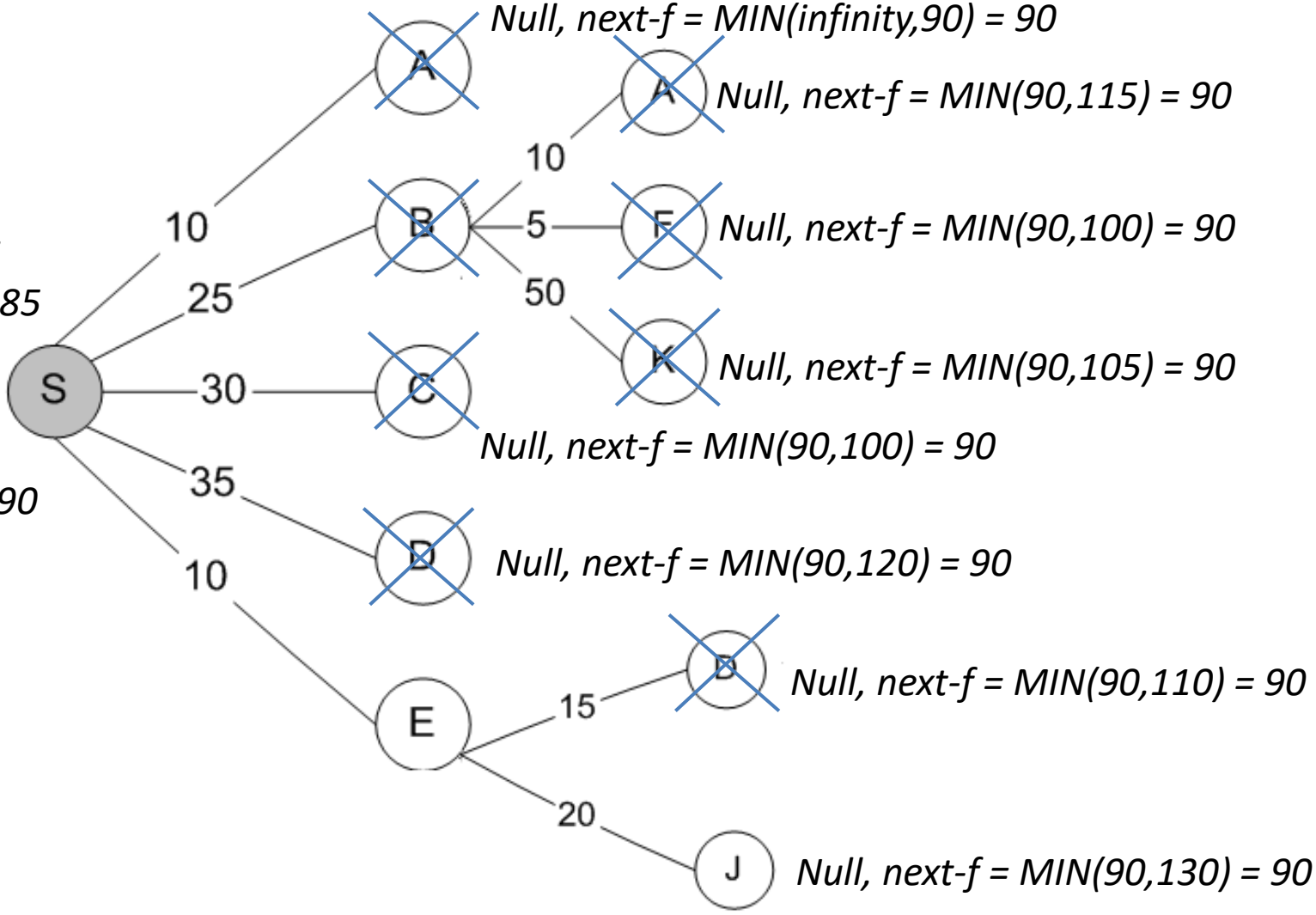
n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

