

Searching

(IFP30143) Sistem Cerdas
Informatika
Universitas Majalengka

Pustaka Suyanto, S.T. and Sc, M., 2014. Artificial Intelligence Searching Reasoning Planning and Learning Revisi Kedua. Penerbit Informatika, Bandung, Indonesia.

Konten

- Ruang Masalah
- Sistem Produksi
- Metode Pencarian
 - Blind/ Un-informed Search
 - Metode Pencarian Heuristik

Ruang Masalah



Teknik Pencarian

- Definisikan Ruang Masalah, Initial State, Goal State
- Definisikan Aturan Produksi
- Pilih Metode Pencarian yang Tepat

1	$(x,y) \rightarrow (4,y)$ If $x < 4$	Isi penuh jerigen 4 galon
2	$(x,y) \rightarrow (x,3)$ If $y < 3$	Isi penuh jerigen 3 galon
3	$(x,y) \rightarrow (x-d,y)$ If $x > 0$	Buang sebagian air dari jerigen 4 galon
4	$(x,y) \rightarrow (x,y-d)$ If $y > 0$	Buang sebagian air dari jerigen 3 galon
5	$(x,y) \rightarrow (0,y)$ If $x > 0$	Kosongkan jerigen 4 galon
6	$(x,y) \rightarrow (x,0)$ If $y > 0$	Kosongkan jerigen 3 galon
7	$(x,y) \rightarrow (4,y-(4-x))$ If $x+y \geq 4$ and $y > 0$	Tuangkan air dari jerigen 3 galon ke jerigen 4 galon sampai jerigen 4 galon penuh
8	$(x,y) \rightarrow (x,(3-x),3)$ If $x+y \geq 3$ and $x > 0$	Tuangkan air dari jerigen 4 galon ke jerigen 3 galon sampai jerigen 3 galon penuh
9	$(x,y) \rightarrow (x+y,0)$ If $x+y \leq 4$ and $y > 0$	Tuangkan seluruh air dari jerigen 3 galon ke jerigen 4 galon
10	$(x,y) \rightarrow (0,x+y)$ If $x+y \leq 3$ and $x > 0$	Tuangkan seluruh dari jerigen 4 galon ke jerigen 3 galon
11	$(0,2) \rightarrow (2,0)$	Tuangkan seluruh air dari jerigen 3 galon ke jerigen 4 galon
12	$(2,y) \rightarrow (0,y)$	Buang 2 galon air dalam jerigen 3 galon ke jerigen 4 galon sampai habis

Jumlah air dalam jerigen 4 galon	Jumlah air dalam jerigen 3 galon	Aturan produksi yang diaplikasikan
0	0	-
0	3	2
3	0	9
3	3	2
4	2	7

Sistem Produksi

- Himpunan Aturan
- Pengetahuan
- Strategi Kontrol
 - Cause Motion
 - Systematic
- Pengaplikasian Aturan

Metode Pencarian

- Blind/ Un-informed Search
- Metode Pencarian Heuristik

Ukuran Performa :

- Completeness
- Time complexity
- Space complexity
- Optimality

Blind/ Un-informed Search

- Breadth First Search (BFS)
- Uniform Cost Search (UCS)
- Depth First Search (DFS)
- Depth-Limited Search (DLS)
- Iterative-Deepening Search (IDS)
- Bi-directional search (DBS)

Metode Pencarian Heuristik

- Generate and Test
- Hill Climbing
 - Simple HC
 - Steepest Ascent HC
- Simulated Annealing
- Best First Search
 - Greedy Best First Search
 - A*

Blind/ Un-informed Search

(IFP30143) Sistem Cerdas

Informatika

Universitas Majalengka

Breadth First Search (BFS)

- Pencarian dari Kiri ke Kanan
- Complete dan Optimal
- Membangkitkan Semua Simpul
 - b = Faktor Percabangan
 - d = Kedalaman Solusi
 - $O(b^d)$

Breadth First Search (BFS)

$$b = 10 \quad \& \quad d = 8$$

$$10^0 + 10^1 + 10^2 + 10^3 + 10^4 + 10^5 + 10^6 + 10^7 + 10^8 \\ = 111.111.111.111 \approx 10^8$$

$$d = 14 \Rightarrow 10^{15}$$

Depth First Search (DFS)

$$b = 10 \quad \& \quad d = 3$$

$$\text{DFS} \Rightarrow 1 + 10 + 10 + 10 = 31$$

$$\text{BFS} \Rightarrow 1 + 10 + 100 + 1000 = 1.111$$

Depth-Limited Search (DLS)

Uniform Cost Search (UCS)

- BFS => Level
- UCS => Biaya /Jarak
- Biaya Terendah
- Biaya => $g(n)$

Iterative-Deepening Search (IDS)

IDS = Kelebihan BFS(Complete+Optimal)+Kelebihan DFS (Space Complexity Rendah)

but

Time Complexity menjadi Tinggi

Bi-directional search (DBS)

- Pencarian Maju dan Pencarian Mundur
- Solusi ditemukan jika Simpul yang Sama Dibangkitkan dari Ke Dua Arah
- BFS $\Rightarrow O(b^d)$
- DBS $\Rightarrow O(b^{d/2}) \approx O(b^{d/2})$
- $b=10$ & $d=6$
- BFS $\Rightarrow 1.111.111$
- DBS $\Rightarrow 2.222$

Bi-directional search (DBS)

- Pencarian Mundur?
- Aturan Produksi dibalik?
- Harus selalu diuji!

Metode Pencarian Heuristik

(IFP30143) Sistem Cerdas

Informatika

Universitas Majalengka

Generate and Test

Hill Climbing

Simulated Annealing

BEST FIRST SEARCH

- Merupakan kombinasi kelebihan teknik *depth first search* dan *breadth first search*
- Pencarian diperkenankan mengunjungi node yang ada di level yg lebih rendah jika ternyata node pada level yg lebih tinggi ternyata memiliki nilai heuristik yg buruk

Best First Search

- Best First Search akan membangkitkan node berikutnya dari semua node yg pernah dibangkitkan
- Pertanyaannya :

Bagaimana menentukan sebuah node terbaik saat ini?

