BEST FIRST SEARCH

Irvanizam Zamanhuri, M.Sc
Zulfan, S.Si, M.Sc
Dalila Husna Yunardi, B.Sc, M.Sc

Jurusan Informatika
Universitas Syiah Kuala

BEST FIRST SEARCH

- Merupakan kombinasi kelebihan teknik *depth first search* dan *breadth first search* dengan mengambil kelebihan dari kedua metode tersebut.
- 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.
- Untuk mengimplementasikan metode ini, dibutuhkan 2 antrian yang berisi node–node, yaitu :
  - OPEN →→berisi simpul--simpul yang masih memiliki peluang (peluangnya masih terbuka) untuk terpilih sebagai simpul terbaik.
  - CLOSED →→berisi simpul--simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup).

BEST FIRST SEARCH

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

  Bagaimana menentukan sebuah node terbaik saat ini?

  Dilakukan dengan menggunakan biaya perkiraan

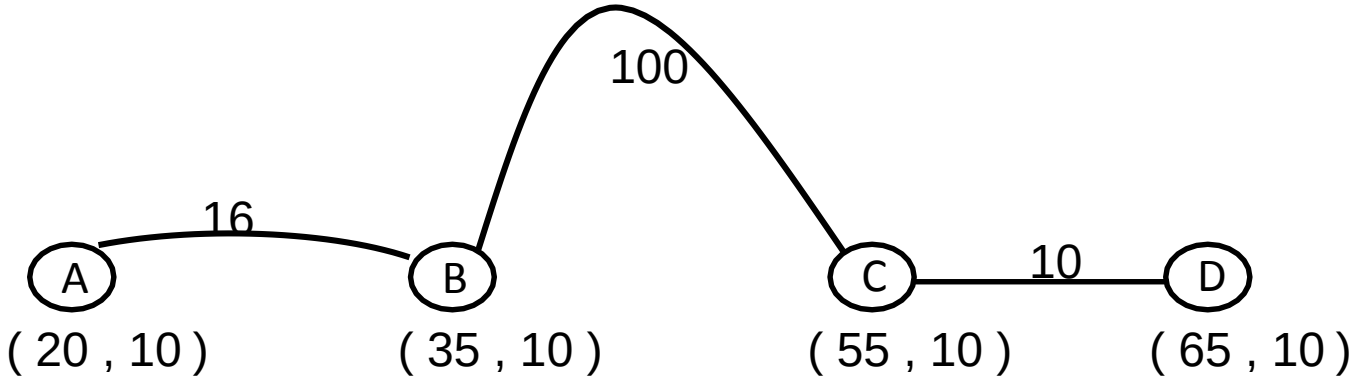
  Bagaimana caranya menentukan biaya perkiraan?

  Biaya perkiraan dapat ditentukan dengan fungsi heuristic

Best First Search
Fungsi Heuristic

- Suatufungsi heuristic dikatakan baik jika bisa memberikan biaya perkiraan yang mendekati biaya sebenarnya.
- Semakinmendekati 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 :

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

$d_{AB} = 15$

$d_{BC} = 20$

$d_{CD} = 10$

Algoritma Best First Search

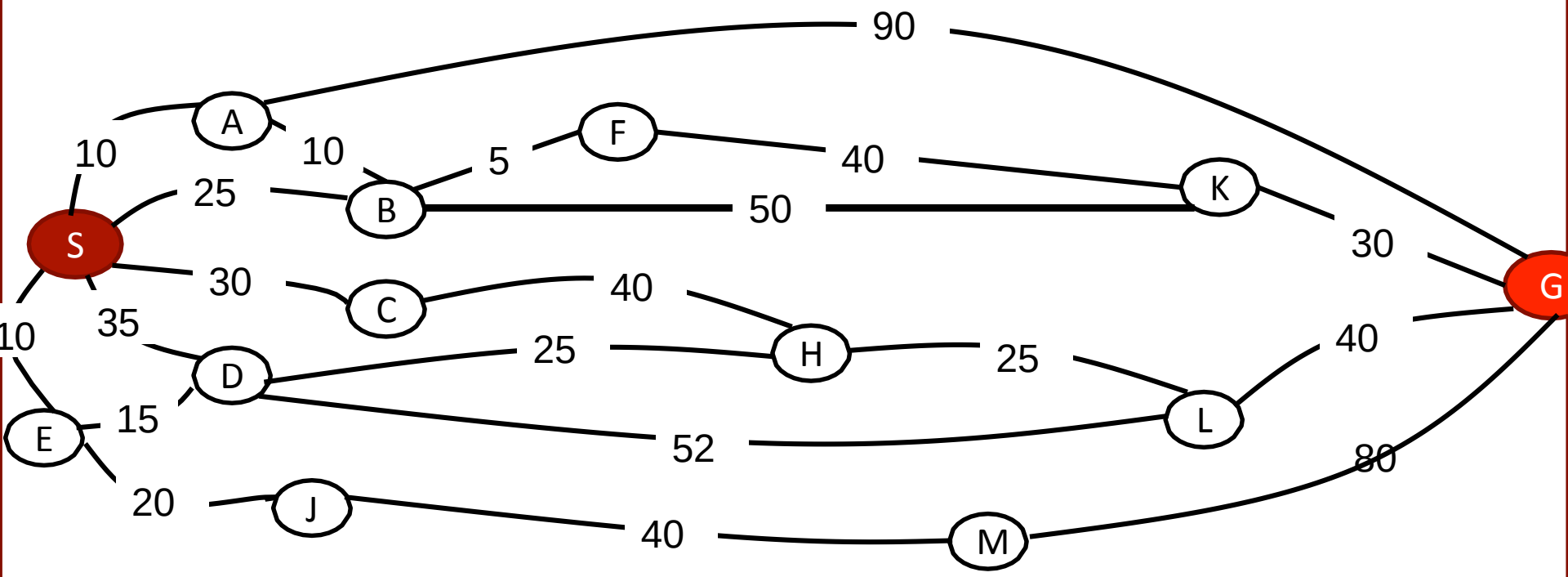
- Greedy Best First Search
- Algoritma A*