IDA*

Langkah 3

Inputs :
- *Node = S*
- *F-limit = 85*

Returns :
- *Null*
- *Next-f = 90*

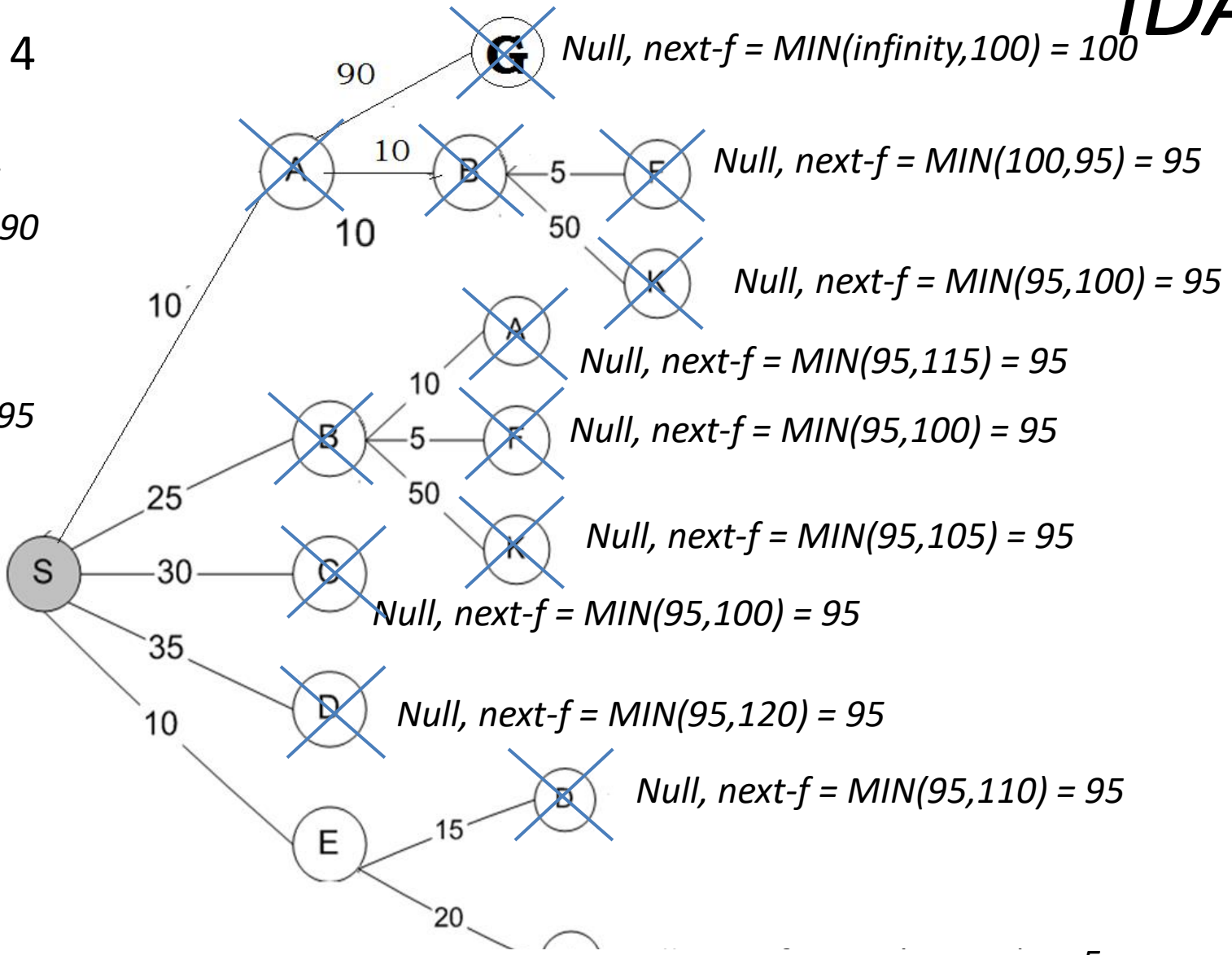


IDA*

Langkah 4

Inputs :
- *Node = S*
- *F-limit = 90*

Returns :
- *Null*
- *Next-f = 95*



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

IDA*

Langkah 5

n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

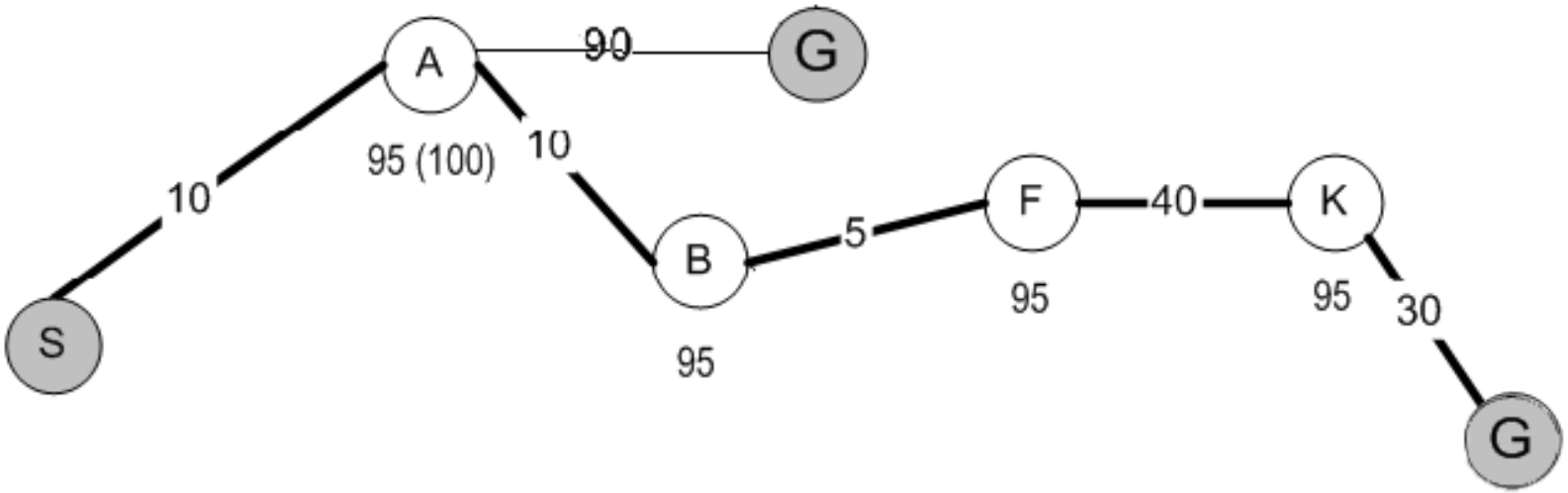
Inputs :