Dilakukan dengan menggunakan biaya perkiraan

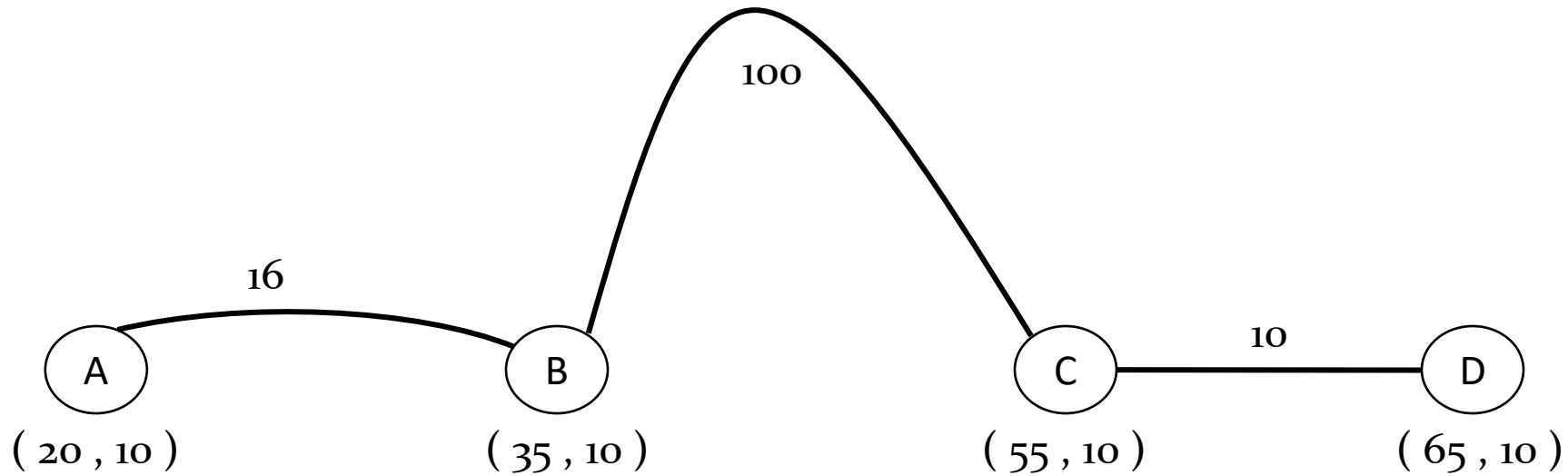
Bagaimana caranya menentukan biaya perkiraan?

Biaya perkiraan dapat ditentukan dengan fungsi heuristic

FUNGSI HEURISTIC

- Suatu fungsi heuristic dikatakan baik jika bisa memberikan biaya perkiraan yang mendekati biaya sebenarnya.
- Semakin mendekati biaya sebenarnya, fungsi heuristic tersebut semakin baik.

Contoh



Dalam kasus pencarian rute terpendek, biaya sebenarnya adalah panjang jalan Raya yang sebenarnya. Sedangkan fungsi heuristiknya adalah garis lurus dari 1 kota ke kota lainnya. Untuk itu, bisa digunakan rumus berikut :

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



$$d_{AB} = 15$$

$$d_{BC} = 20$$

$$d_{CD} = 10$$

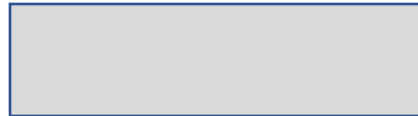
Algoritma Best First Search

- Greedy Best First Search
- Algoritma A*

Greedy Best First Search

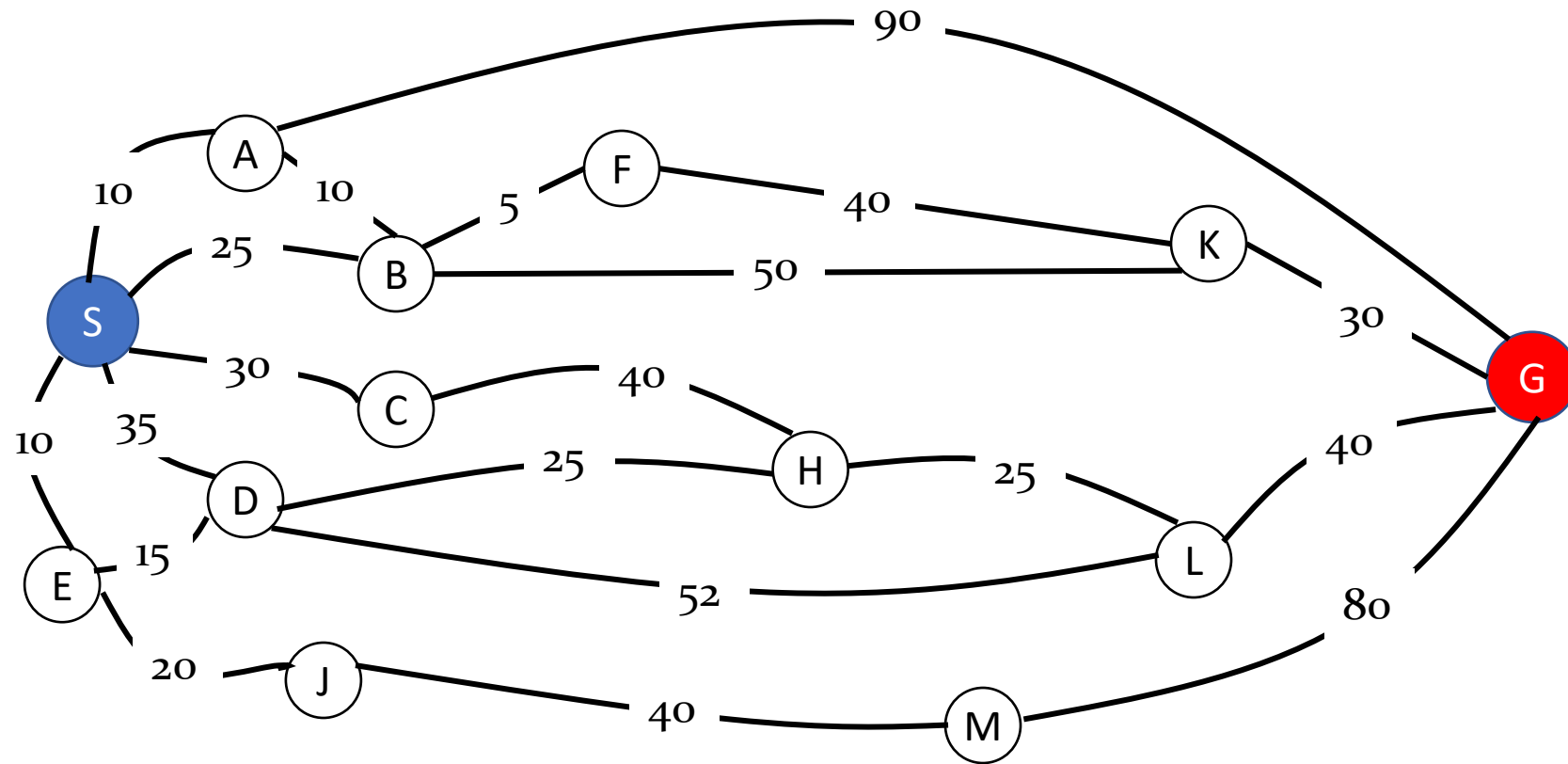
- Algoritma ini merupakan jenis algoritma Best First Search yg paling sederhana
- Algoritma ini hanya memperhitungkan biaya perkiraan saja