Greedy Best First Search

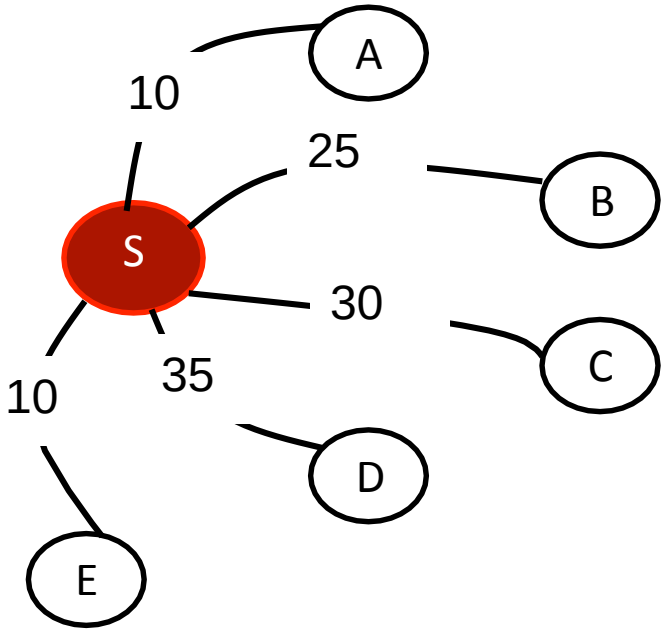
- Algoritma ini merupakan jenis algoritma Best First Search yg paling sederhana
- Algoritma ini hanya memperhitungkan biaya perkiraan saja