- *Node = S*
- *F-limit = 95*

Returns :

- *Solusi : S-A-B-F-K-G*
- *Next-f = 95*

Null. next-f = MIN(infinitv.100) = 100



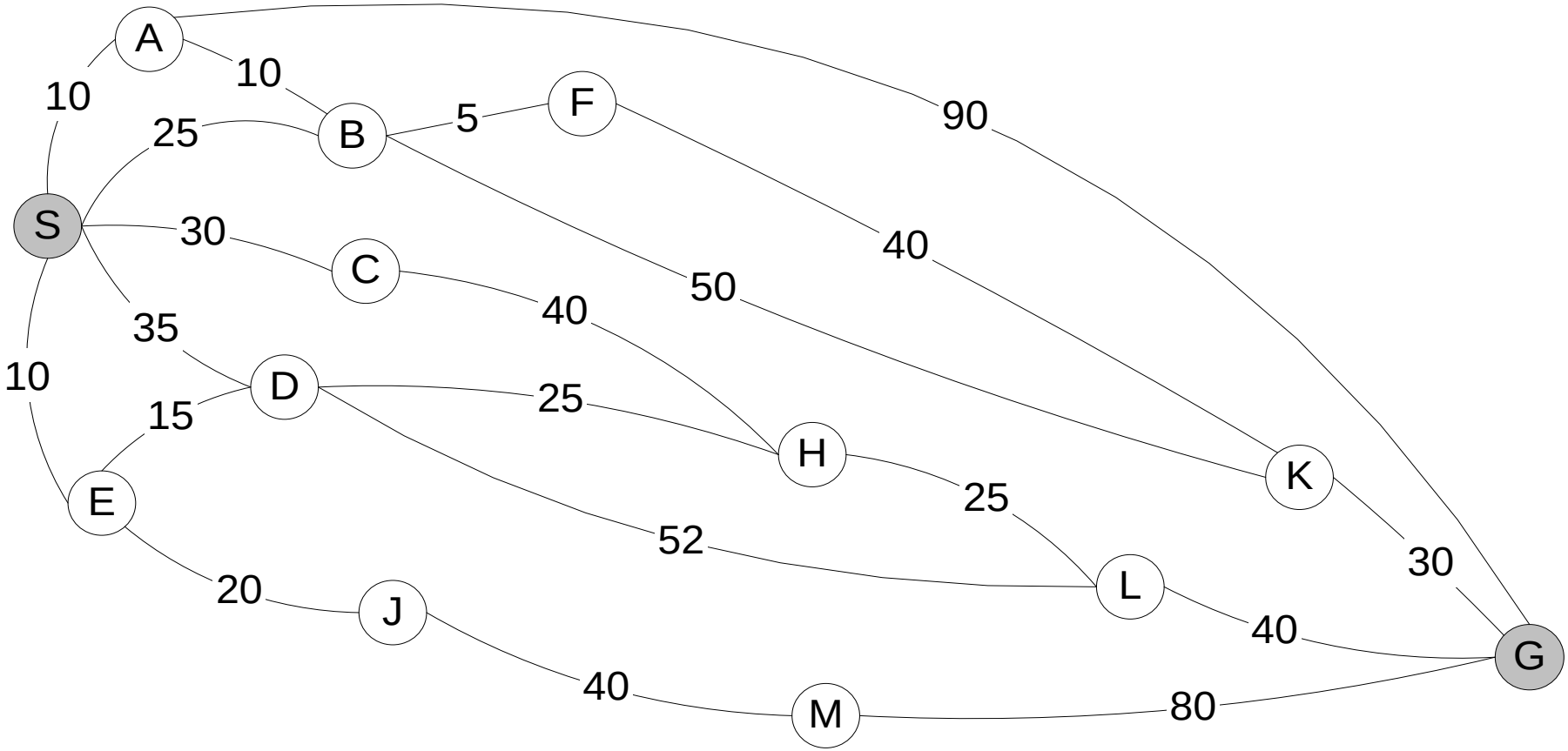
*Simplified Memory-Bounded A** (SMA*)

*Simplified Memory-Bounded A** (SMA*)

- IDA* yang hanya mengingat satu *f-limit*
- SMA* mengingat *f-Cost* dari setiap iterasi sampai sejumlah simpul yang ada di dalam memori.
- Jika memori komputer hanya mampu menyimpan 100 simpul, maka kita bisa melakukan pencarian sampai level 99.

function SMA*(*masalah*) **returns** *solusi*
inputs: *masalah*, sebuah masalah
local variables: *Queue*, antrian nodes yang terurut berdasarkan *f-cost*

