TRÍ TUỆ NHÂN TẠO

Artificial Intelligence

Đoàn Vũ Thịnh
Khoa Công nghệ Thông tin
Đại học Nha Trang
Email: thinhdv@ntu.edu.vn

Nha Trang, 06-2023

- Các giải thuật tìm kiếm không sử dụng thông tin phản hồi (hay là giải thuật tìm kiếm mù): duyệt theo chiều rộng (BFS) và duyệt theo chiều sâu (DFS) có đặc điểm:
- Xây dựng tất cả không gian lời giải tiềm năng theo cách vét cạn, không bỏ sót và không lặp lại.
- Trong rất nhiều trường hợp, các giải thuật như vậy không khả thi vì không gian trạng thái bài toán quá lớn, tốc độ xử lý và bộ nhớ của máy tính không cho phép duyệt các lời giải tiềm năng.

- Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)

- OPEN: tập chứa các trạng thái được sinh ra nhưng chưa được xét đến (vì ta đã chọn trạng thái khác). Thực ra OPEN là một hàng đợi ưu tiên mà trong đó, phần tử có độ ưu tiên cao nhất là phần tử tốt nhất.
- CLOSE: tập chứa các trạng thái đã được xét đến. Chúng ta cần lưu trữ những trạng thái này trong bộ nhớ để đề phòng trường hợp khi một trạng thái mới được tạo ra trùng với trạng thái mà ta đã xét trước đó.

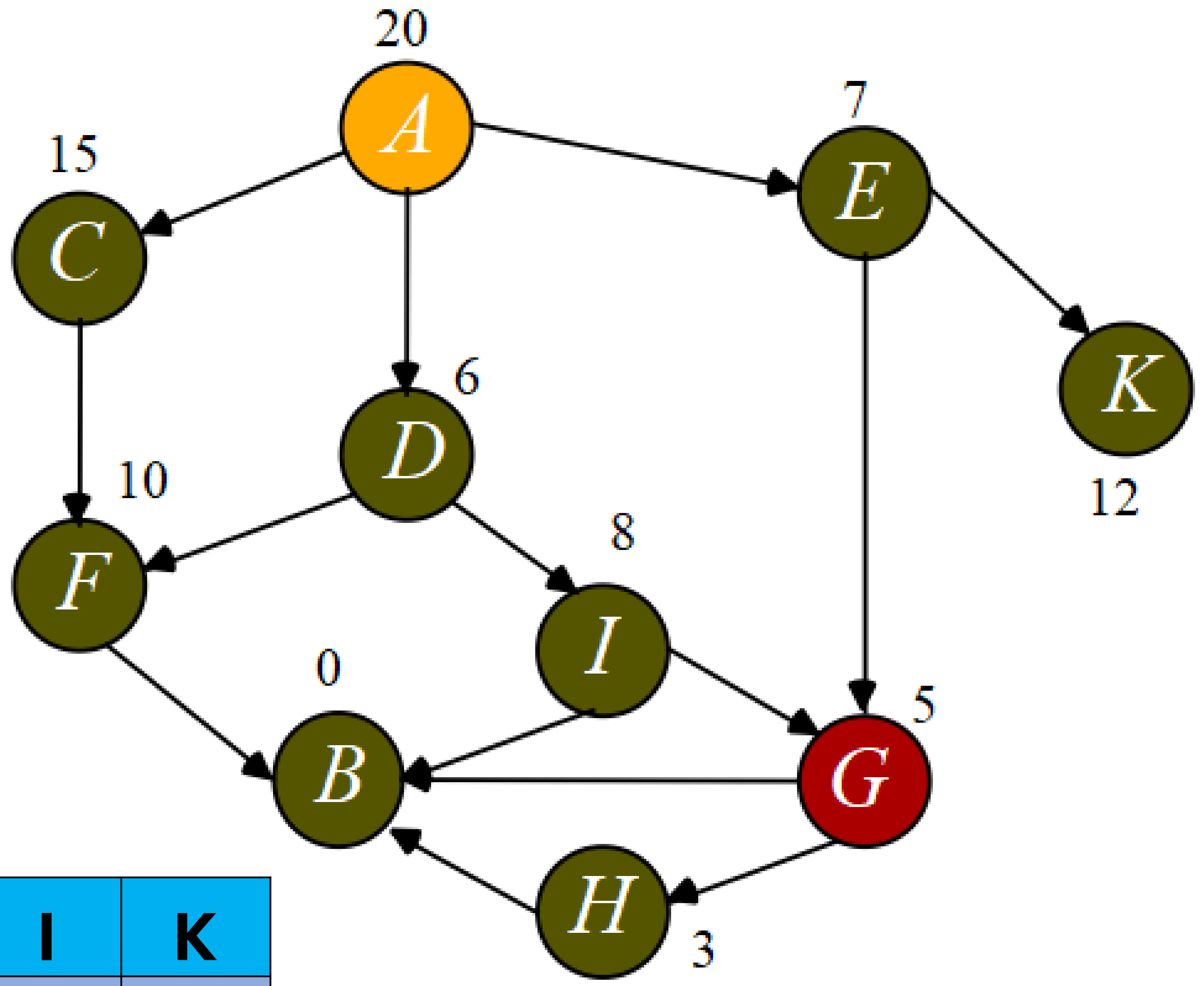
Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)

- 1. Đặt OPEN chứa trạng thái đầu
- 2. Lặp cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện:
  - 2.1. Chọn trạng thái tốt nhất (Tmax) trong OPEN (Xóa Tmax ra khỏi danh sách OPEN)
  - 2.2. Nếu Tmax là trạng thái đích thì Kết thúc
  - 2.3. Ngược lại, tạo ra các trạng thái kế tiếp Tk có thể có từ trạng thái Tmax. Đối với mỗi Tk thực hiện:
    - Tính f(Tk); Thêm Tk vào OPEN

- Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)

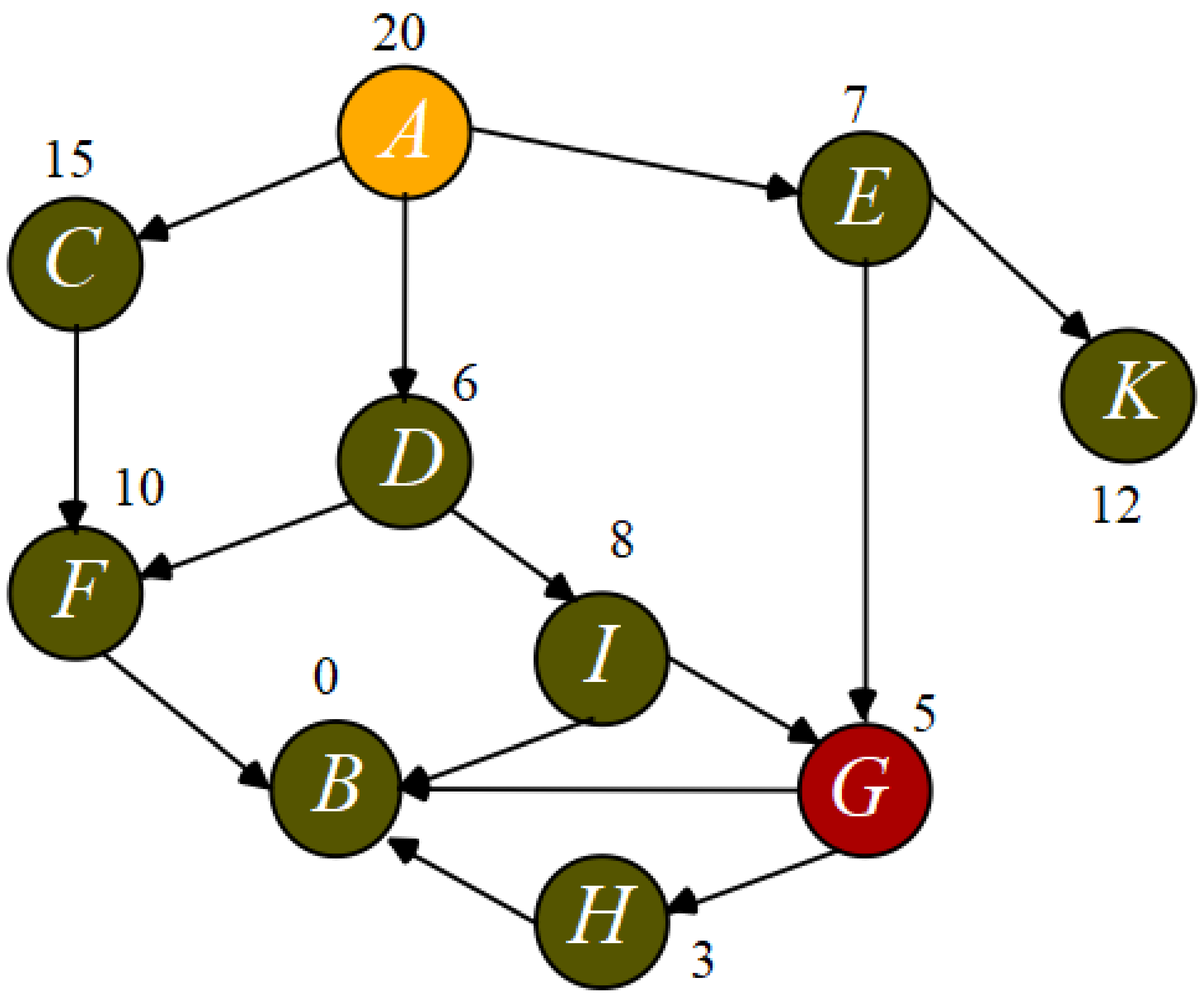
- Trình bày thuật toán Best-First Search (BFS). Áp dụng BFS để tìm đường đi ngắn nhất từ đỉnh A đến G trên đồ thị với các ước lượng heuristic của các trạng thái so với trạng thái đích được liệt kê:



A	B	C	D	E	F	G	H	I	K
20	0	15	6	7	10	5	3	8	12

- Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)

Bước	n	Γ(n)	Open	Close	$f_{SORT\uparrow}$
0			{A}	∅	
1	A	{C,D,E}	{D,E,C}	{A}	$f(C,D,E)$
2	D	{F,I}	{E,I,F,C}	{A,D}	$f(C,E,F,I)$
3	E	{K,G}	{G,I,F,K,C}	{A,D,E}	$f(C,F,I,K,G)$
4	G	TRUE			