$$f(n) = h'(n)$$



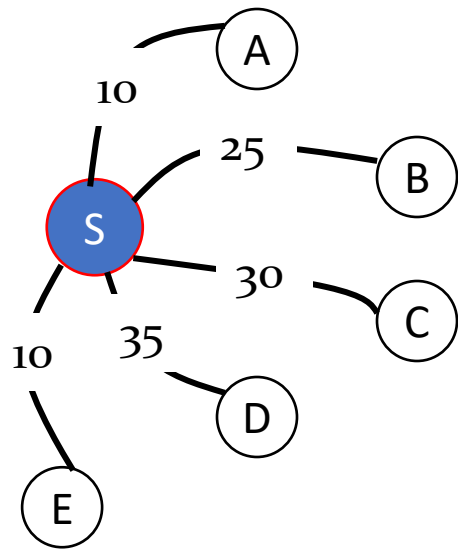
- Karena hanya memperhitungkan biaya perkiraan yang belum tentu kebenarannya, maka algoritma ini menjadi **tidak optimal**

Contoh



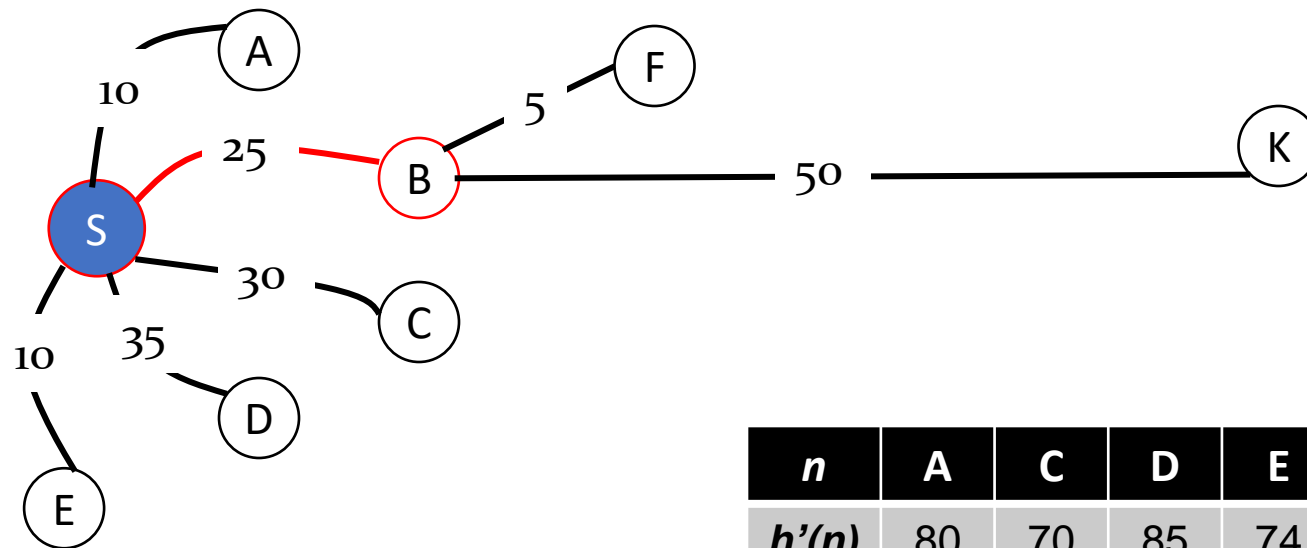
n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h'(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