Queue ← MAKE-QUEUE({MAKE-SIMPUL(*INITIAL-STATE*[*masalah*])})
***loop* do**
if *Queue* kosong **then return** gagal
n ← simpul di *Queue* yang memiliki *f-cost* terkecil dan level terdalam
if *GOAL*-TEST(*n*) **then return** sukses
s ← NEXT-SUCCESSOR(*n*)
if *s* bukan *goal* dan levelnya sudah maksimum **then**
f(*s*) ← INFINITE
else
f(*s*) ← MAX(*f*(*n*),*g*(*s*)+*h*(*s*))
end
if semua suksesor dari *n* sudah dibangkitkan **then**
Ganti *f-cost* pada *n* dengan nilai f(*s*) yang terkecil. Gantikan nilai f(*s*) terkecil ini ke semua *ancestors* dari *n* (ayah, kakek, dan seterusnya ke atas) kecuali *ancestors* yang memiliki *f-cost* lebih kecil daripada f(*s*) terkecil itu.
if semua SUCCESSORS(*n*) sudah di memori **then**
Keluarkan *n* dari *Queue* (tetapi tidak dihapus secara fisik di memori)
if memori penuh **then**
Hapus simpul terburuk yang memiliki *f-cost* terbesar dan level terdangkal (artinya: jika terdapat lebih dari satu simpul dengan *f-cost* terbesar, maka dipilih simpul yang levelnya terdangkal).
Keluarkan simpul terburuk tersebut dari daftar suksesor *parent*-nya.
Masukkan *parent* dari simpul terburuk tersebut ke *Queue* jika *parent* tersebut tidak ada di *Queue*.
end
insert *s* in *Queue*
end

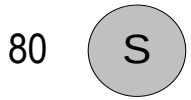


n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

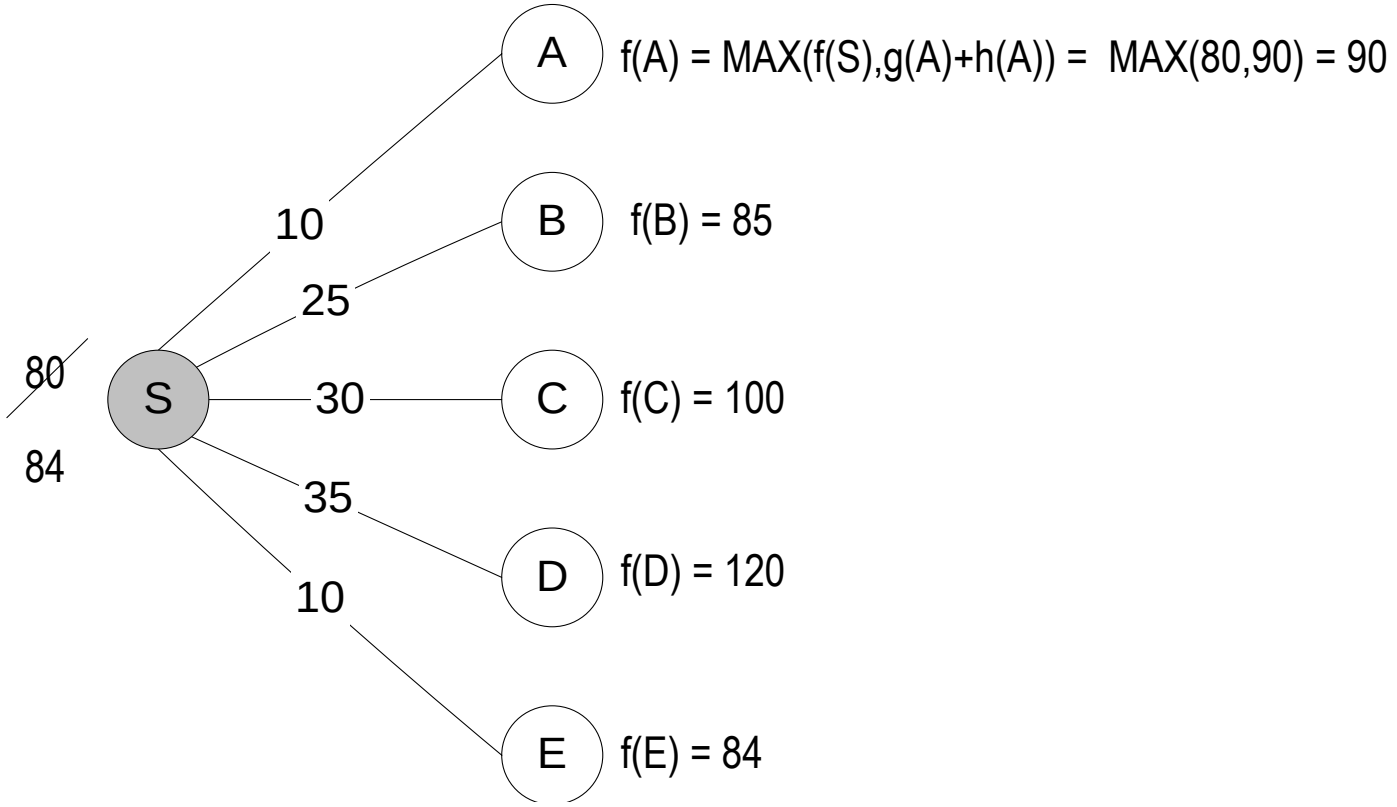
SMA*

- Pada kasus ini, misalkan memori komputer hanya mampu menyimpan 6 simpul.
- Oleh karena itu, level maksimum yang dapat dijangkau oleh SMA* adalah level 5.