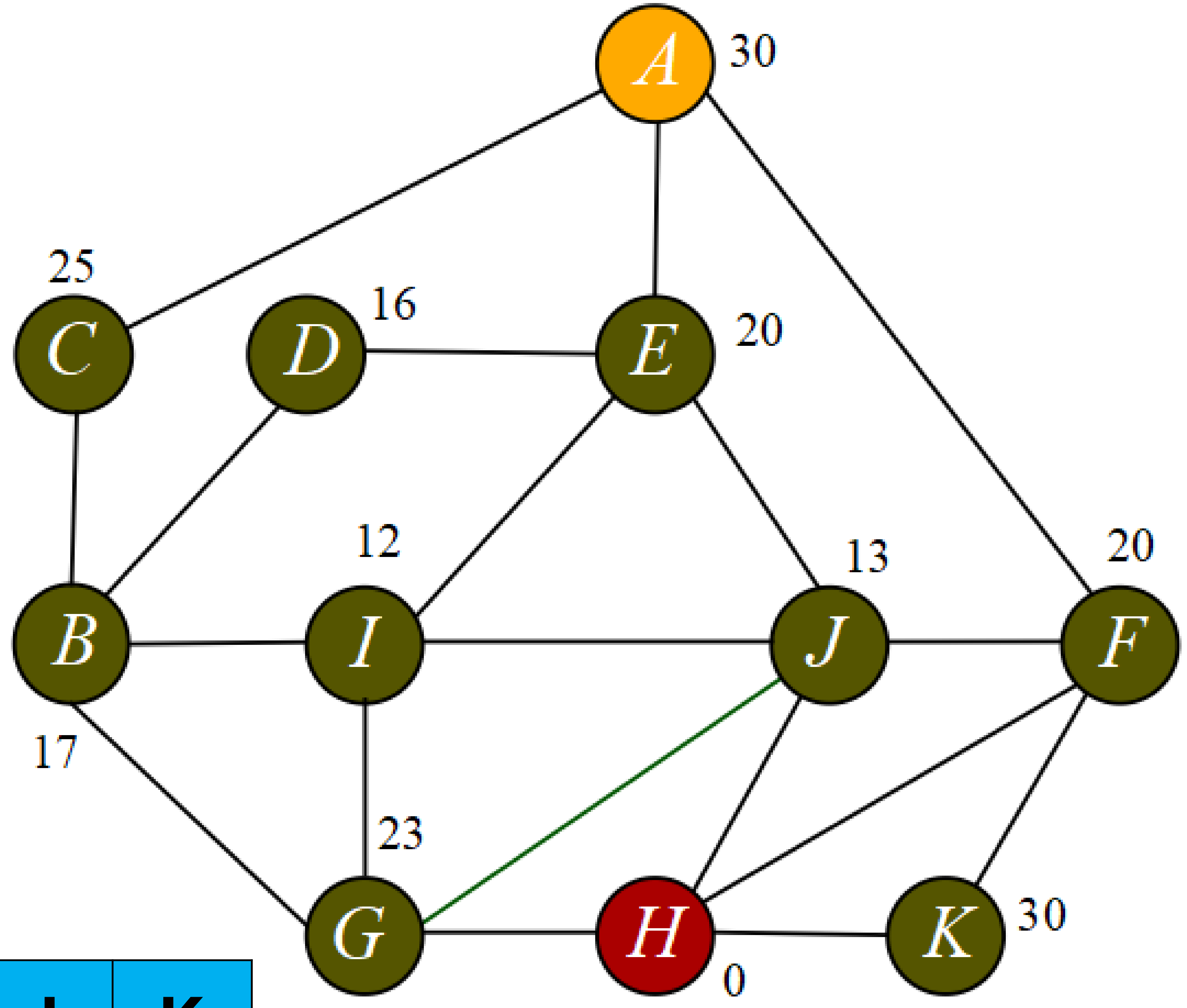


- A ⇨ E ⇨G

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)

▪ Trình bày thuật toán Best-First Search (BFS). Áp dụng BFS để tìm đường đi ngắn nhất từ đỉnh **A** đến **H** trên đồ thị với các ước lượng heuristic của các trạng thái so với trạng thái đích được liệt kê:



A	B	C	D	E	F	G	H	I	J	K
30	17	25	16	20	20	23	0	12	13	30

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật AT (Algorithm for Tree)

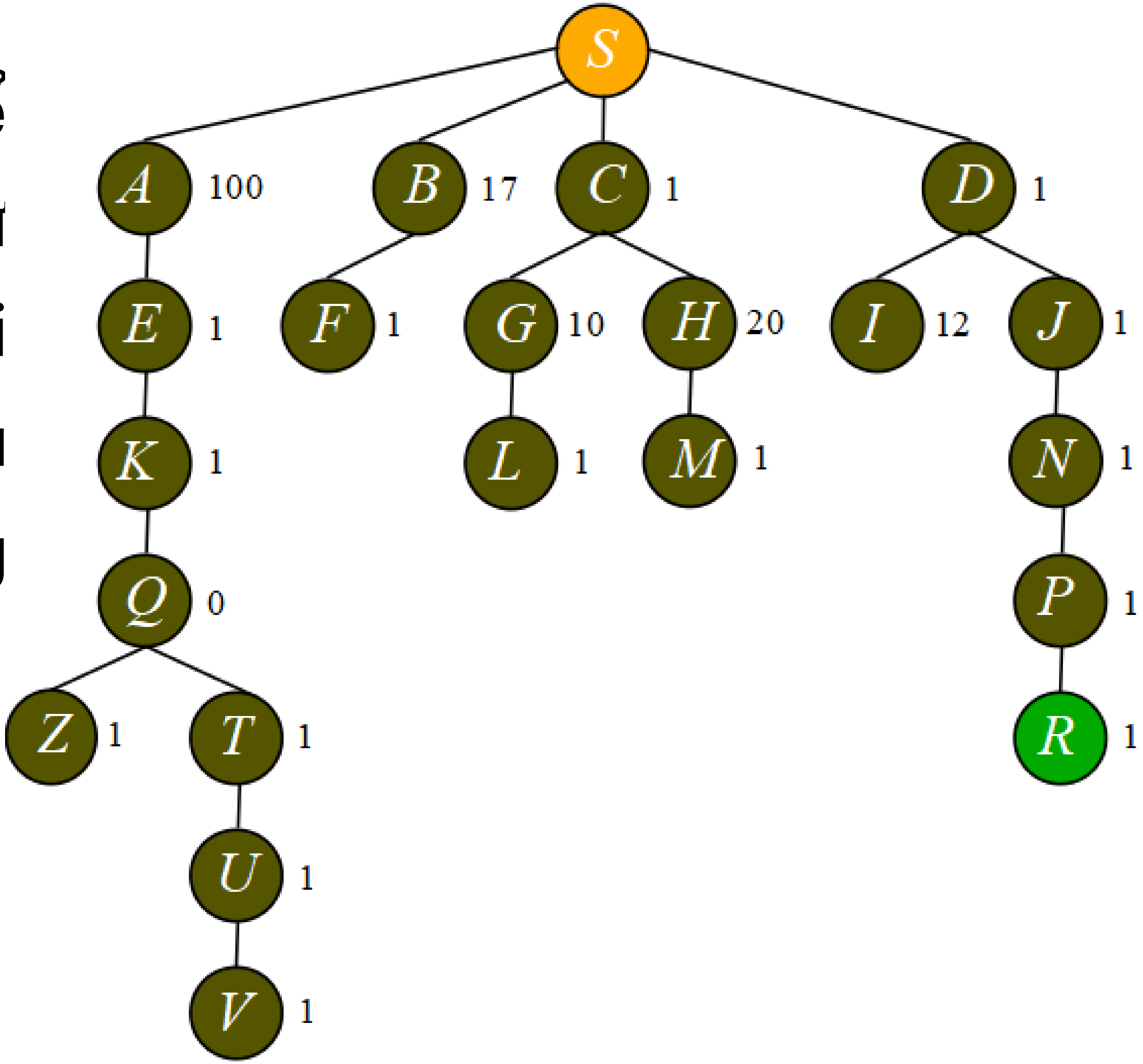
- Thuật giải BFS khá đơn giản. Tuy vậy, trên thực tế cũng như tìm kiếm theo chiều rộng và chiều sâu, hiếm khi ta dung BFS một cách trực tiếp. Thông thường, người ta thường dùng các phiên bản của BFS là AT, AKT và A*
- Thông thường, trong các phương án tìm kiếm BFS. Độ tốt f của một trạng thái được tính dựa vào 2 giá trị **g** và **h'**.
  - h': ước lượng chi phí từ trạng thái hiện hành – trạng thái đích
  - g: chiều dài đoạn đường đã đi từ trạng thái ban đầu đến trạng thái hiện tại (g là chi phí thực sự chứ không phải ước lượng)

Giải thuật AT (Algorithm for Tree)

- Bước 1: Chọn s = đỉnh xuất phát; g(s)=0
- Bước 2: Chọn 1 đỉnh từ tập OPEN: =n, có g(n) = min
  - 2.1. Nếu n ≡ đích: đường đi từ s ⇨ n là tối ưu
  - 2.2. Nếu OPEN = ∅: không tìm thấy đường đi
  - 2.3. Nếu ∃! nhiều hơn 1 đỉnh n có g(n) = min:
    - Kiểm tra (∃! ≡ đích): dừng
    - Ngược lại, chọn ngẫu nhiên một đỉnh: =n
- Bước 3: Đưa đỉnh đã duyệt vào CLOSE, tạo Γ(n): các đỉnh mà n trỏ tới
  - for mỗi nút con m của Γ(n):
    - $if\ (m\ !\in OPEN)$:
      - $g(m) := g(n) + cost(n, m);\ OPEN := OPEN\ \cup \{m\}$
      - So sánh g(m) với $g_{NEW}(m)$ và cập nhật
- Bước 4: Quay lại bước 2

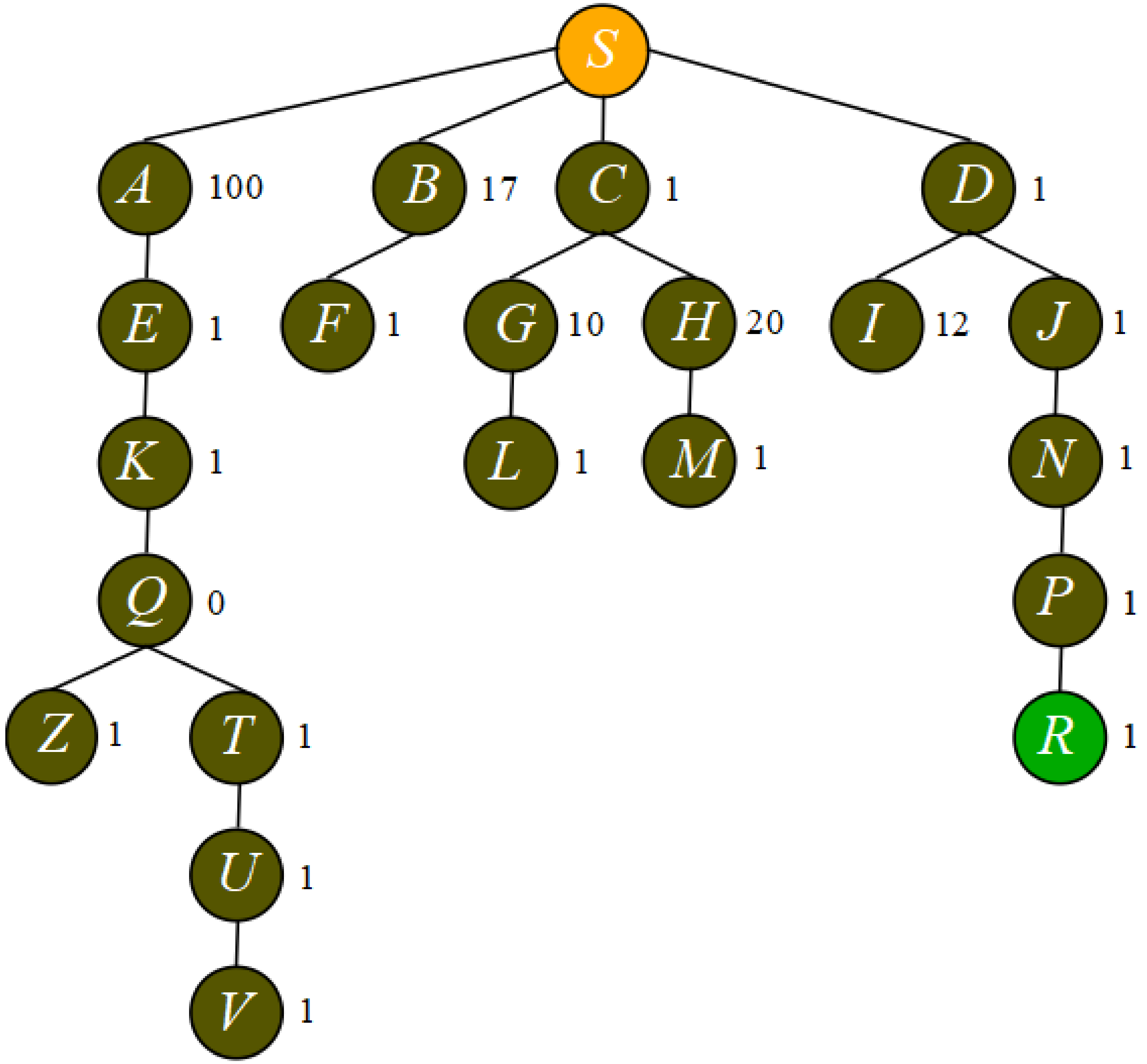
- Giải thuật AT (Algorithm for Tree)

- Trình bày thuật toán AT để tìm đường đi ngắn nhất từ đỉnh **S** đến **R** trên đồ thị với các ước lượng heuristic của các trạng thái so với trạng thái đích được liệt kê như hình