Langkah 1



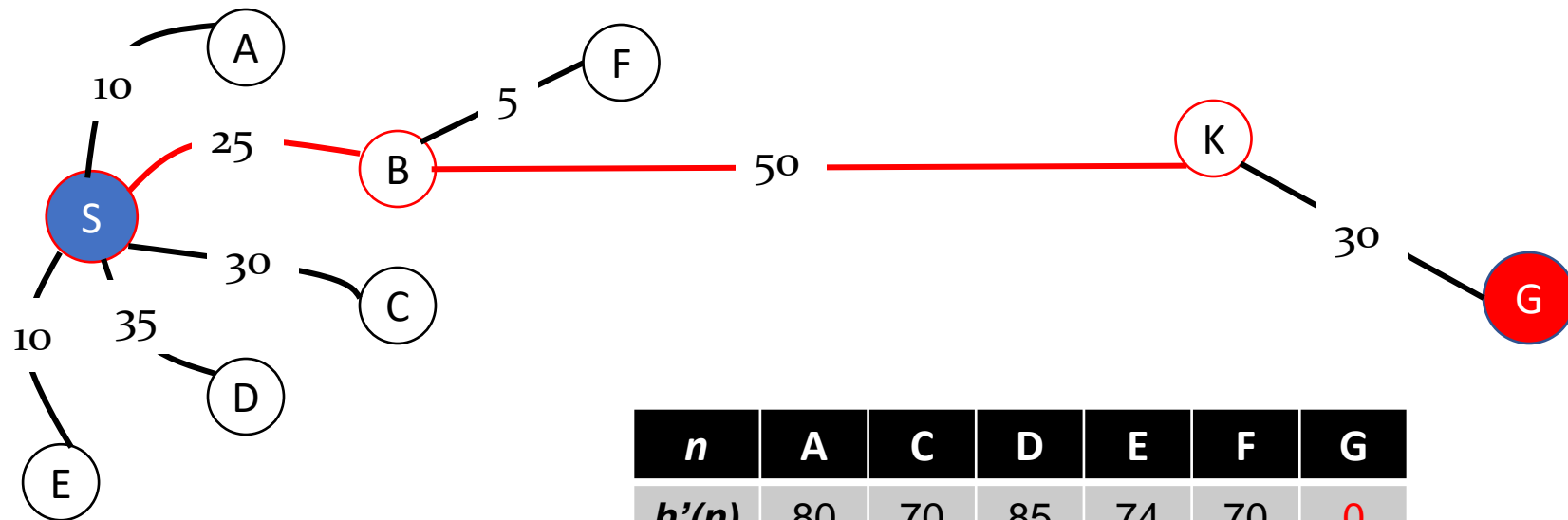
n	S	A	B	C	D	E
$h'(n)$	80	80	60	70	85	74

Langkah 2

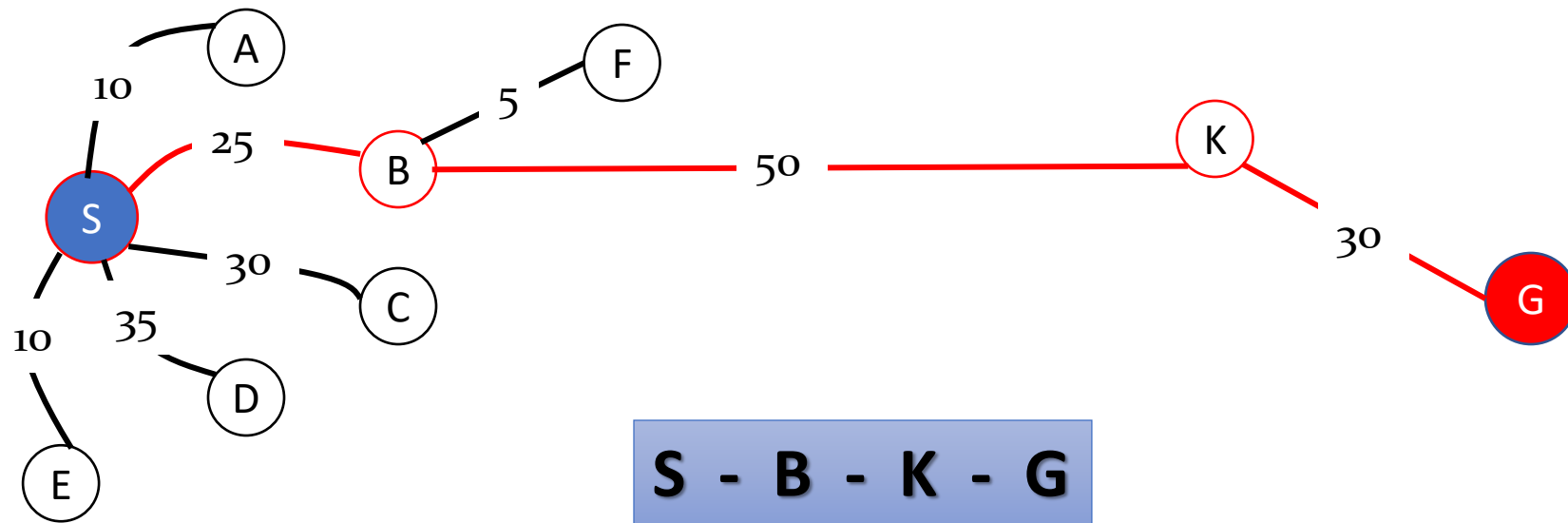


n	A	C	D	E	F	K
$h'(n)$	80	70	85	74	70	30

Langkah 3



SOLUSI




Dengan Total Jarak = 105

PENJELASAN

- Dari contoh di atas, Greedy akan menemukan solusi **S-B-K-G** dengan total jarak **105**
- Padahal ada solusi lain yg lebih optimal, yakni **S-A-B-F-K-G** dengan total jarak hanya **95**
- Dari situ bisa disimpulkan bahwa Greedy Best First Search tidak bisa menemukan solusi yang optimal