Langkah 1

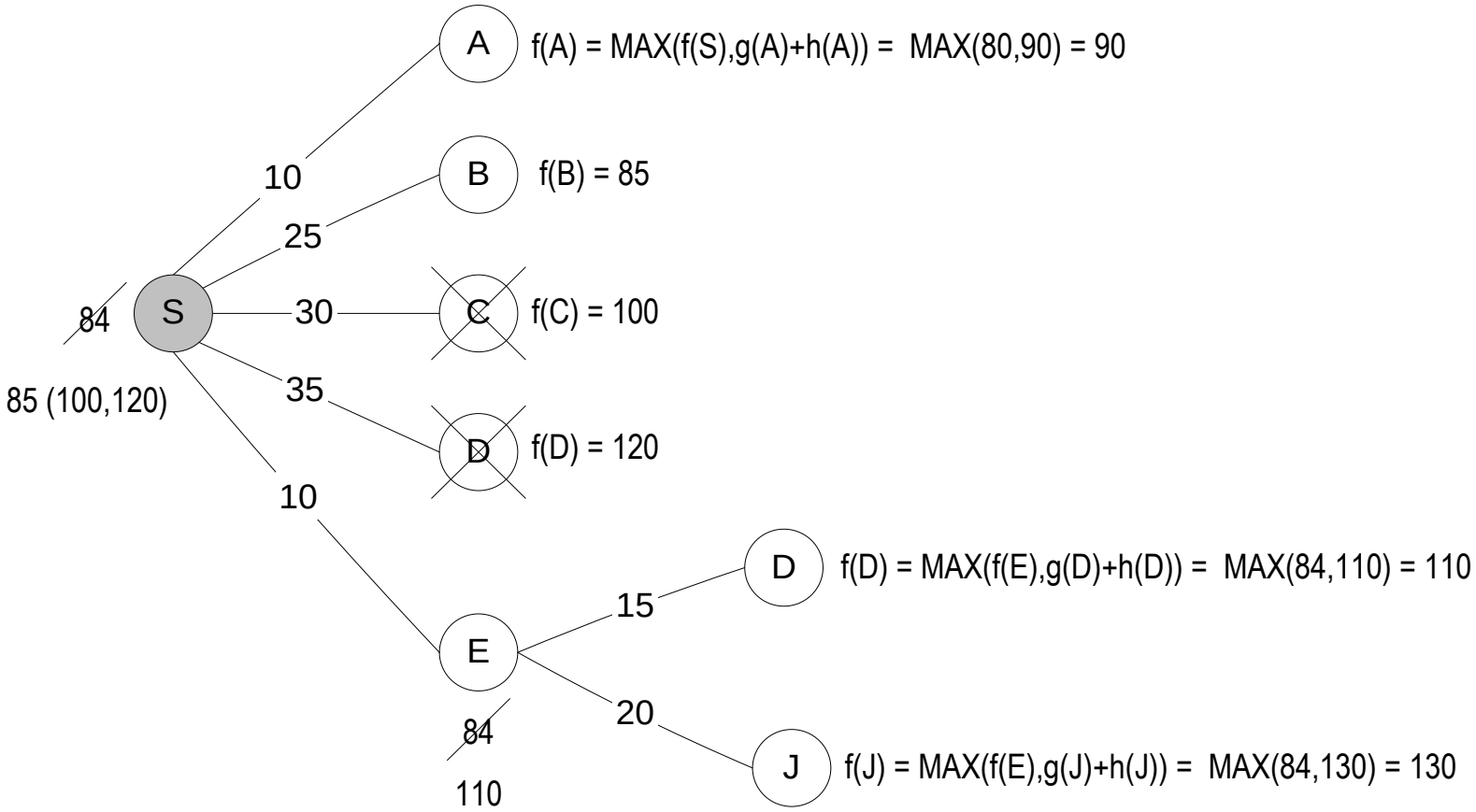


Langkah 2



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

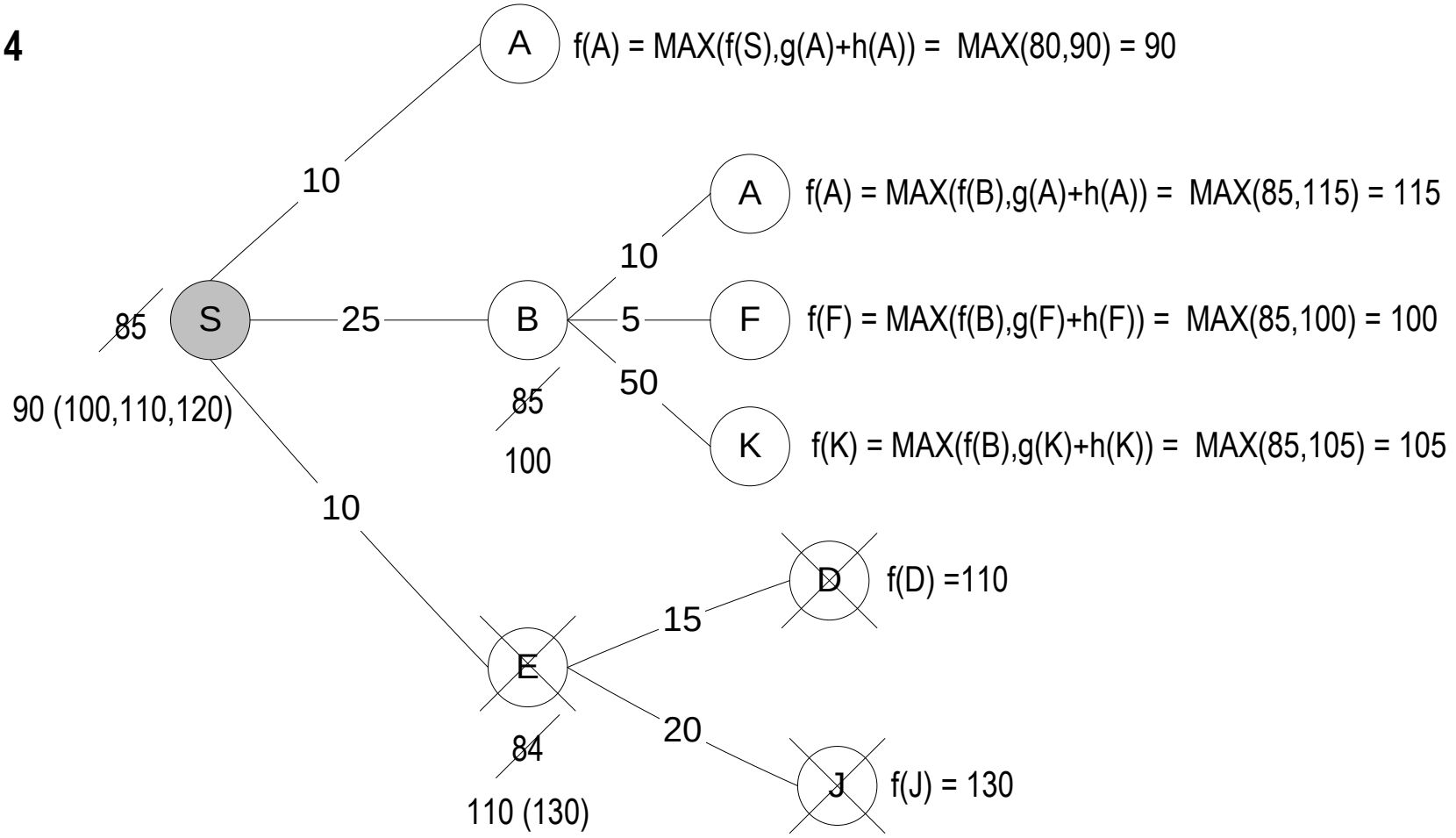