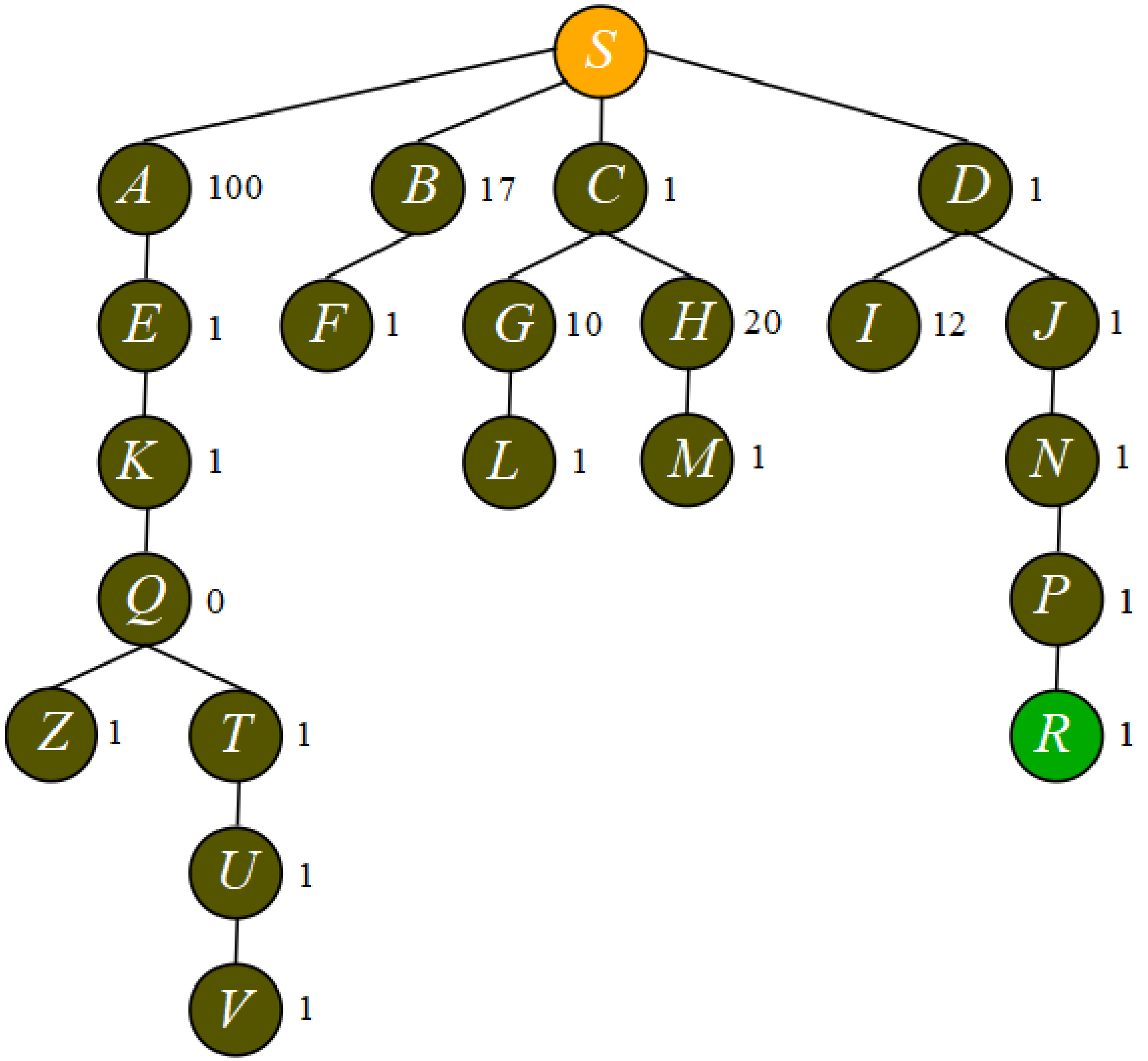


Giải thuật AT (Algorithm for Tree)

- Bước 1: OPEN(S); g(S)=0
- Bước 2: n=S, Γ(n)={A,B,C,D}, CLOSE={S}
  - g(A$\notin$OPEN)=g(S) + g(S$\rightarrow$A)
    = 0+100 = 100
  - g(B$\notin$OPEN)=g(S) + g(S$\rightarrow$B) = 0+17 = 17
  - g(C$\notin$OPEN)=g(S) + g(S$\rightarrow$C) = 0+1 = 1
  - g(D$\notin$OPEN)=g(S) + g(S$\rightarrow$D) = 0+1 = 1
  - g(min) = g(C)
  - OPEN={C,D,B,A}



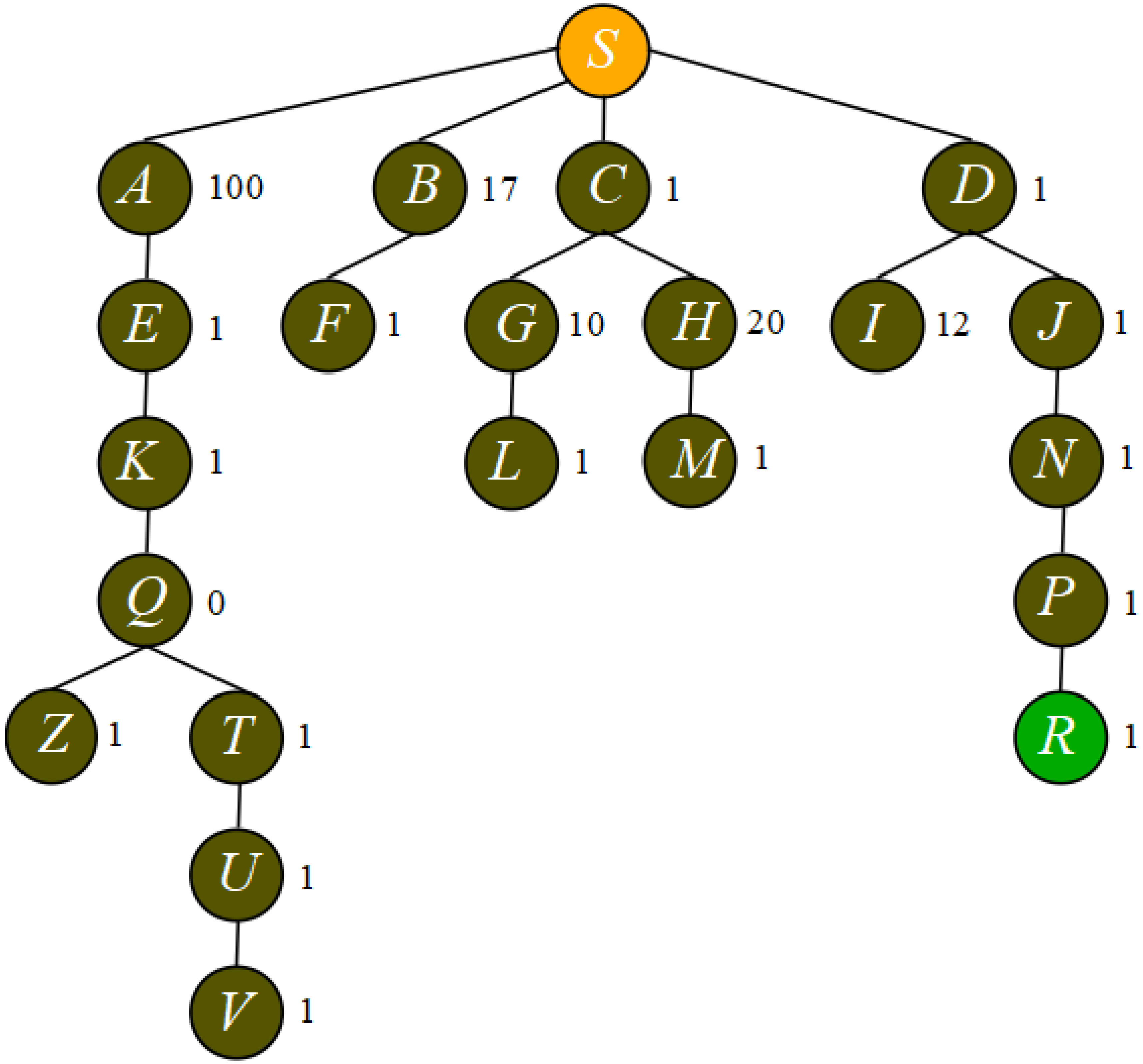
- Giải thuật AT (Algorithm for Tree)



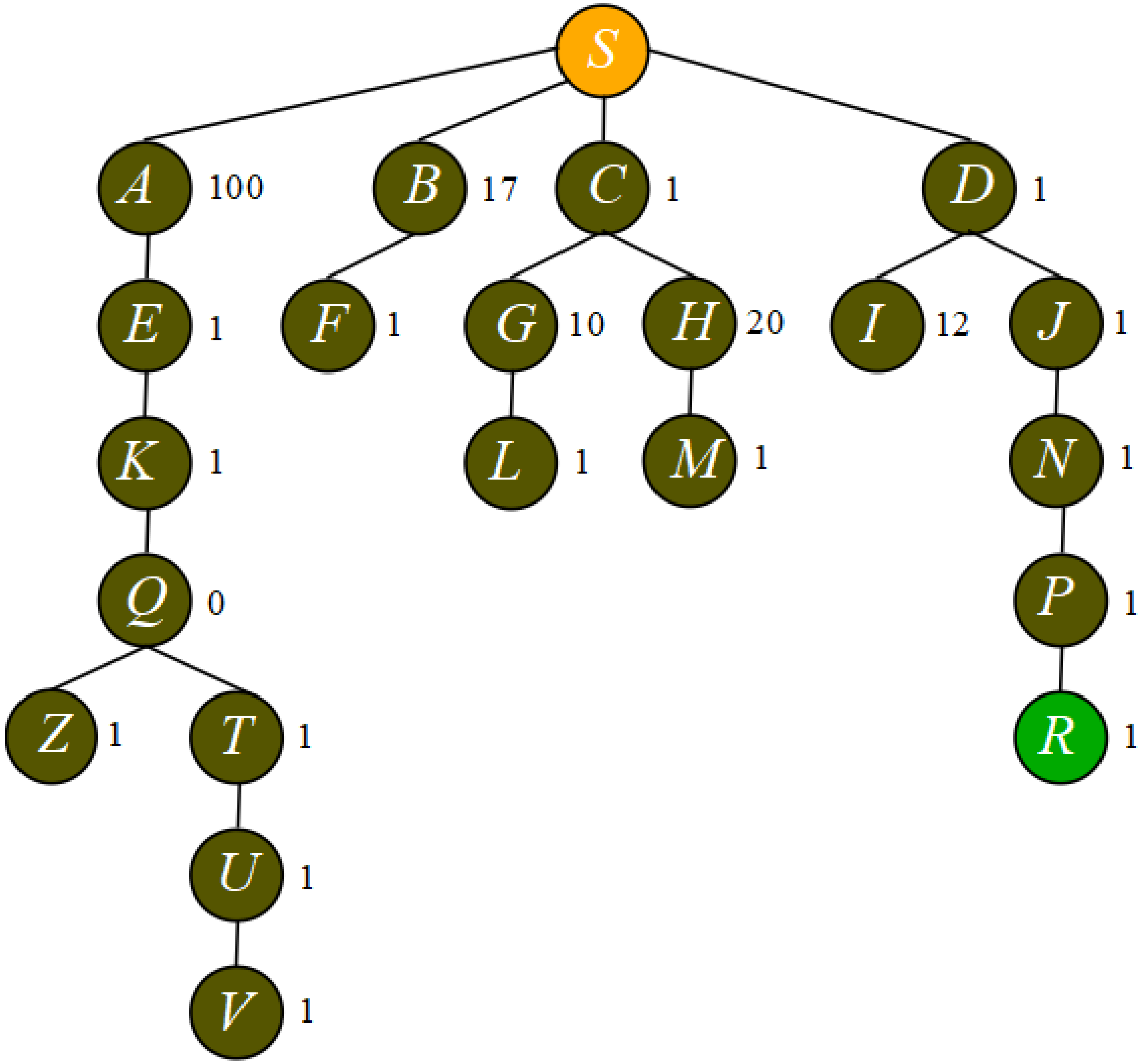
- Bước 3: n=C, Γ(n)={G,H}, CLOSE={S,C}
  - g(A) = 100
  - g(B) = 17
  - g(D) = 1
  - $g(G \notin OPEN) = g(C) + g(C \to G)$
    $= 1+10 = 11$
  - $g(H \notin OPEN) = g(C) + g(C \to H)$
    $= 1+20 = 21$
  - g(min) = g(D)
  - OPEN={D,G,B,C,A}