Algoritma A*

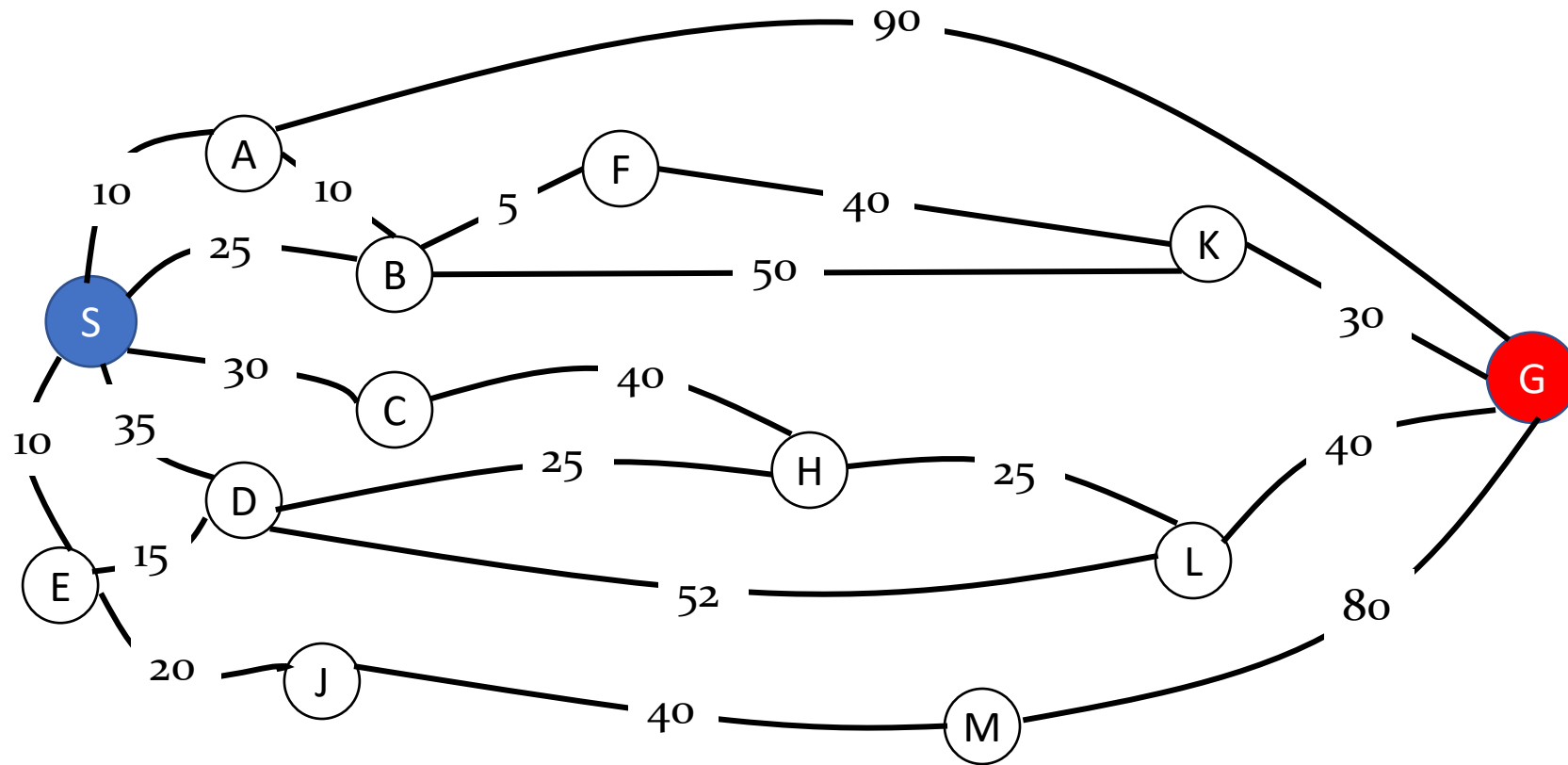
- Berbeda dg Greedy, algoritma ini akan menghitung fungsi heuristic dengan cara menambahkan biaya sebenarnya dengan biaya perkiraan. Sehingga didapatkan rumus :

$$f(n) = g(n) + h'(n)$$


$g(n)$ = Biaya sebenarnya dari Node Awal ke Node n

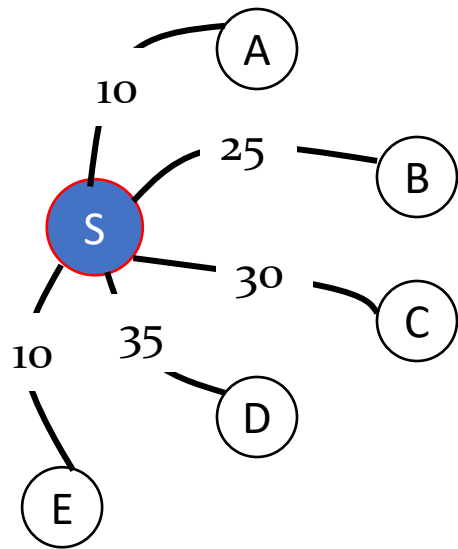
$h'(n)$ = Biaya perkiraan dari Node n ke Node Tujuan

Contoh



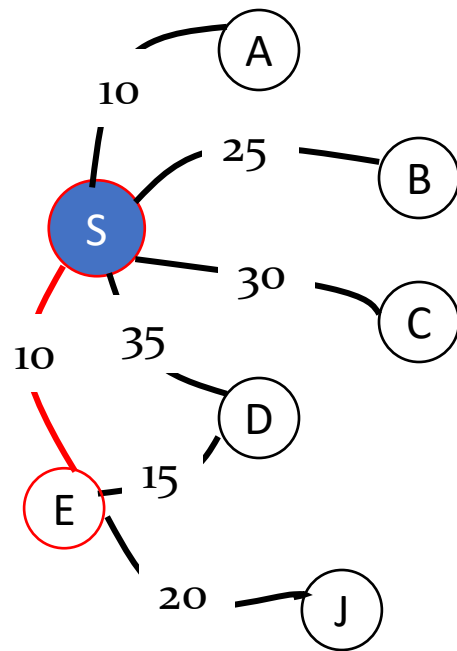
n	S	A	B	C	D	E	F	G	H	J	K	L	M
$h'(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