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

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

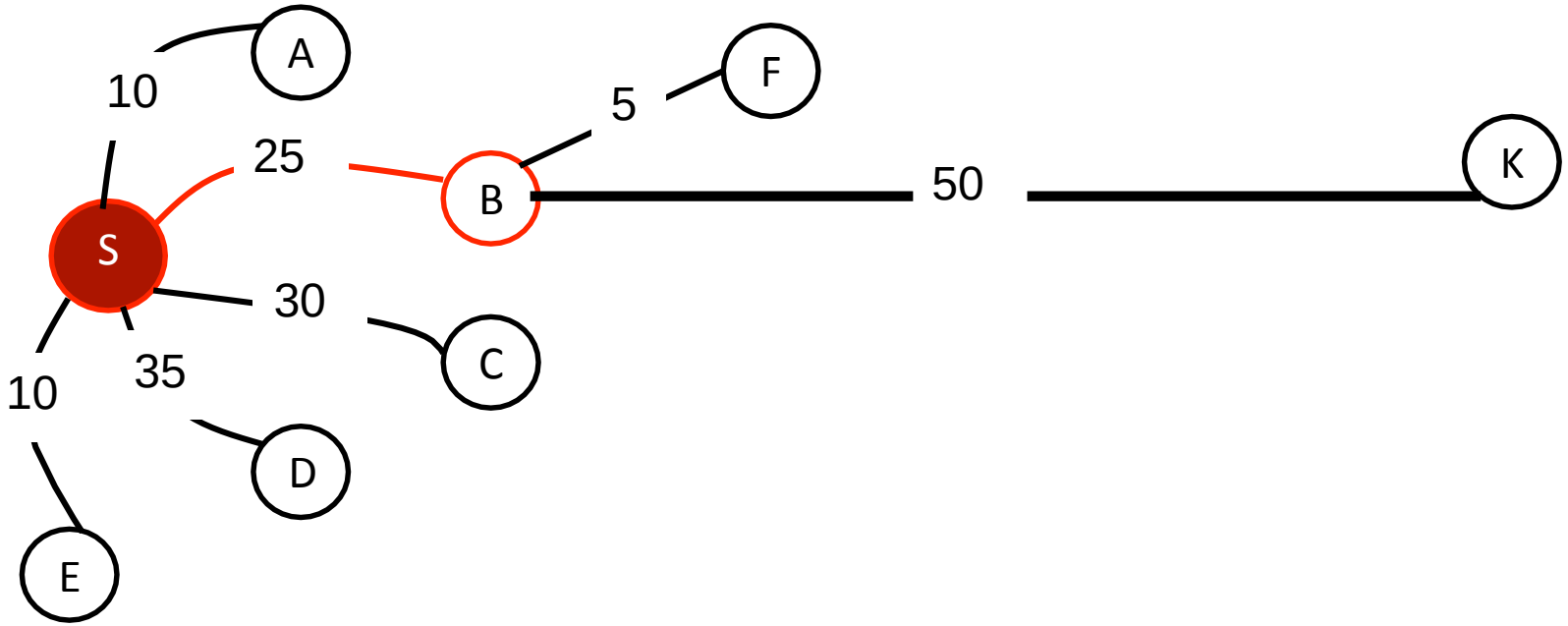


n	S	A	B	C	D	E	F	G	H	J	K	L	M
h'(n)	80	80	60	70	85	74	70	0	40	100	30	20	70

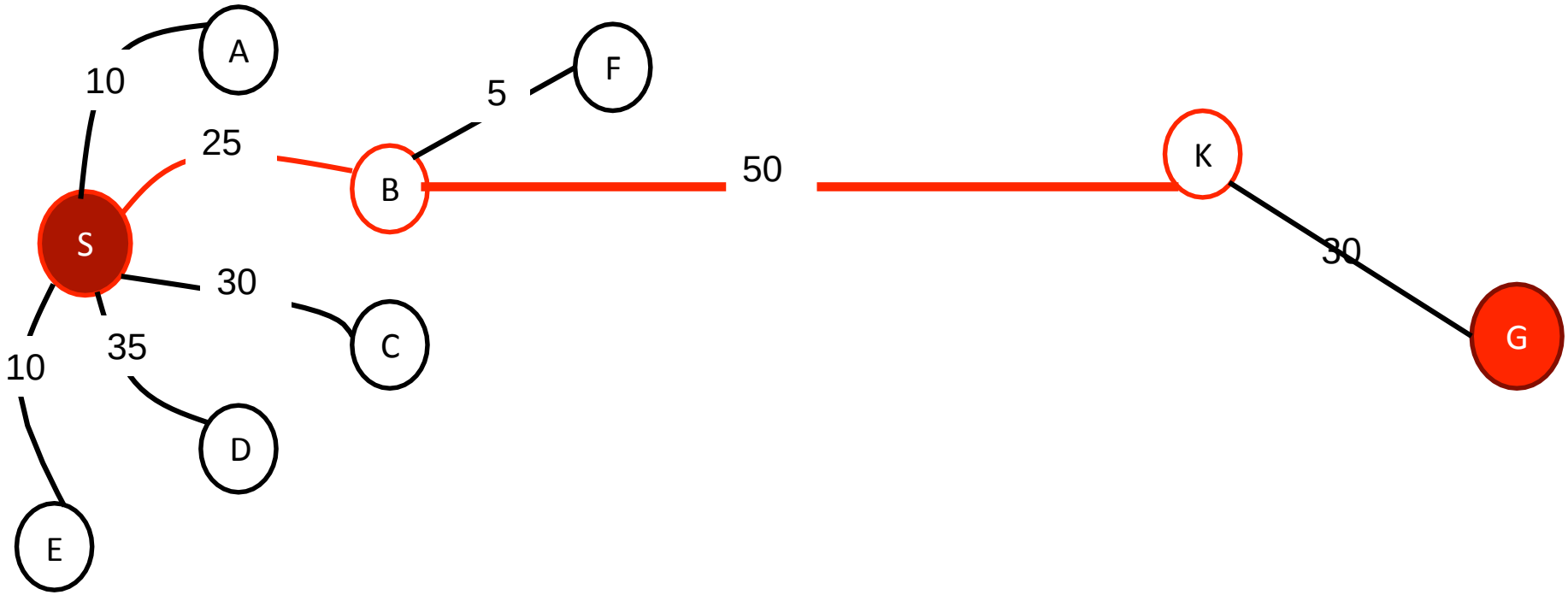


LANGKAH 1

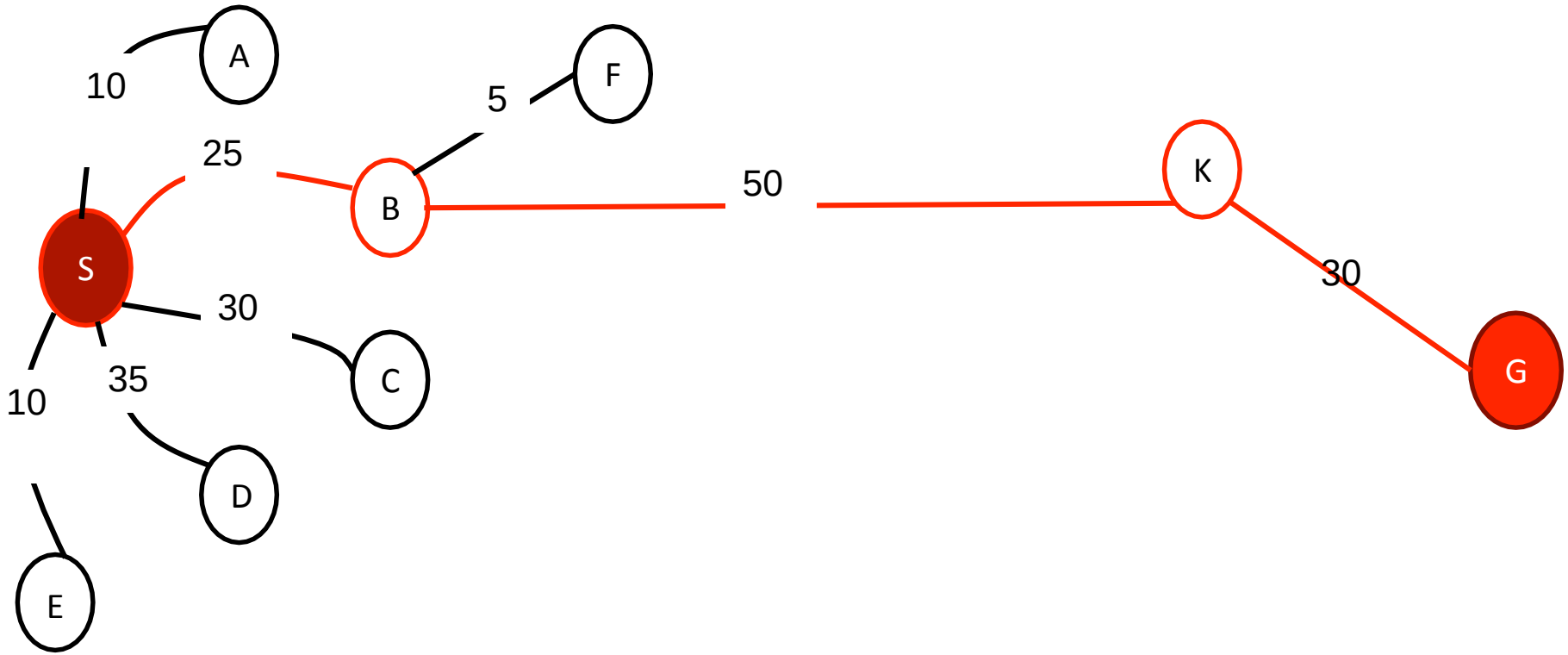
n	S	A	B	C	D	E
h'(n)	80	80	60	70	85	74



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



n	A	C	D	E	F	G
h'(n)	80	70	85	74	70	0



S – B – K – G

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 dengan 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