Giải thuật AT (Algorithm for Tree)

- Bước 4: n=D, Γ(n)={I,J}, CLOSE={S,C,D}
  - g(A)= 100
  - g(B)= 17
  - g(G)= 11
  - g(H)= 21
  - g(I$\notin$OPEN)=g(D) + g(D$\rightarrow$I) = 1+ 12 = 13
  - g(J$\notin$OPEN)=g(D) + g(D$\rightarrow$J) = 1+ 1 = 2
  - g(min) = g(J)
  - OPEN={J,G,I,B,H,A}



Giải thuật AT (Algorithm for Tree)

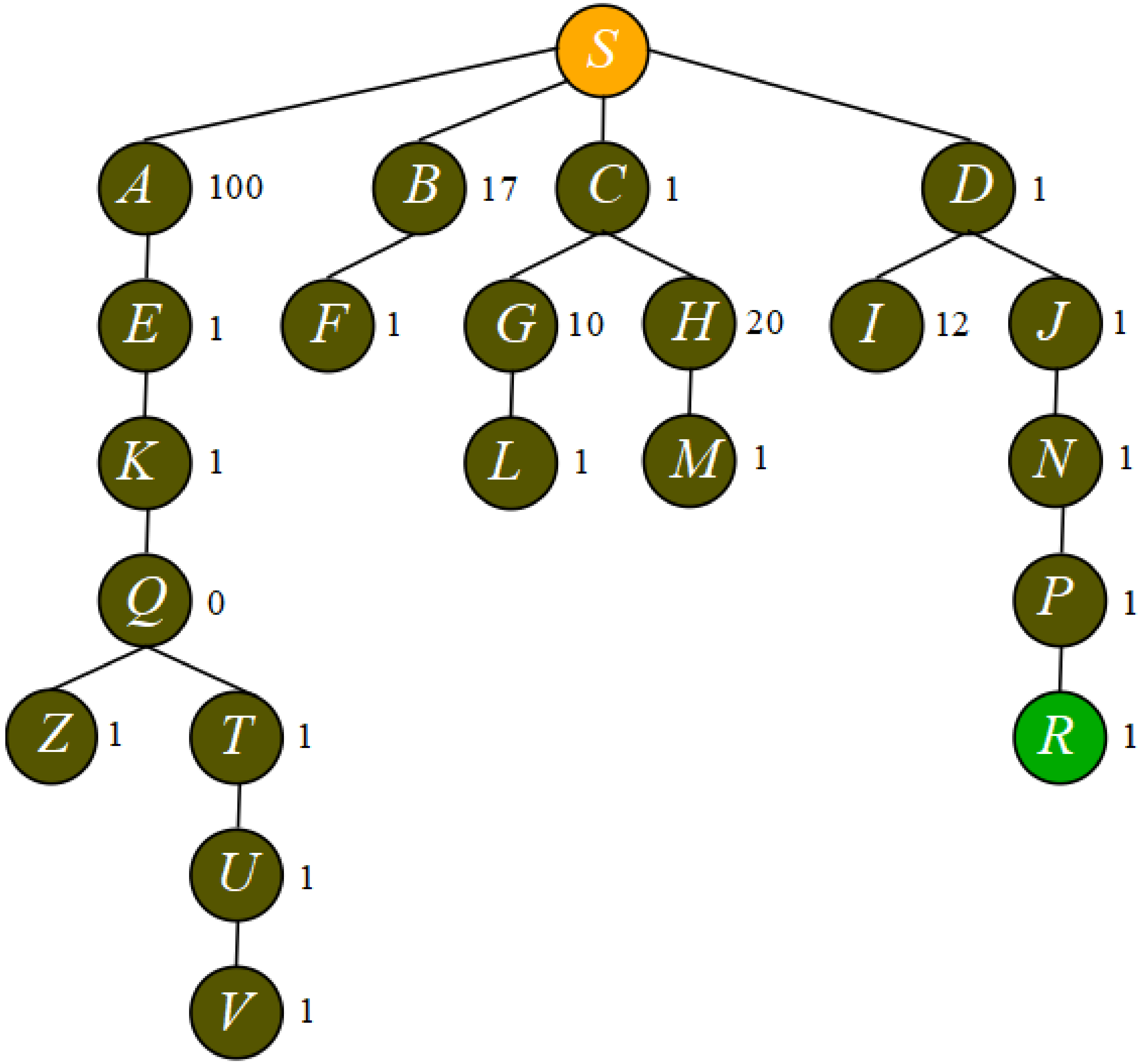


- Bước 5: n=J, Γ(n)={N}, CLOSE={S,C,D,J}
  - g(A)= 100
  - g(B)= 17
  - g(G)= 11
  - g(H)= 21
  - g(I)= 13
  - g(N$\notin$OPEN)=g(J) + g(J$\rightarrow$N) = 2+1 = 3
  - g(min) = g(N)
  - OPEN={N,G,I,B,H,A}

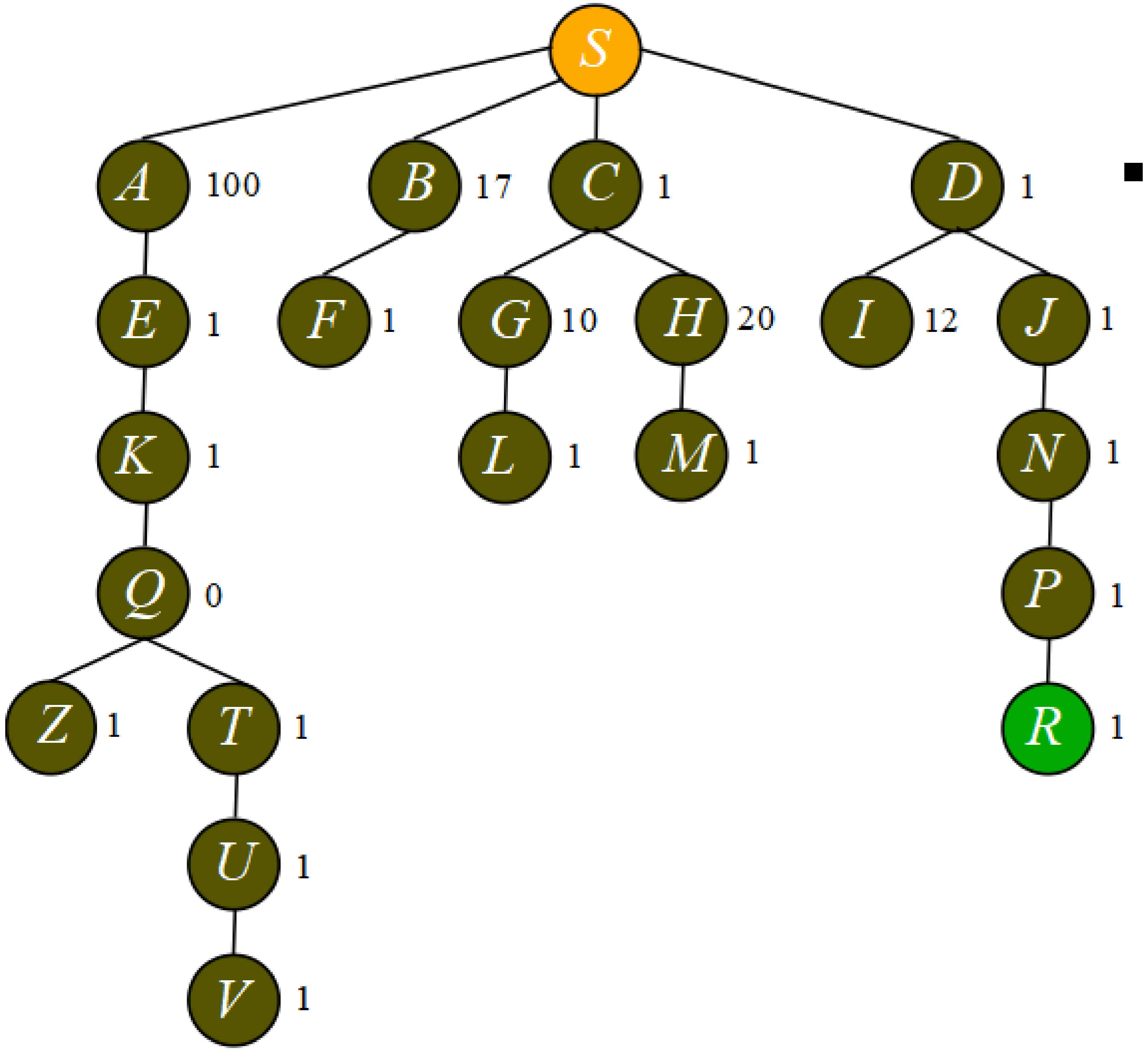
Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật AT (Algorithm for Tree)

- Bước 6: n=N, Γ(n)={P}, CLOSE={S,C,D,J,N}
  - g(A)= 100
  - g(B)= 17
  - g(G)= 11
  - g(H)= 21
  - g(I)= 13
  - g(P$\notin$OPEN)=g(N) + g(N$\rightarrow$P) = 3+ 1 = 4
  - g(min) = g(P)
  - OPEN={P,G,I,B,H,A}



▪ Giải thuật AT (Algorithm for Tree)



- Bước 7: n=P, Γ(n)={R}, CLOSE={S,C,D,J,N,P}
  - g(A)= 100
  - g(B)= 17
  - g(G)= 11
  - g(H)= 21
  - g(I)= 13
  - g(R$\notin$OPEN)=g(P) + g(P$\rightarrow$R) = 4+1 = 5
  - g(min) = g(R)
  - OPEN={R,G,I,B,H,A}

- Bước 8: n=R TRUE

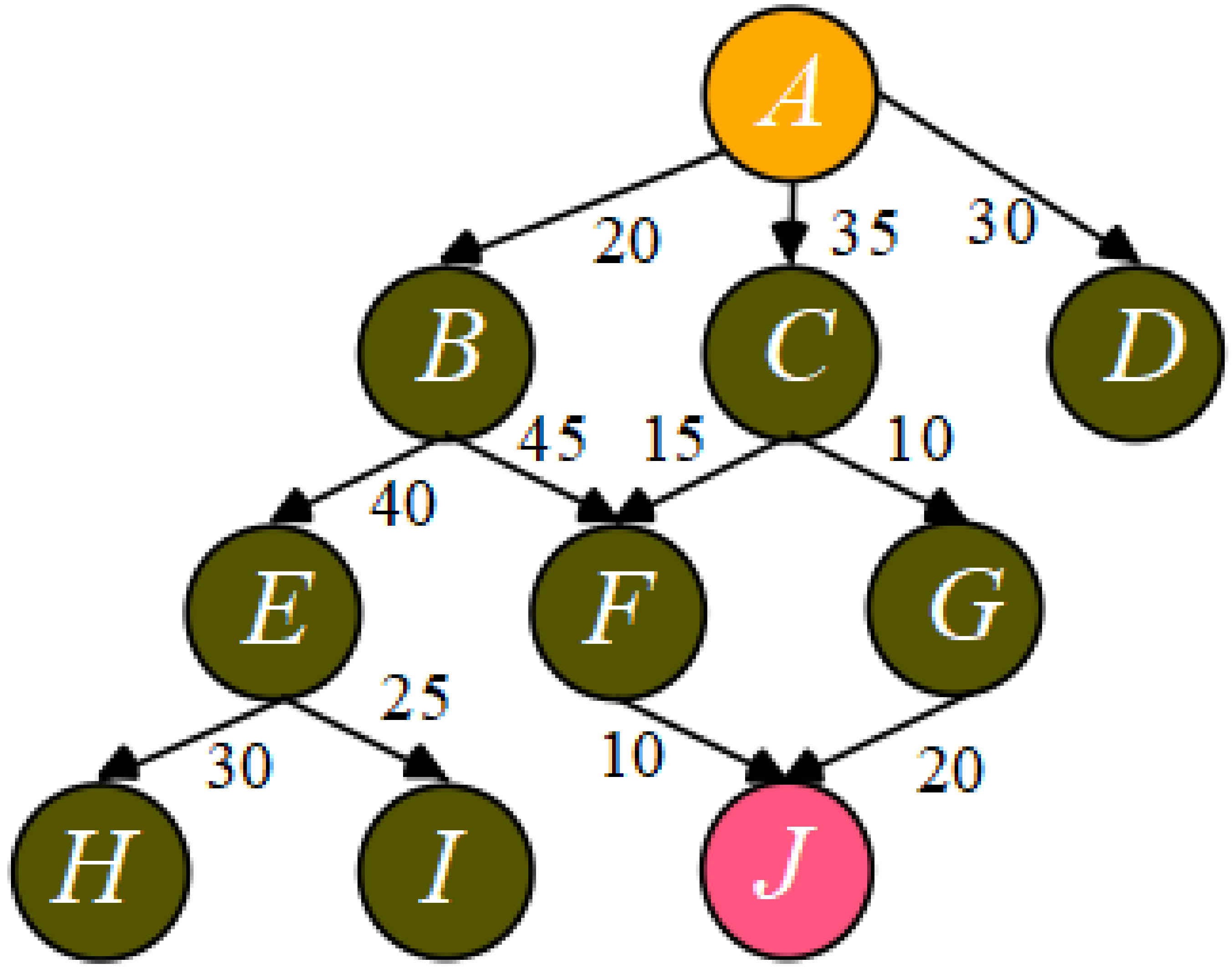
Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật Cost Minimazation Search (CMS)