Langkah 3



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

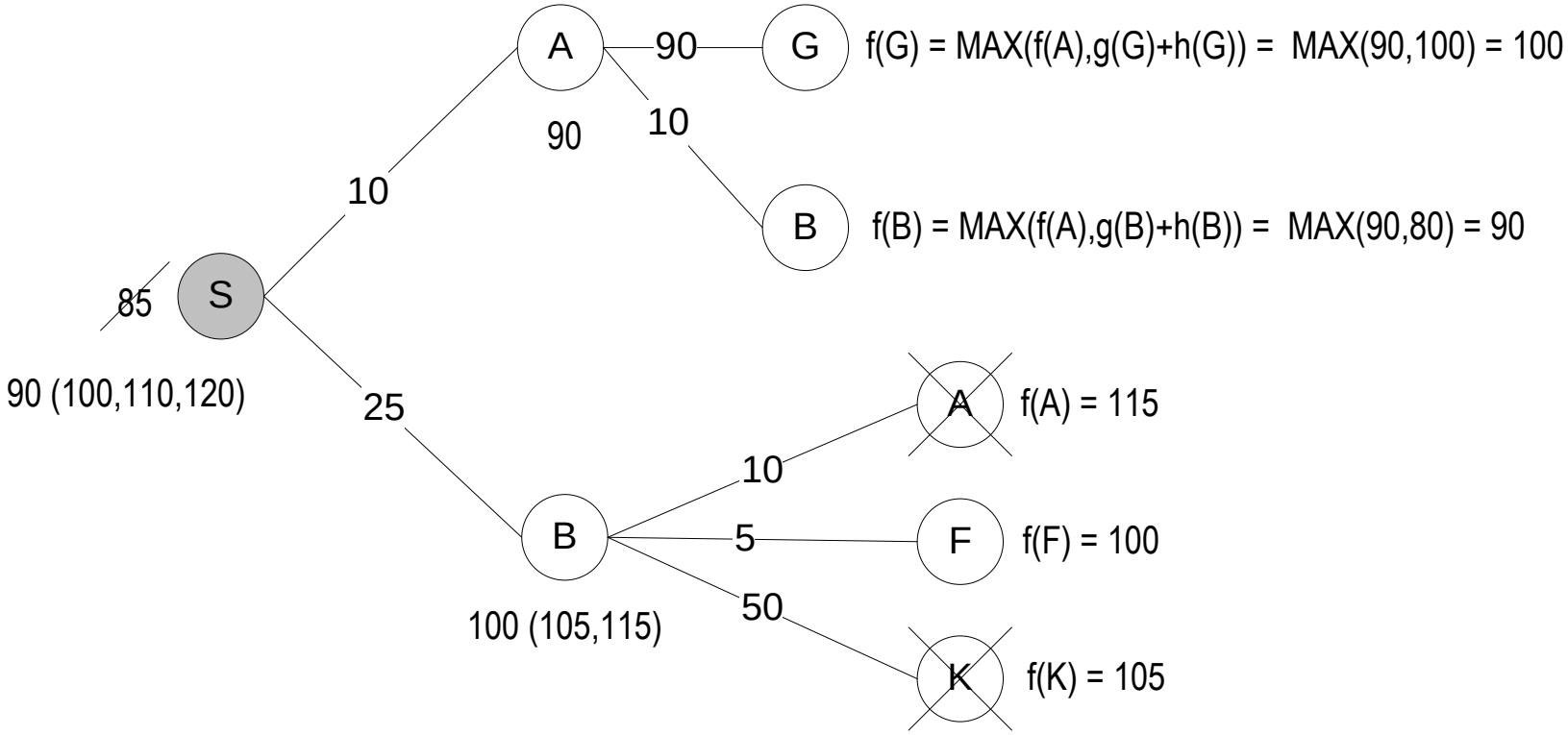
n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