- Dengan perhitungan biaya seperti ini, algoritma A* adalah **complete** dan **optimal**

ALGORITMA A*

- Algoritma A* juga menggunakan dua senarai, yaitu :
  - OPEN
  - CLOSED
- Terdapat tiga kondisi bagi setiap suksessor yang dibangkitkan, yaitu :
  - Sudah berada di OPEN
  - Sudah berada di CLOSE
  - Tidak berada di OPEN maupun CLOSE

ALGORITMA A*

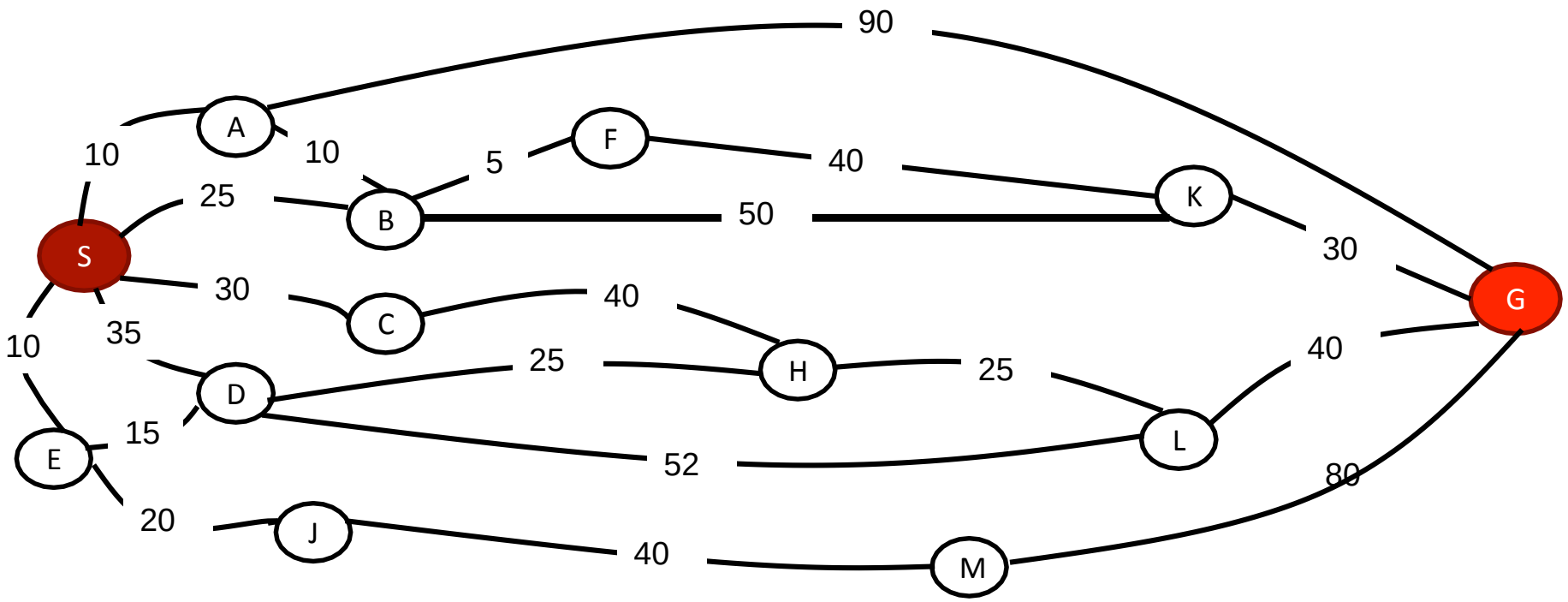
- Jika Suksessor sudah pernah berada di OPEN, maka :
  - Dilakukan pengecekan apakah perlu pengubahan *parent* atau tidak tergantung pada nilai *g* nya melalui parent lama atau parent baru
  - Jika melalui *parent* baru memberikan nilai g yang lebih kecil, maka dilakukan pengubahan parent.
  - Jika pengubahan *parent* dilakukan, maka dilakukan perbaruan (update) nilai **g** dan **f** pada suksessor tersebut.
  - Dengan perbaruan ini, suksessor tersebut memiliki kesempatan yang lebih besar untuk terpilih sebagai simpul terbaik