- Bước 1: Chọn s = đỉnh xuất phát; g(s)=0
- Bước 2: Chọn 1 đỉnh từ tập OPEN: =n, có g(n) = min
  - 2.1. Nếu n ≡ đích: đường đi từ s ⇨ n là tối ưu
  - 2.2. Nếu OPEN = ∅: không tìm thấy đường đi
  - 2.3. Nếu ∃! nhiều hơn 1 đỉnh n có g(n) = min:
    - Kiểm tra (∃! ≡ đích): dừng
    - Ngược lại, chọn ngẫu nhiên một đỉnh: =n
- Bước 3: Đưa đỉnh đã duyệt vào CLOSE, tạo Γ(n): các đỉnh mà n trỏ tới
  - for mỗi nút con m của Γ(n):
    - $g(m) := g(n) + cost(n, m)$
    - $OPEN := OPEN \cup \{m\}$
    - So sánh g(m) với $g_{NEW}(m)$ và cập nhật
- Bước 4: Quay lại bước 2

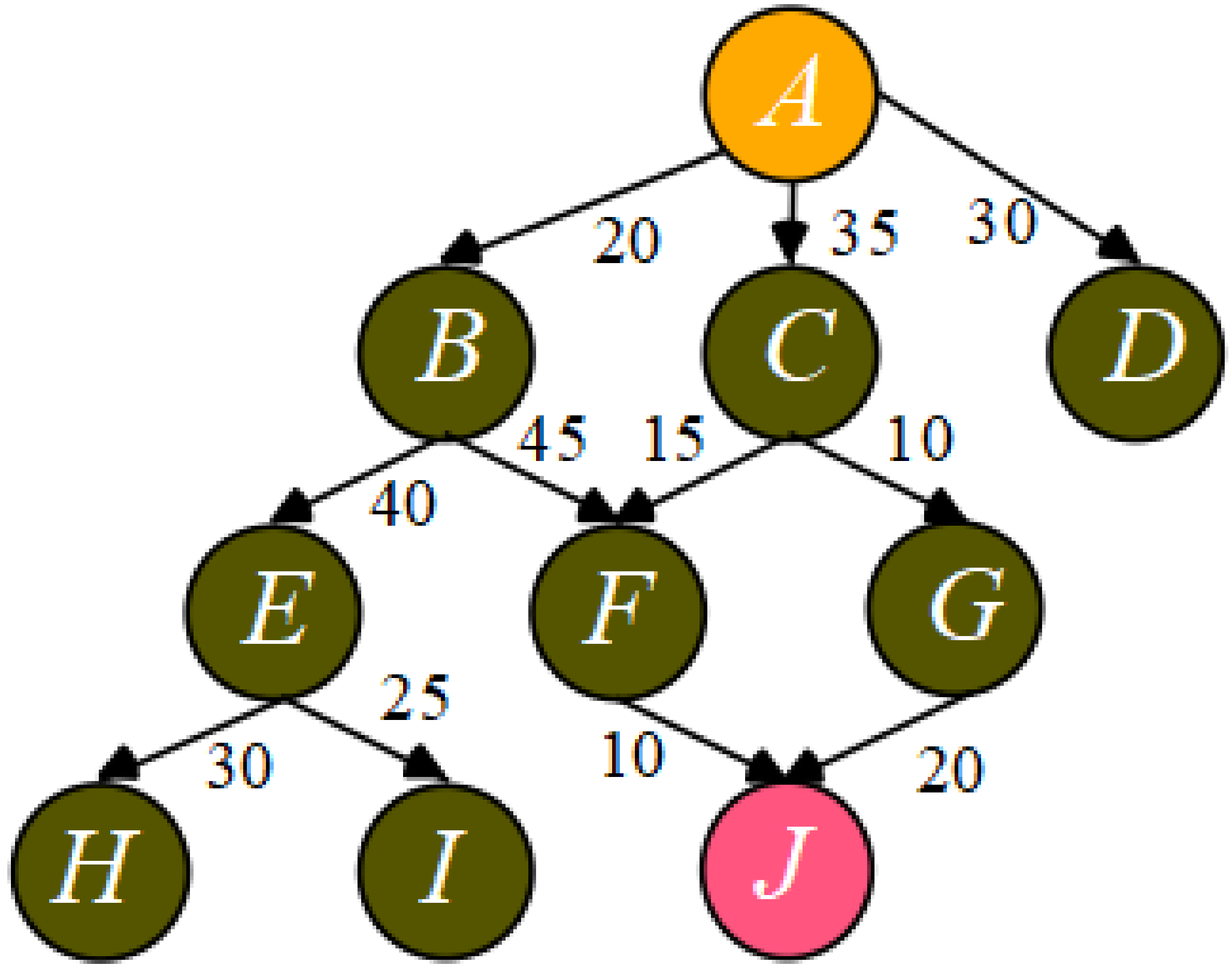
▪ Giải thuật Cost Minimazation Search (CMS)

- Trình bày thuật toán CMS để tìm đường đi ngắn nhất từ đỉnh **A** đến **J** trên đồ thị với các ước lượng heuristic của các trạng thái so với trạng thái đích được liệt kê như hình



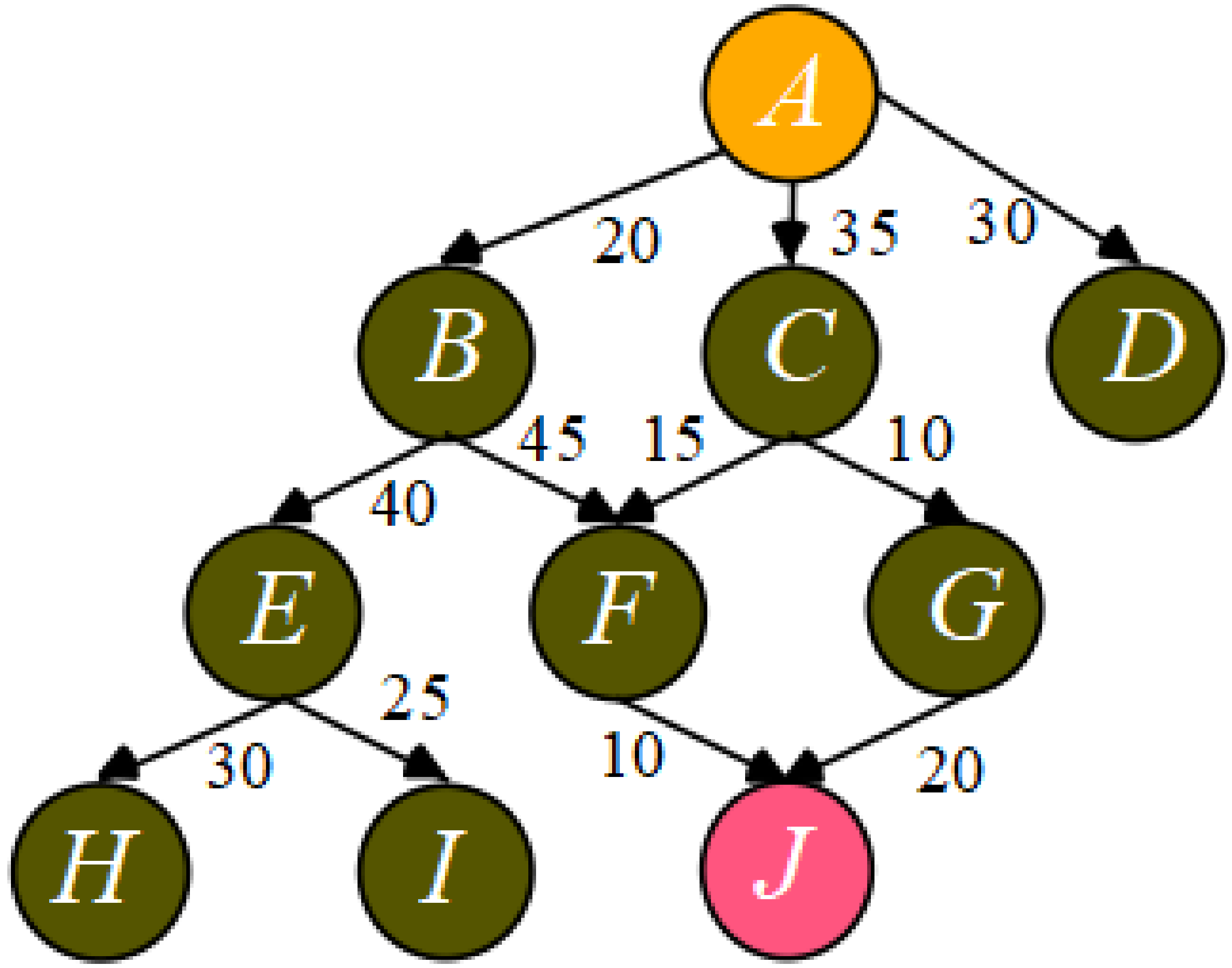
▪ Giải thuật Cost Minimazation Search (CMS)

- Bước 1: OPEN(A); g(A)=0
- Bước 2: n=A, Γ(n)={B,C,D}, CLOSE={A}
  - g(B) = g(A) + g(A→B) = 0+20 = 20; Father(B) = A
  - g(C) = g(A) + g(A→C) = 0+35= 35; Father(C) = A
  - g(D) = g(A) + g(A→D) = 0+30 = 30; Father(D) = A
  - g(min) = g(B)
  - OPEN={B,D,C}