Langkah 1



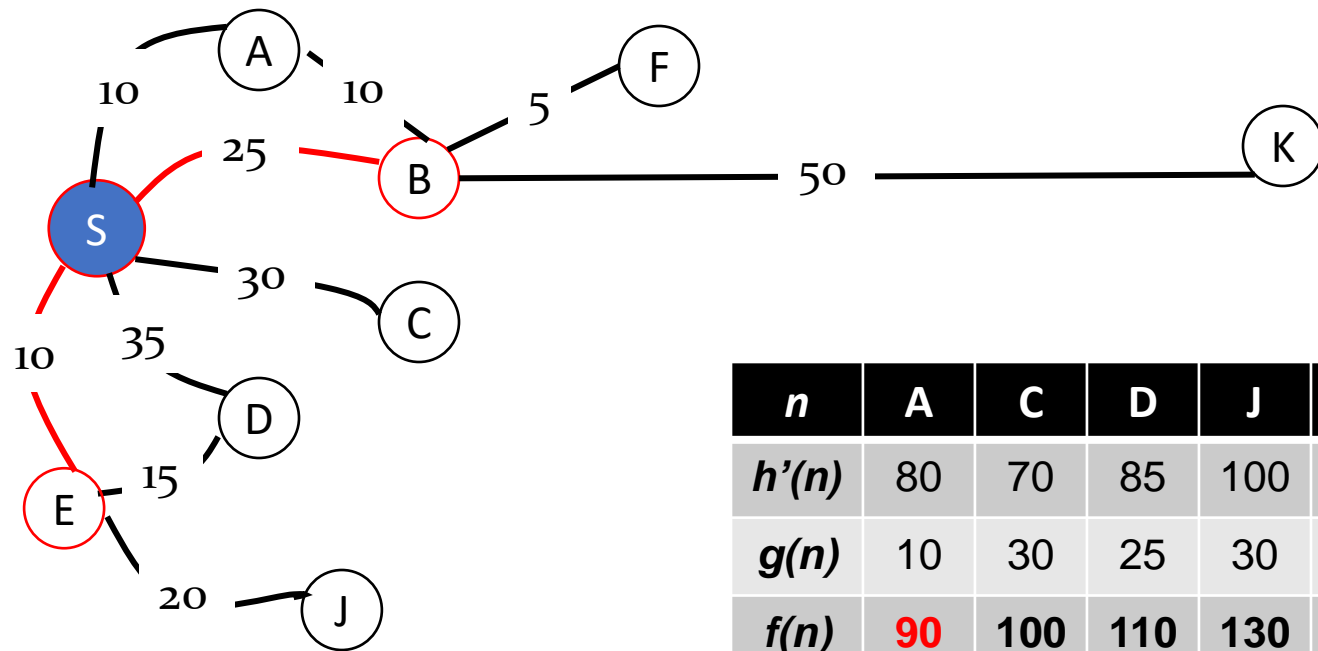
n	S	A	B	C	D	E
$h'(n)$	80	80	60	70	85	74
$g(n)$	0	10	25	30	35	10
$f(n)$	80	90	85	100	120	84

Langkah 2



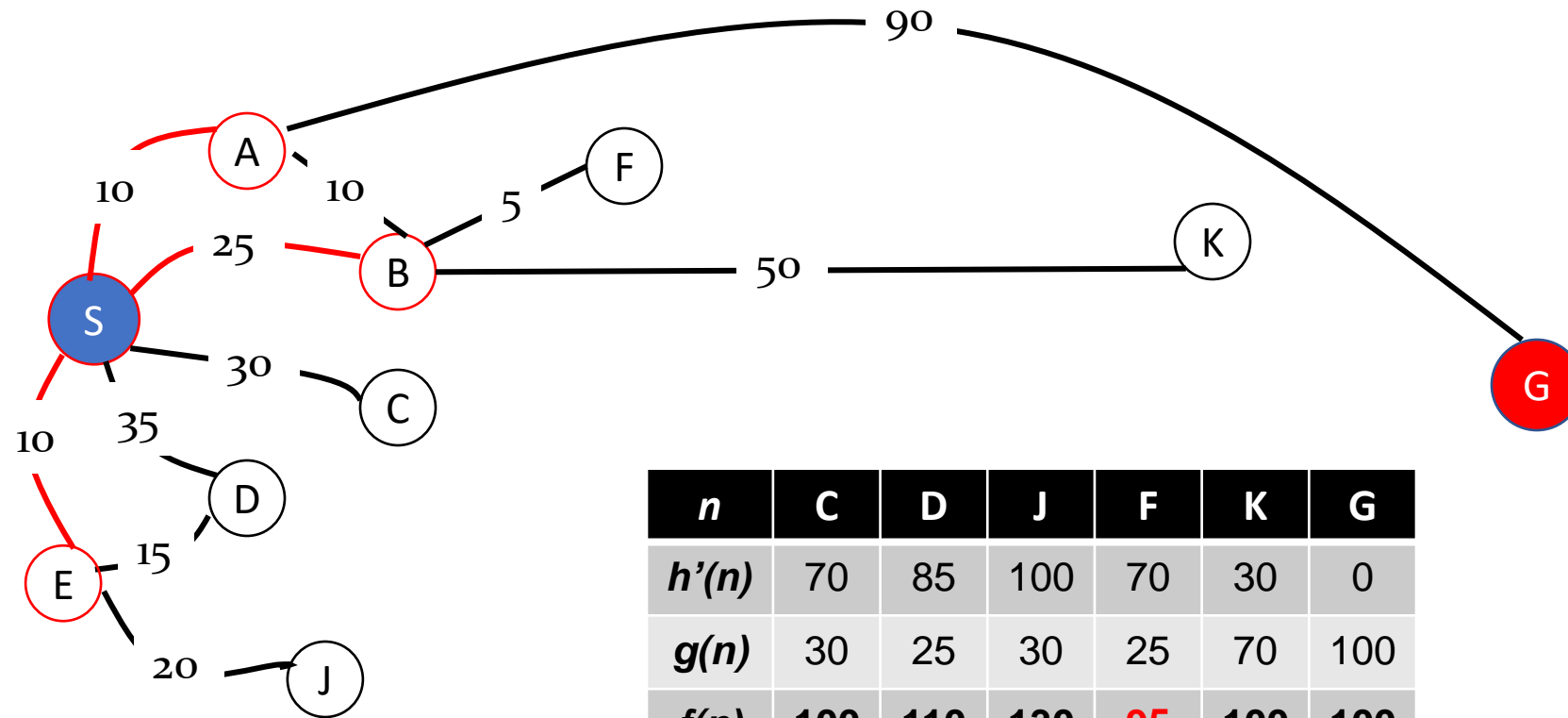
n	A	B	C	D	J
$h'(n)$	80	60	70	85	100
$g(n)$	10	25	30	25	30
$f(n)$	90	85	100	110	130