ALGORITMA A*

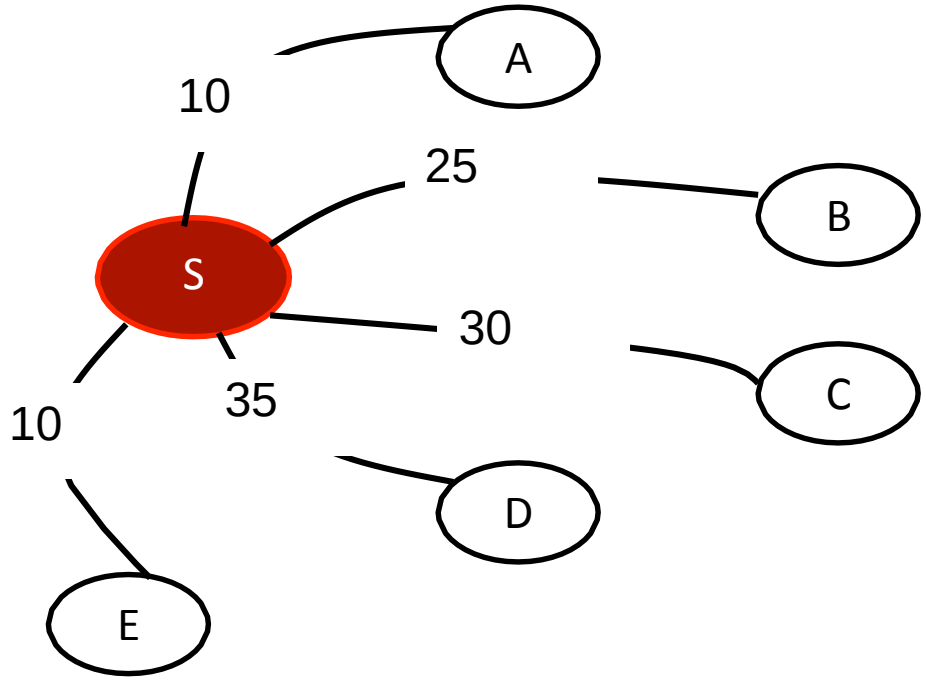
- Jika Suksessor sudah pernah berada di CLOSED, maka:
  - Dilakukan pengecekan apakah perlu pengubahan *parent* atau tidak.
  - Jika ya, maka dilakukan perbaruan(update) nilai **g** dan nilai **f** pada suksessor tersebut pada semua "anak cucunya" yang sudah pernah berada di OPEN.
  - Dengan perbaruan ini, maka semua anak cucunya memiliki kesempatan lebih besar untuk terpilih sebagai simpul terbaik.

ALGORITMA A*

- Jika suksessor tidak berada di OPEN maupun di CLOSED, maka :
  - Suksessor tersebut dimasukkan ke dalam OPEN
  - Tambahkan suksessor tersebut sebagai suksessornya best node.
  - Hitung biaya suksessor tersebut dengan rumus f=g+h