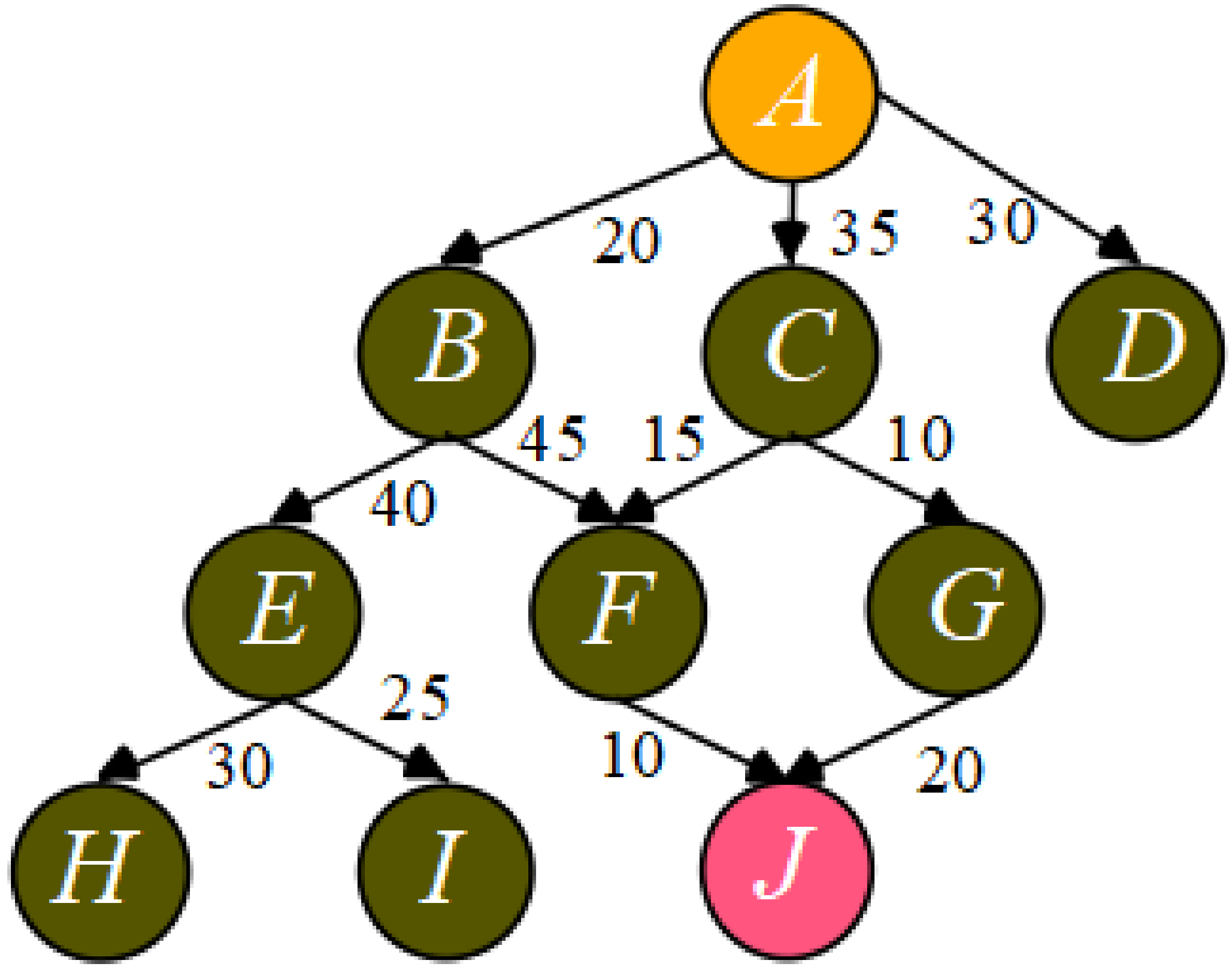
▪ Giải thuật Cost Minimazation Search (CMS)

- Bước 3: n=B, Γ(n)={E,F}, CLOSE={A,B}
  - g(C) = 35
  - g(D) = 30
  - g(E) = g(E) + g(B→E) = 20+40 = 60 ; Father(E) = B
  - g(F) = g(B) + g(B→F) = 20+45= 65 ; Father(F) = B
  - g(min) = g(D)
  - OPEN={D,C,E,F}



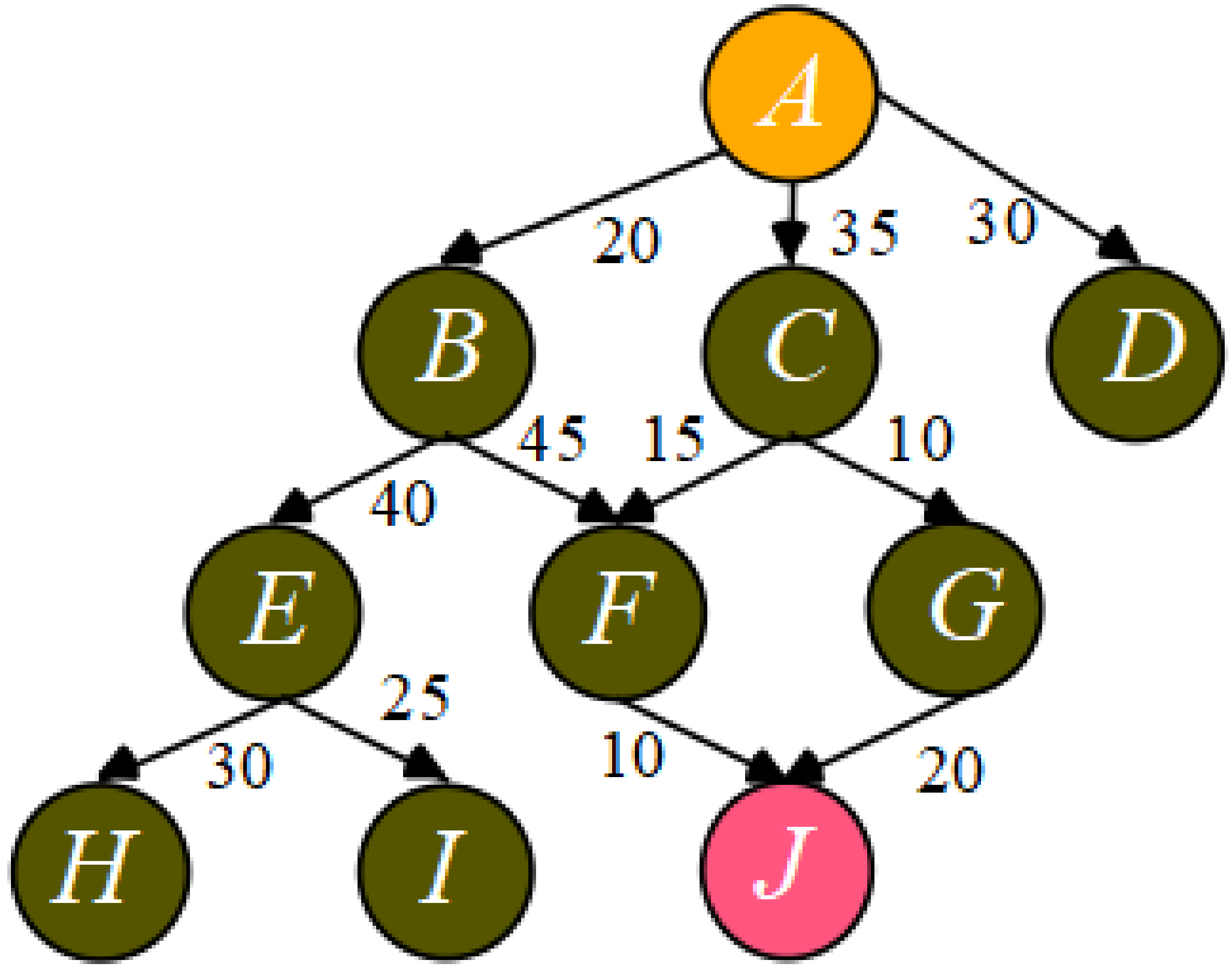
Giải thuật Cost Minimazation Search (CMS)

- Bước 4: n=D, $\Gamma(n)=\varnothing$, CLOSE={A,B,D}
  - g(C) = 35
  - g(E) = 60
  - g(F) = 65
  - OPEN={C,E,F}
- Bước 5: n=C, $\Gamma(n)$={F,G}, CLOSE={A,B,D,C}
  - g(E) = 60
  - g(F) = 65
  - g(F) =g(C) + g(C$\rightarrow$F) = 35+10= 50
    - g(F) = min(50,65) = 50; Father(F) = C
  - g(G) =g(C) + g(C$\rightarrow$G) = 35+10= 45; Father(G) = C
  - g(min) = g(G)
  - OPEN={G,F,E}



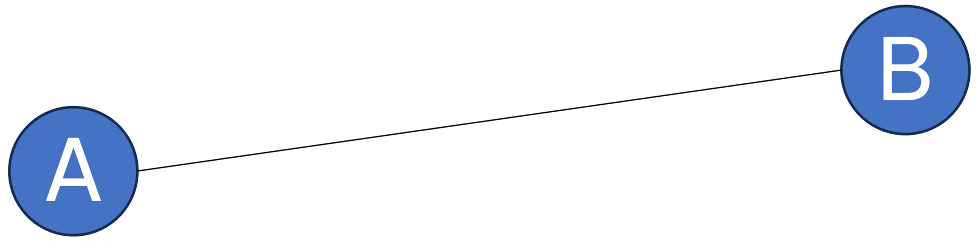
Giải thuật Cost Minimazation Search (CMS)

- Bước 6: n=G, Γ(n)={J}, CLOSE={A,B,D,C,G}
  - g(E) = 60
  - g(F) = 65
  - g(J) = g(G) + g(G→J) = 45+20= 65 ; Father(J) = G
  - OPEN={F,E,J}
- Bước 7: n=F, Γ(n)={J}, CLOSE={A,B,D,C,G,F}
  - g(E) = 60
  - g(J) = 65
  - g(J) =g(G) + g(G→J) = 45+20= 65
  - g(J) = min(65,65) = 65; Father(J) = F
  - OPEN={E,J}
- Bước 8: n=J, TRUE

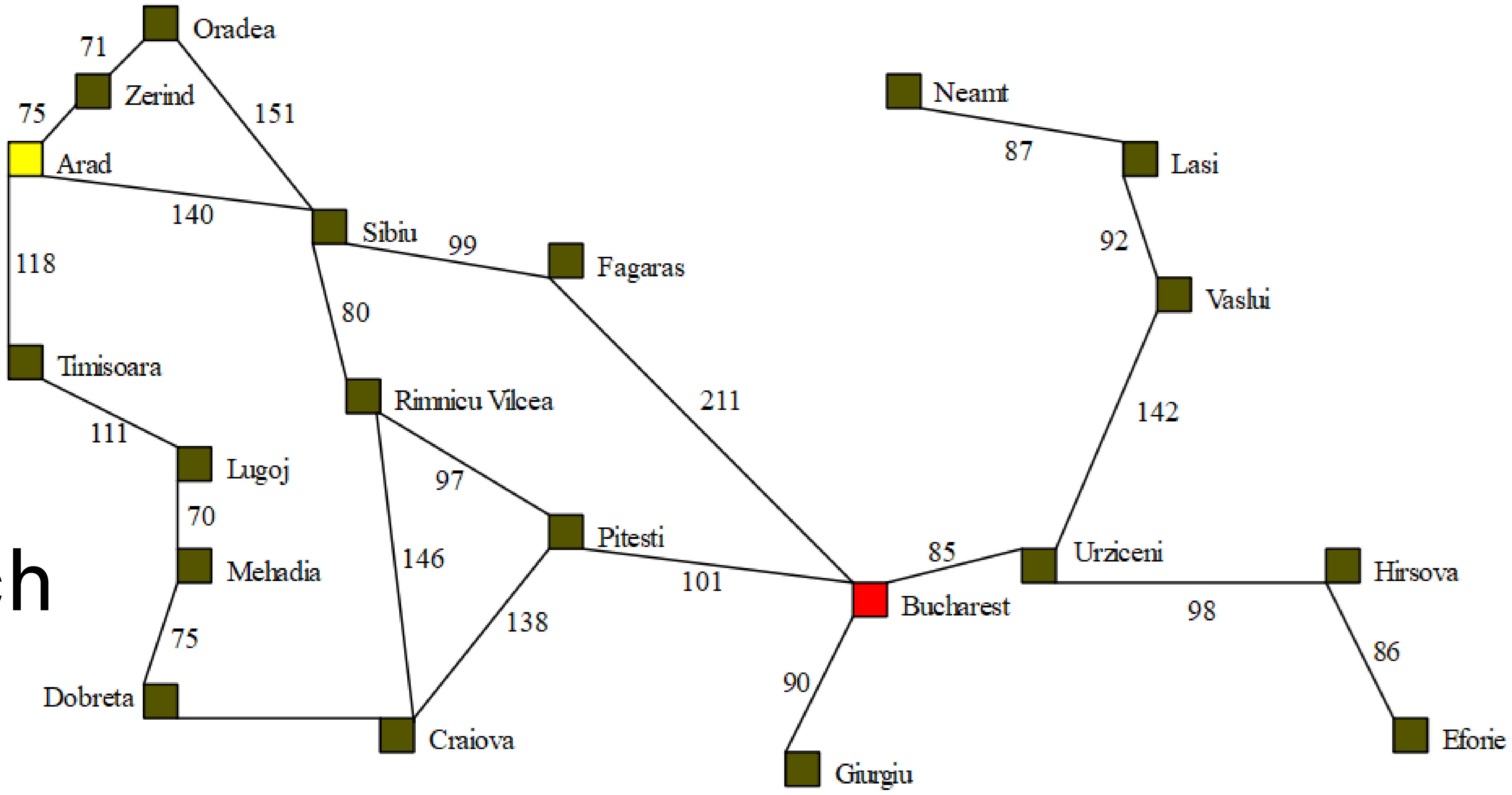


- A ⇨ C ⇨F ⇨J

- Algorithm for Knowlegeable Tree Search (AKT)
- Thuật giải A^{KT} là mở rộng của giải thuật A^{T} bằng cách sử dụng thêm thông tin ước lượng h’ (khoảng cách Manhattan).
- Độ tốt của hàm f = g+ h’