Langkah 3



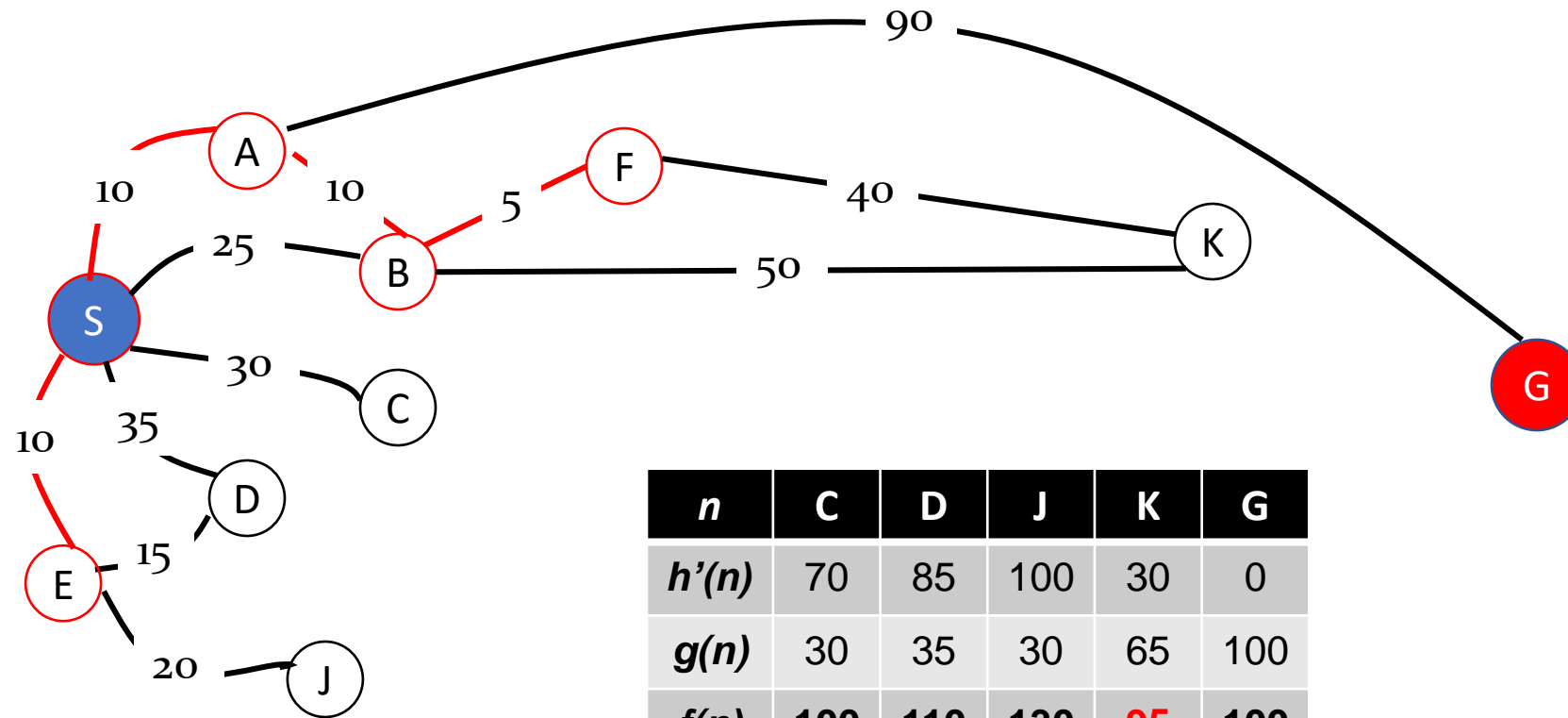
n	A	C	D	J	F	K
$h'(n)$	80	70	85	100	70	30
$g(n)$	10	30	25	30	30	75
$f(n)$	90	100	110	130	100	105

Langkah 4



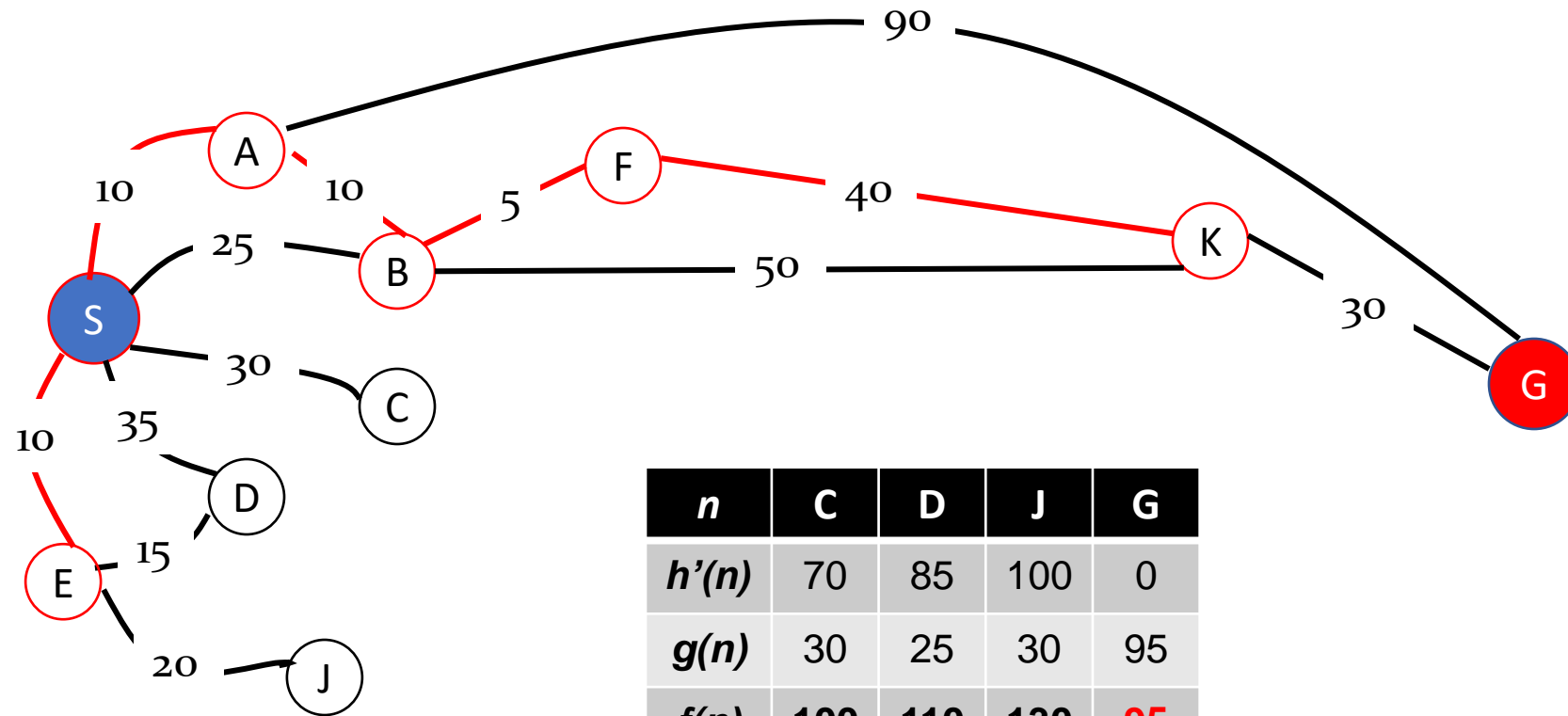
n	C	D	J	F	K	G
$h'(n)$	70	85	100	70	30	0
$g(n)$	30	25	30	25	70	100
$f(n)$	100	110	130	95	100	100