Langkah 4

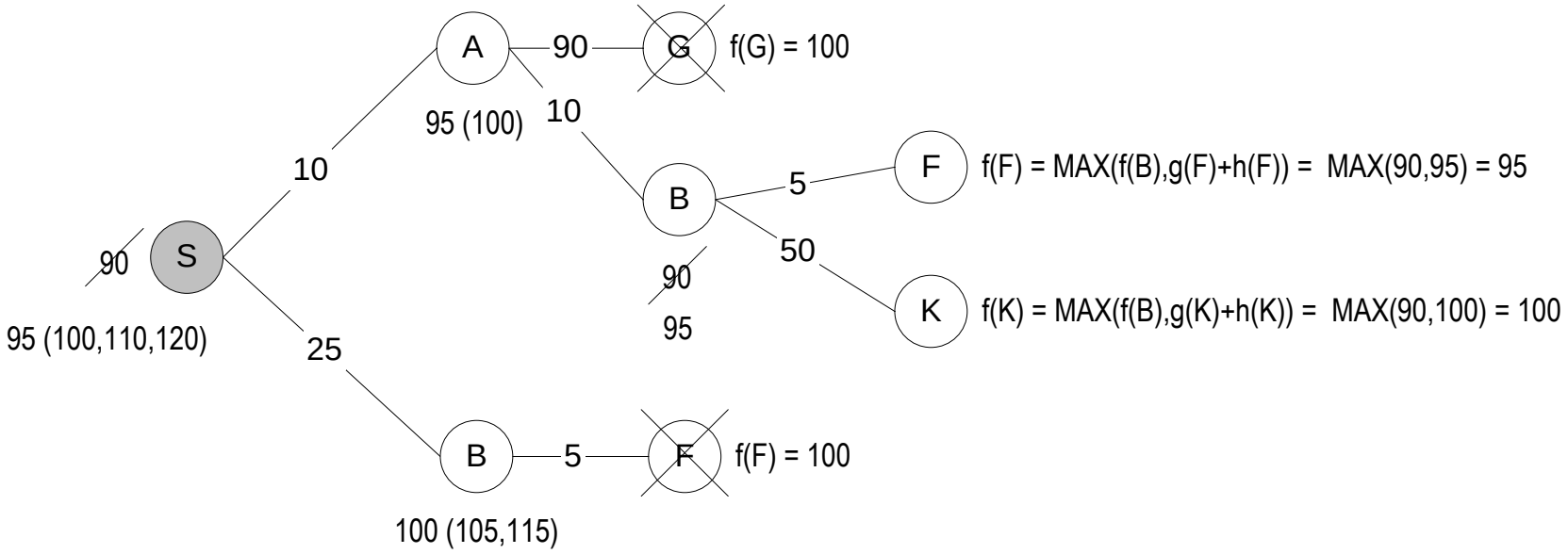


n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

Langkah 5

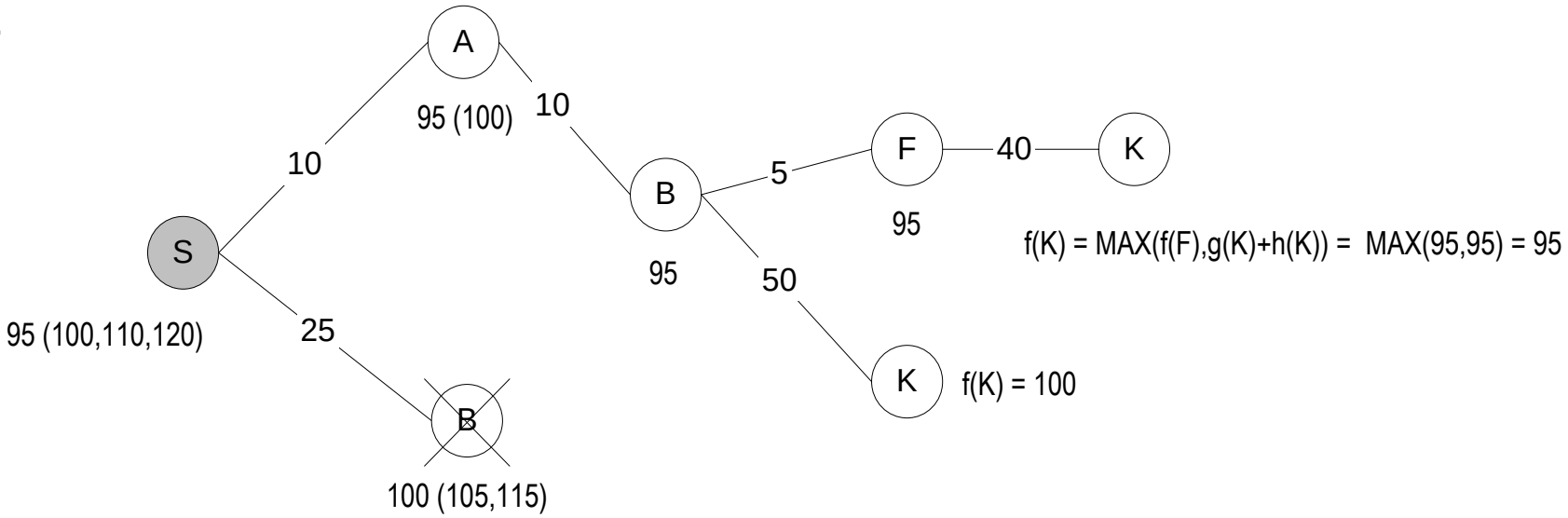


Langkah 6



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