$$h' = |X_A - X_B| + |Y_A - Y_B|$$

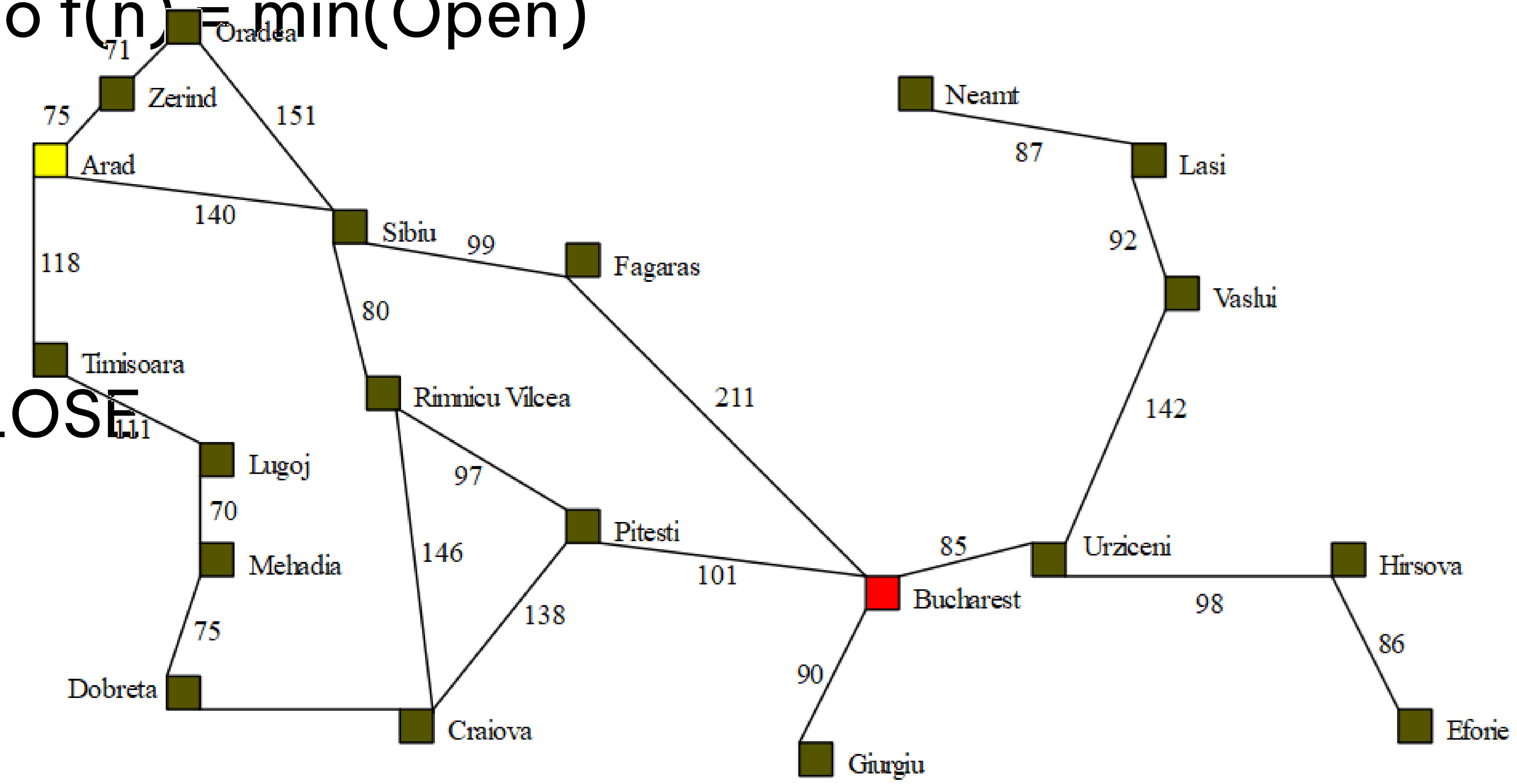


- h’ còn được gọi là khoảng cách Euclid

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

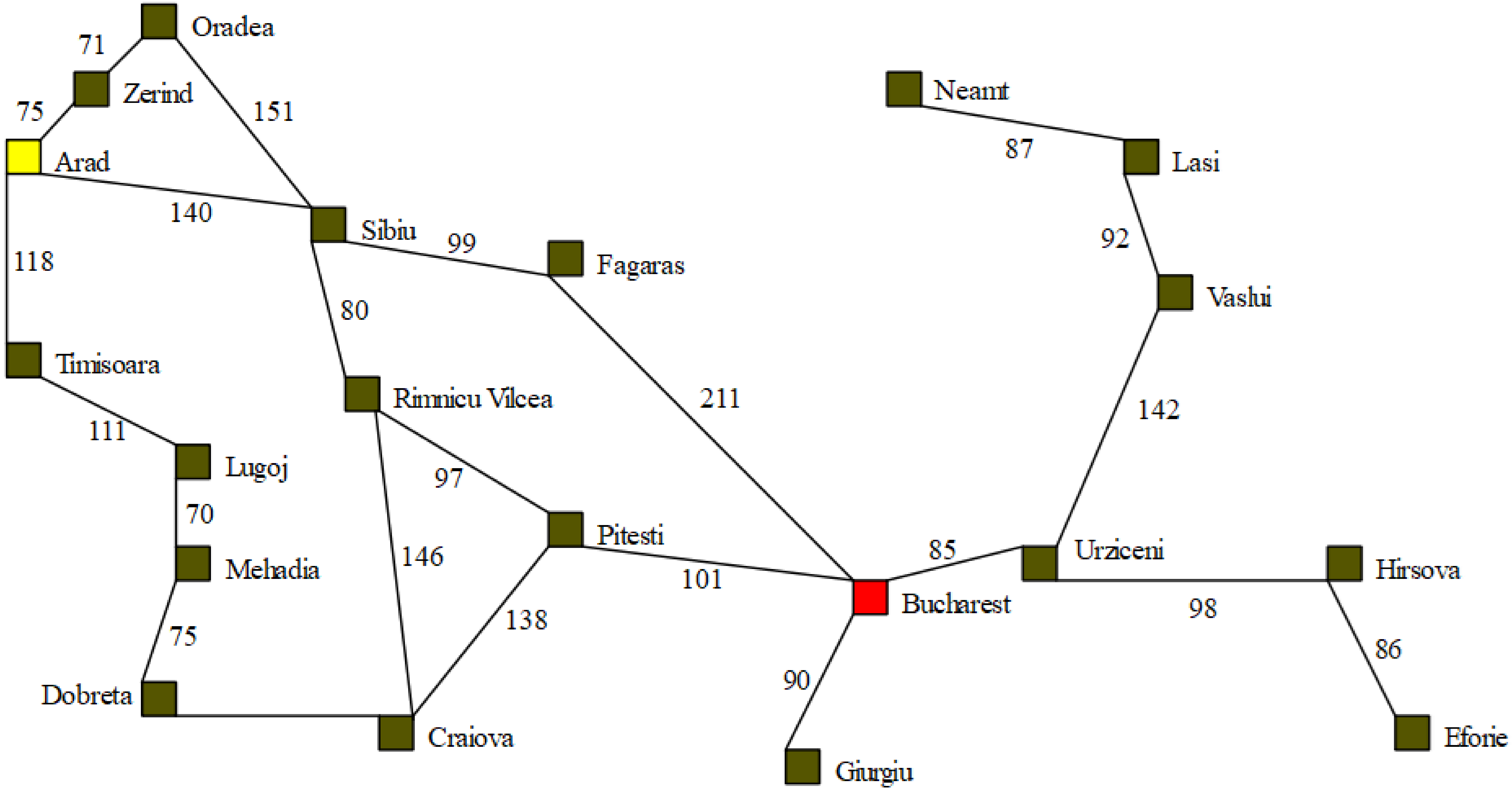
Algorithm for Knowlegeable Tree Search (AKT)

- Open:={Start};
- While (Open <> ∅)
  - n= Retrieve(Open);// Chọn n sao cho f(n) = min(Open)
  - If (n ≡ Goal): Return True
  - Else
    - Tạo Γ(n);
    - for mỗi nút con m của Γ(n) ∉ CLOSE
      - g(m)=g(n)+Cost(n,m)
      - f(m)=g(m)+h'(m)
      - OPEN = OPEN ∪ {m}
      - CLOSE = CLOSE ∪ {n}
- Return False;



- Algorithm for Knowlegeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitestti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



▪ Algorithm for Knowlegeable Tree Search (AKT)

▪ Tìm đường đi ngắn nhất từ Arad → Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitestti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- Start = Arad, Γ(n) ={Timisoara, Sibiu, Zerind},
- CLOSE = $\varnothing$
  - g(Timisoara) = g(Arad) + cost(Arad → Timisoara) = 0 + 118 = 118
  - f(Timisoara) = g(Timisoara) + h'(Timisoara) = 118 + 329 = 447
  - g(Sibiu) = g(Arad) + cost(Arad → Sibiu) = 0 + 140 = 140
  - f(Sibiu) = g(Sibiu) + h'(Sibiu) = 140 + 253 = 393
  - g(Zerind) = g(Arad) + cost(Arad → Zerind) = 0 + 75 = 75
  - f(Zerind) = g(Zerind) + h'(Zerind) = 75 + 374 = 449
- OPEN= {Sibiu – 393, Timisoara – 447, Zerind-449}
  - fmin= f(Sibiu),
  - Father(Sibiu) = Arad
  - Father(Timisoara) = Arad
  - Father(Zerind) = Arad

- Algorithm for Knowlegeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitestti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- n = Sibiu, Γ(n) ={~~Arad~~, Fagaras, Oradea, Rimnicu Vilcea},
- CLOSE = {Arad, Sibiu}
  - g(Fagaras) = g(Sibiu) + cost(Sibiu → Fagaras) = 140 + 99 = 239
  - f(Fagaras) = g(Fagaras) + h'(Fagaras) = 239 + 178 = 417
  - g(Oradea) = g(Sibiu) + cost(Sibiu → Oradea) = 140 + 151 = 291
  - f(Oradea) = g(Oradea) + h'(Oradea) = 291 + 380 = 671
  - g(Rimnicu Vilcea) = g(Sibiu) + cost(Sibiu → Rim.Vilcea) = 140 + 80 = 220
  - f(Rimnicu Vilcea) = g(Fagaras) + h'(Rimnicu Vilcea) = 220 + 193 = 413
- OPEN= {Rimnicu Vilcea - 413, Fagaras – 417, Timisoara – 447, Zerind-449}
  - fmin= f(Rimnicu Vilcea),
  - Father(Fagaras) = Sibiu
  - Father(Oradea) = Sibiu
  - Father(Rimnicu Vilcea) = Sibiu

- Algorithm for Knowlegeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitestti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- n = Rimnicu Vilcea, Γ(n) ={~~Sibiu~~, Craiova, Pitestti},
- CLOSE = {Arad, Sibiu, Rimnicu Vilcea}
  - g(Craiova) = g(R.Vilcea) + cost(R.Vilcea → Craiova) = 220 + 146 = 366
  - f(Craiova) = g(Craiova) + h'(Craiova) = 366 + 160 = 526
  - g(Pitestti) = g(R.Vilcea) + cost(R.Vilcea → Pitestti) = 220 + 97 = 317
  - f(Pitestti) = g(Pitestti) + h'(Pitestti) = 317 + 98 = 415
- OPEN= {Pitestti - 415, Fagaras – 417, Timisoara – 447, Zerind-449, Craiova - 526}
  - fmin= f(Pitestti),
  - Father(Craiova) = Rimnicu Vilcea
  - Father(Pitestti) = Rimnicu Vilcea