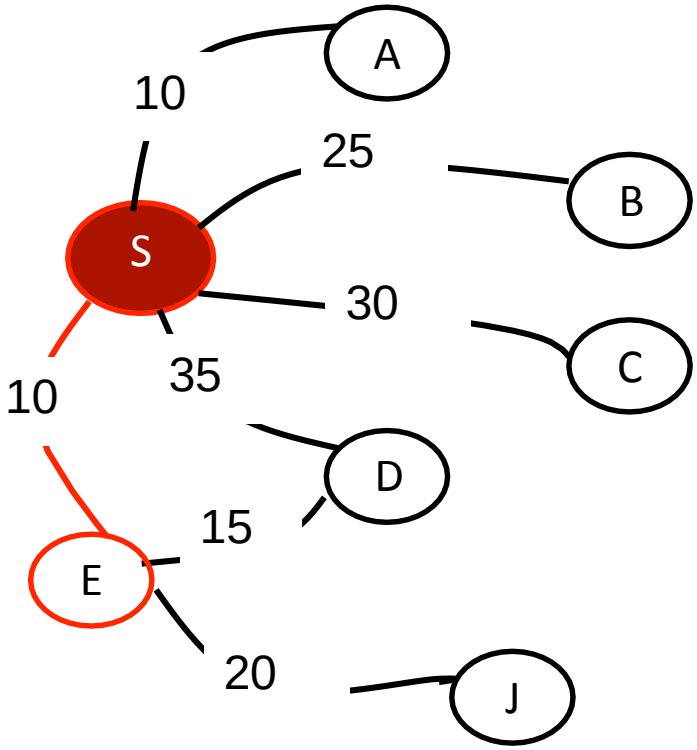


n	S	A	B	C	D	E	F	G	H	J	K	L	M
h'(n)	80	80	60	70	85	74	70	0	40	100	30	20	70



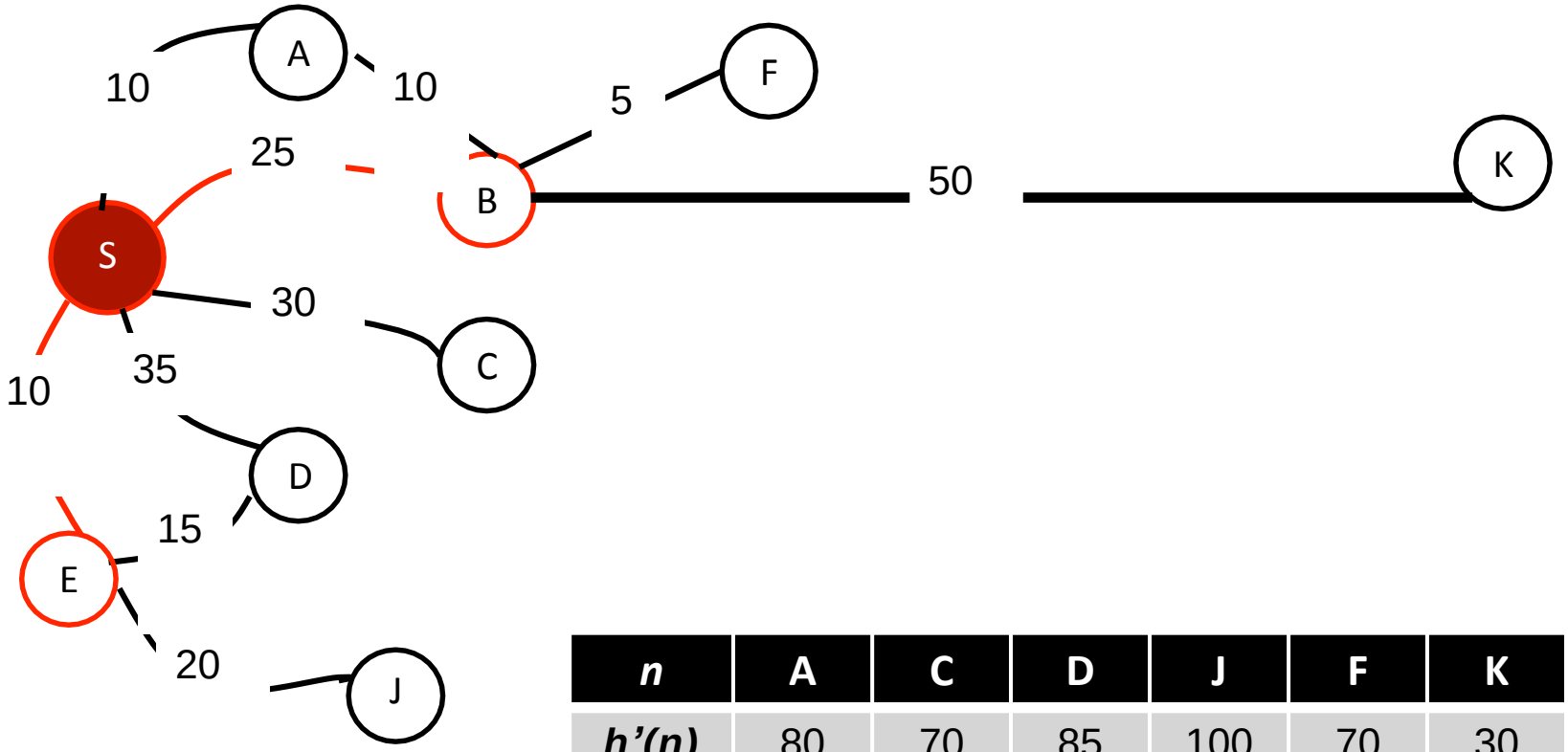
LANGKAH 1

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**



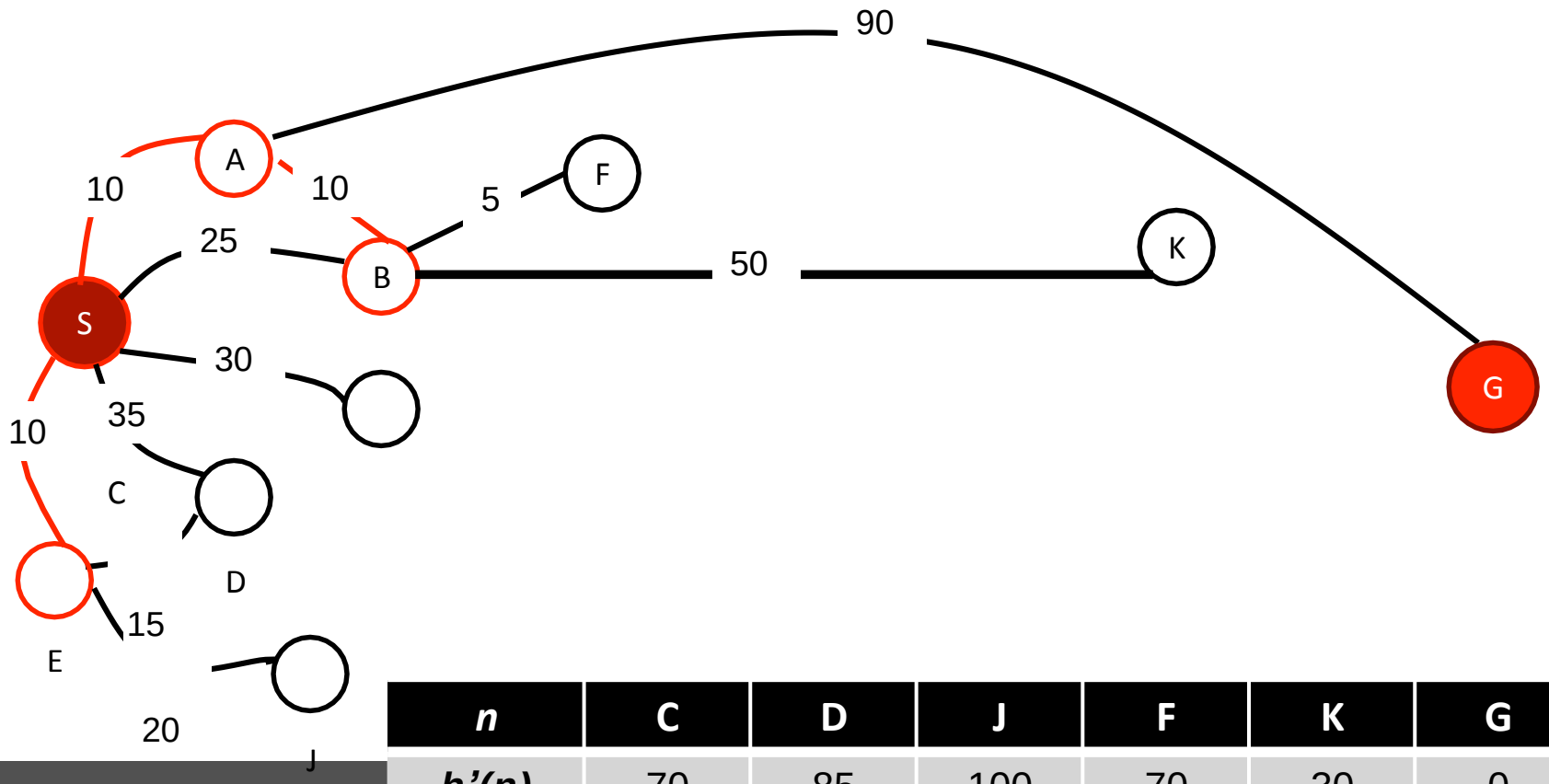
LANGKAH 2