Langkah 5



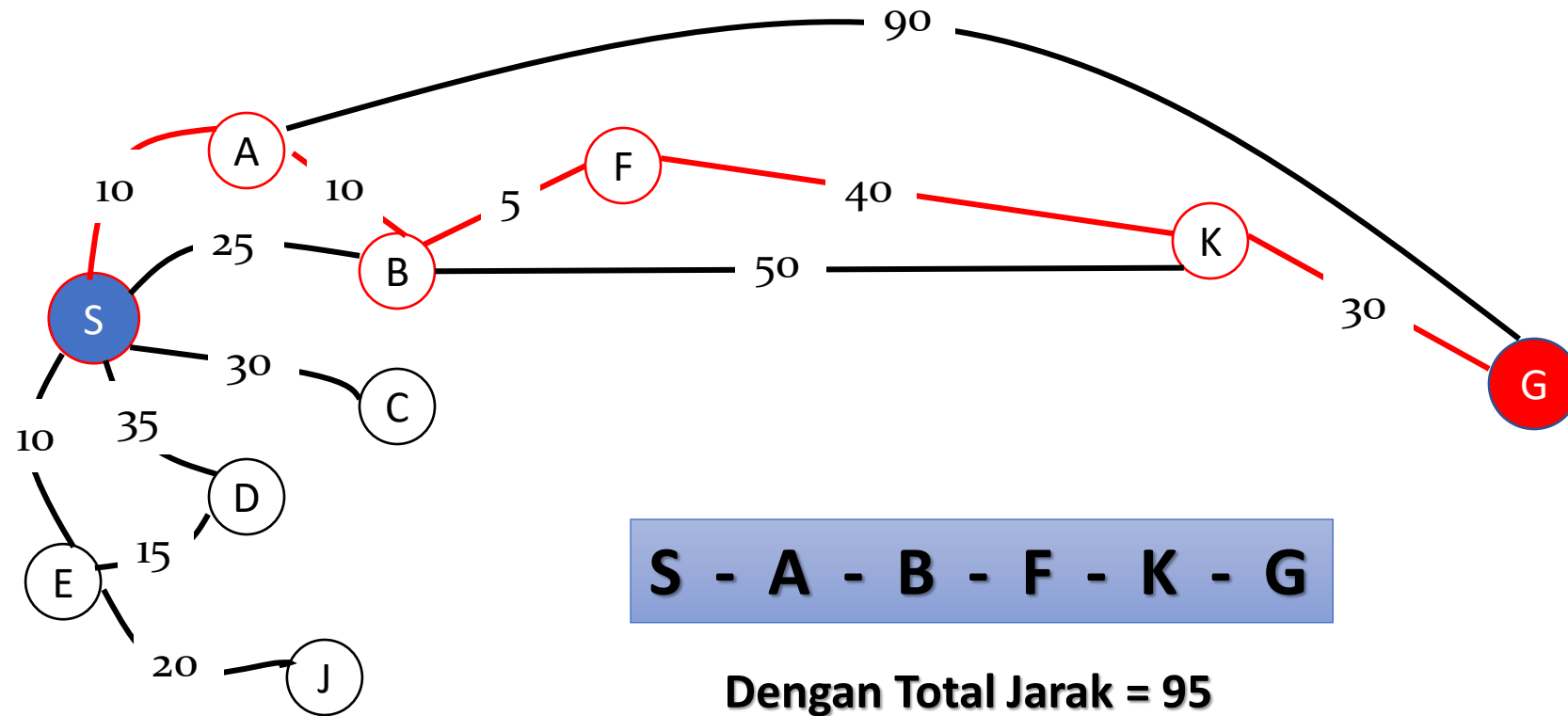
n	C	D	J	K	G
$h'(n)$	70	85	100	30	0
$g(n)$	30	35	30	65	100
$f(n)$	100	110	130	95	100

Langkah 6



n	C	D	J	G
$h'(n)$	70	85	100	0
$g(n)$	30	25	30	95
$f(n)$	100	110	130	95

Solusi



S - A - B - F - K - G

Dengan Total Jarak = 95

Kesimpulan

- Algoritma A* lebih baik dalam melakukan pencarian heuristic daripada Greedy Best First Search karena dapat menghasilkan solusi yang optimal

Latihan

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

Nilai awal puzzle :

1	2	
4	5	3
7	8	6

Goal :

1	2	3
4	5	6
7	8	

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

Bila diketahui bahwa

Nilai heuristic = $f(n) = g(n) + h(n)$

dengan

$g(n)$ = kedalaman dari pohon

$h(n)$ = jumlah angka yang masih salah posisi