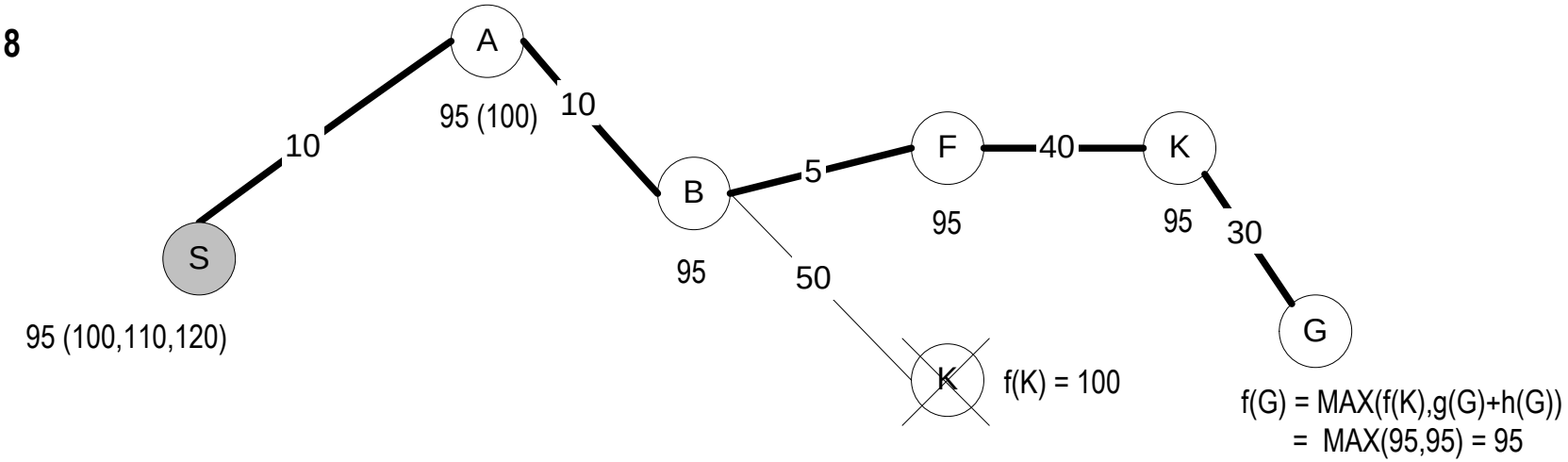
Langkah 7



n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

Langkah 8



Total Biaya = 95

Biaya optimal = 95

If you're not sure, don't guess... **ASK!**

Wrong guesses are **COSTLY!**