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**



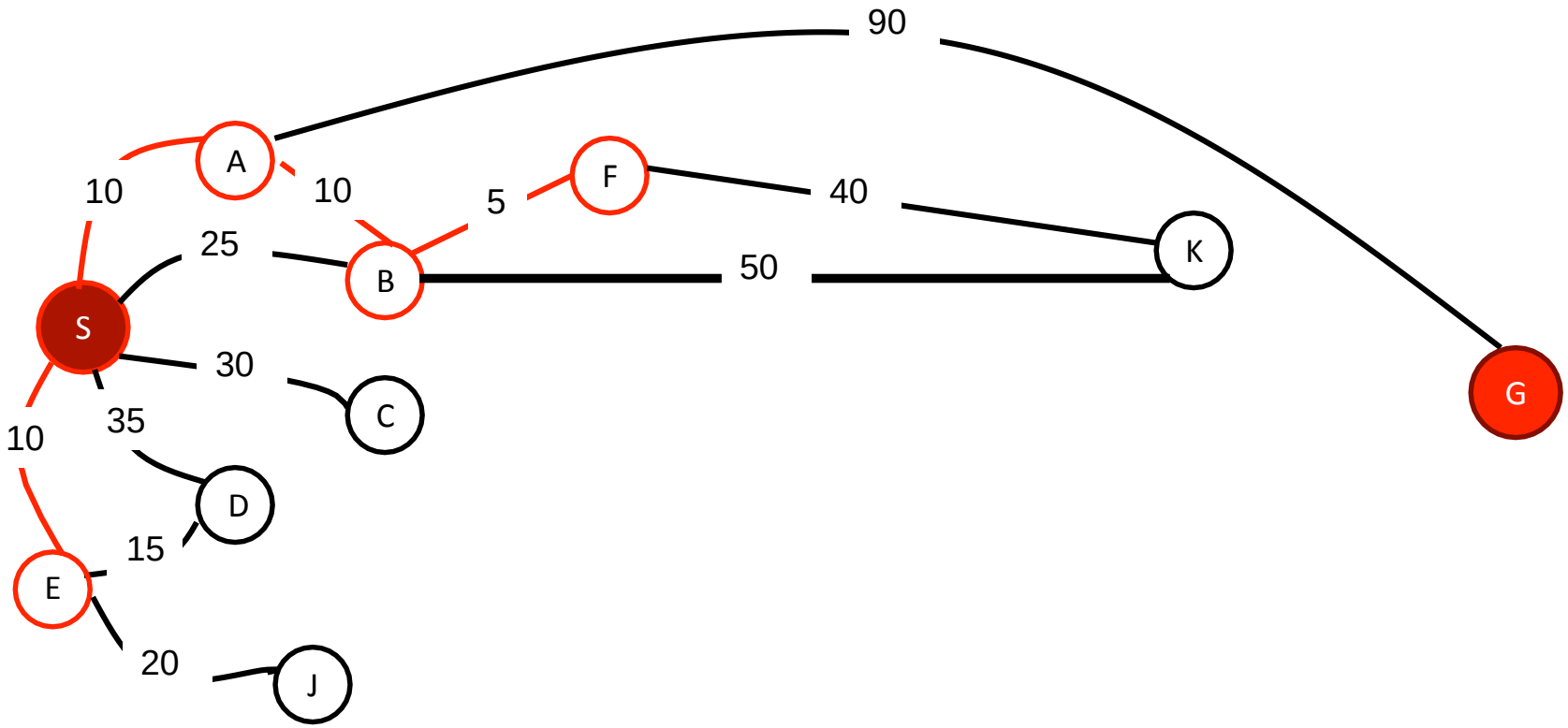
LANGKAH 3

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**



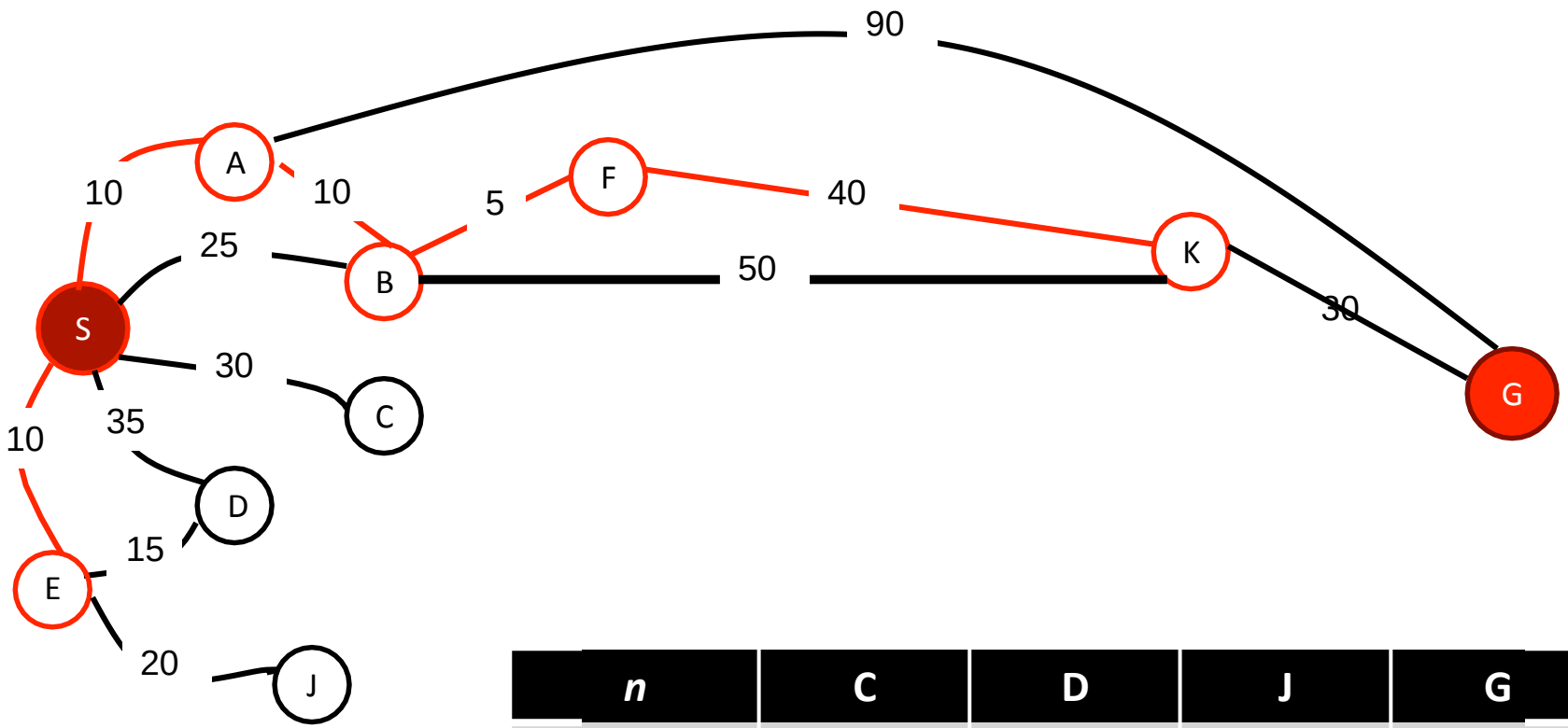
n	C	D	J	F	K	G
h'(n)	70	85	100	70	30	0
g(n)	30	25	30	25	70	100
f(n)	**100**	**110**	**130**	**95**	**100**	**100**

LANGKAH 4



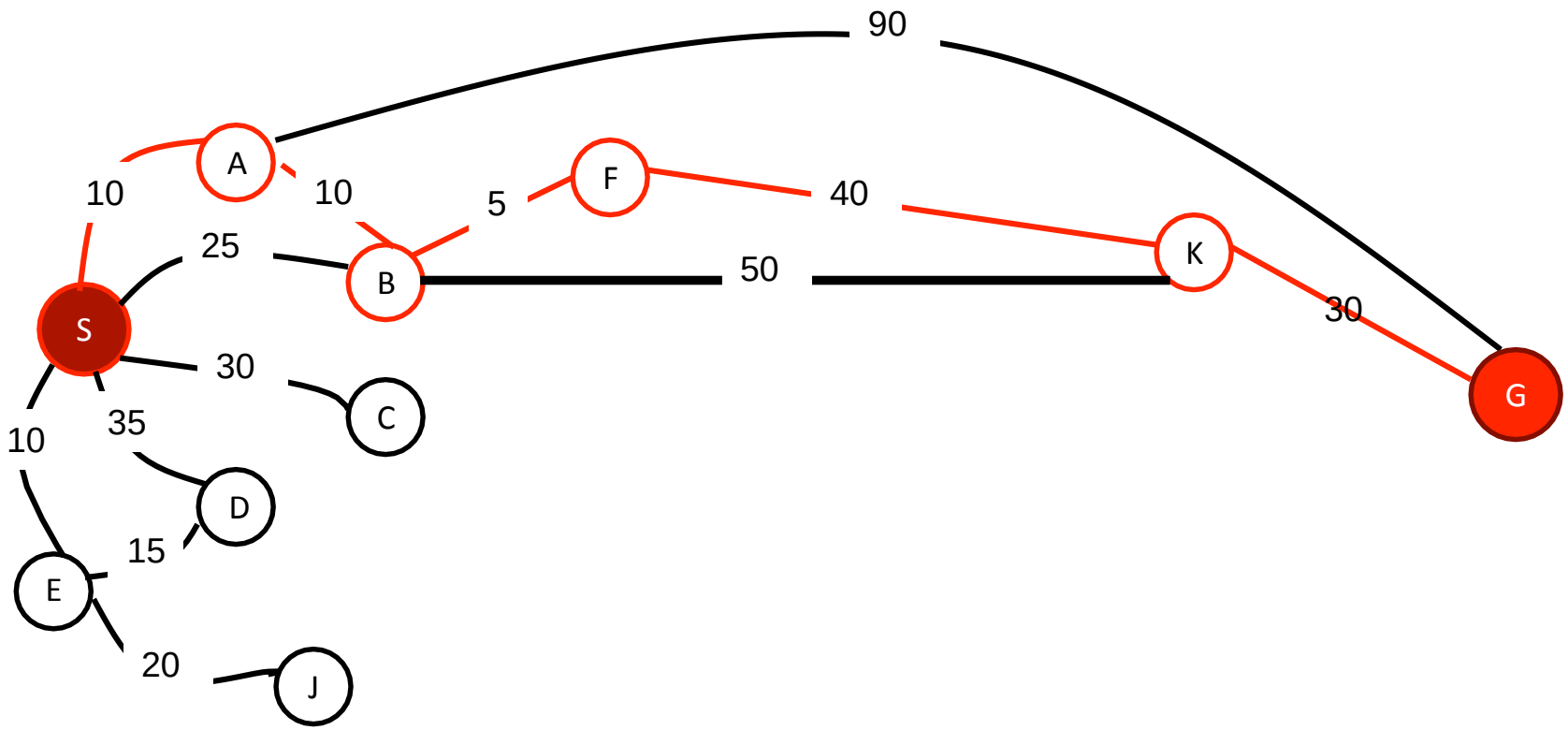
LANGKAH 5

n	C	D	J	K	G
h'(n)	70	85	100	30	0
g(n)	30	25	30	65	100
f(n)	**100**	**110**	**130**	**95**	**100**



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

LANGKAH 6



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 mengasilkan solusi yang optimal.