- Algorithm for Knowlegeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitestti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- n = Pitestti, Γ(n) ={~~Rimnicu Vilcea~~, Bucharest, Craiova},
- CLOSE = {Arad, Sibiu, Rimnicu Vilcea, Pitestti}
  - g(Bucharest) = g(Pitestti) + cost(Pitestti → Bucharest)
    = 317 + 101 = 418
  - f(Bucharest) = g(Bucharest) + h'(Bucharest) = 418 + 0 = 418
  - g_{new}(Craiova) = g(Pitestti) + cost(Pitestti → Craiova)
    = 317 + 138 = 455
  - f_{new}(Craiova) = g(Craiova) + h'(Craiova) = 455 + 160 = 615
- OPEN= {Fagaras – 417, Bucharest – 418, Timisoara – 447, Zerind-449, Craiova – 526 < f_{new}615}
  - fmin= f(Fagaras),
  - Father(Fagaras) = Sibiu (từ slide 28)
  - Father(Craiova) = Rimnicu Vilcea (vẫn giữ nguyên)

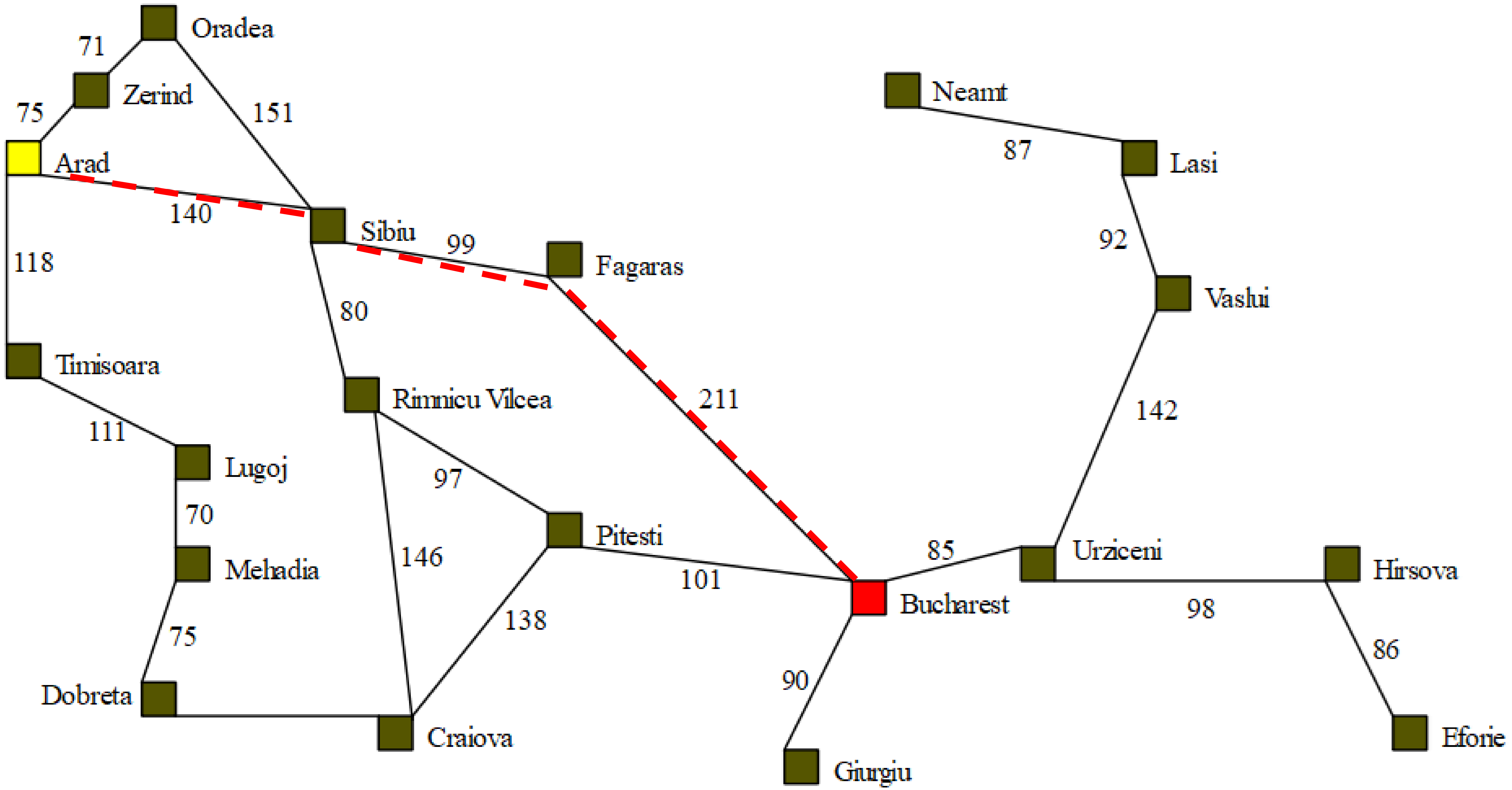
- Algorithm for Knowlegeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitestti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- n = Fagaras, Γ(n) ={~~Sibiu~~, Bucharest},
- CLOSE = {Arad, Sibiu, Rimnicu Vilcea, Pitestti, Fagaras}
  - g_{new}(Bucharest) = g(Fagaras) + cost(Fagaras → Bucharest)
    = 417 + 0 = 417
  - f_{new}(Bucharest) = g_{new}(Bucharest) + h'(Bucharest) = 417 + 0 = 417
- OPEN= {Bucharest_{new} – 417, Timisoara – 447, Zerind-449, Craiova – 526 < f_{new}615}
  - fmin= f(Bucharest),
  - Father(Bucharest) = Fagaras
- n = Bucharest, TRUE

- Algorithm for Knowlegeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitestti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ A star(A*)

- Thuật giải A* tránh việc xét các nhánh tìm kiếm đã xác định (cho đến thời điểm hiện tại) có chi phí cao.
- Sử dụng hàm đánh giá $f(u) = g(u) + h(u)$
  - $g(u)$: $chi\ phí\ của\ đường\ đi\ từ\ nút\ gốc \quad nút\ hiện\ tại$
  - $h(u)$: $ước\ lượng\ heuristic = khoảng\ cách\ Manhattan$
  - $f(u)$: $chi\ phí\ tổng\ thể\ ước\ lượng\ của\ đường\ đi\ qua\ nút\ hiện\ tại\ (u) \rightarrow đích$
- A* là phiên bản đặc biệt của A^{KT} áp dụng cho trường hợp đồ thị
- A* mở rộng A^{KT} bằng cách bổ sung cách giải quyết trường hợp khi mở một nút mà nút này đã có sẵn trong OPEN hoặc CLOSE.

A star(A*)

- OPEN: tập chứa các trạng thái đã được sinh ra nhưng chưa xét đến
  - Các phần tử được sắp xếp theo độ tốt (giá trị của hàm f)
  - Phần tử có độ ưu tiên cao nhất = phần tử tốt nhất
- CLOSE: Danh sách các trạng thái đã xét
- Hàm REMOVE(): Lấy phần tử ở đầu danh sách OPEN
- Hàm INSERT(): Chèn phần tử vào OPEN theo thứ tự ưu tiên
- Hàm FATHER(x): Trạng thái cha của X

$function\ A\_Star\_Search()\{$

Input: start, goal;

Output: Đường đi từ start $\rightarrow$ goal hoặc thất bại

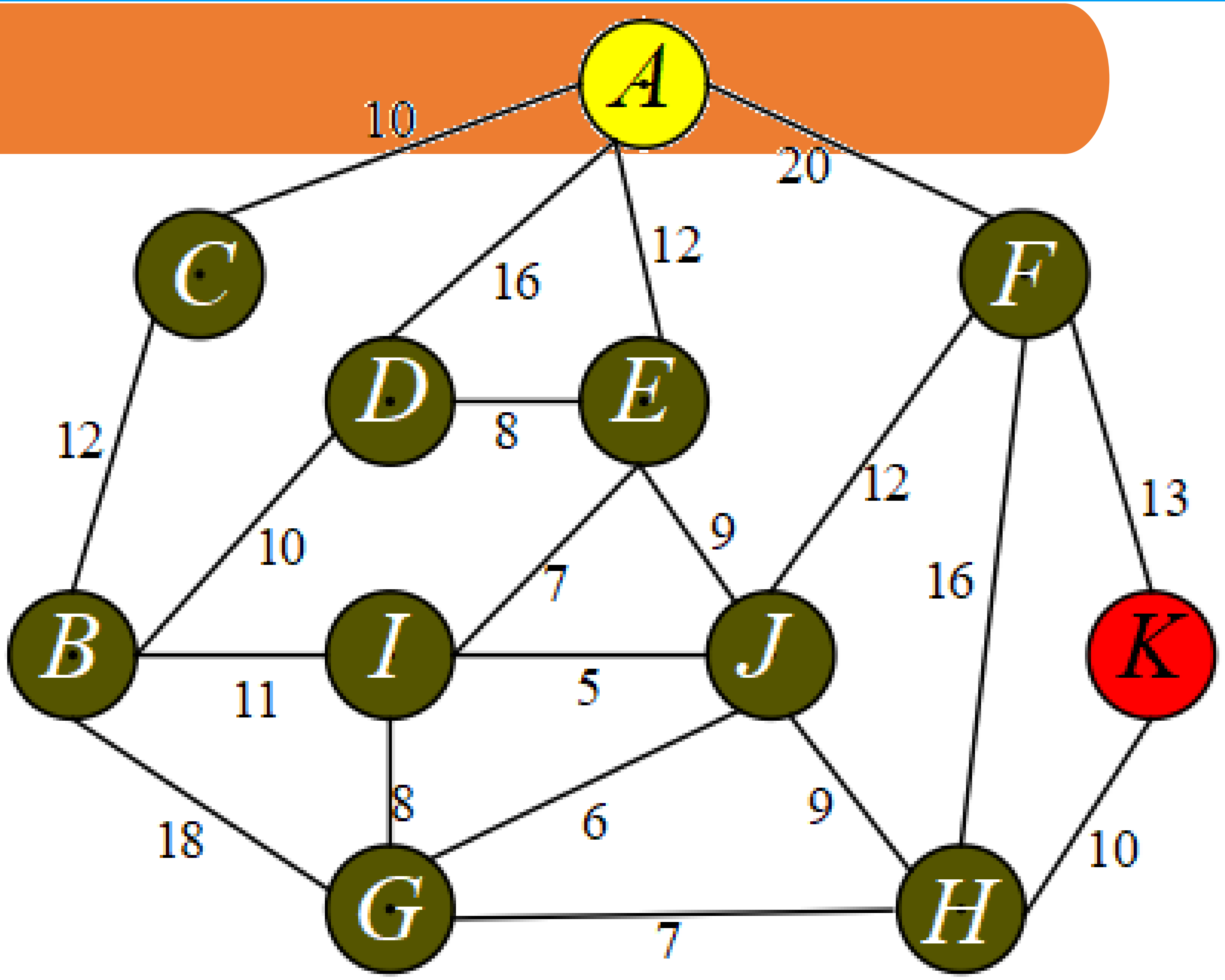
$\}$

A star(A*)

- OPEN = start; CLOSE = ∅; g(start) = 0; f(start) = h(start)
- Loop:
  - if (OPEN ≡ ∅): Thất bại
  - n = REMOVE(OPEN) *//phần tử đầu ds OPEN (f(n) == min)*
  - if (n ≡ goal): Tìm thấy đường đi
  - Ngược lại:
    - Đưa {n} vào danh sách CLOSE
    - for (mỗi trạng thái m là con của {n}:
      - Tính g(m) = g(n) + cost(n,m)
      - Tính f(m) = g(m) + h(m)
    - if (∃! t ∈ OPEN ≡ m): *(cond 1)*
    - elseif (∃! t ∈ CLOSE ≡ m): *(cond 2)*
    - *else INSERT(n,OPEN)//m ∉{OPEN,CLOSE}*

- if (∃! t ∈ OPEN ≡ m):
  - if ($f(m)_{\_new}$ < $f(t)_{\_old}$):
    - $g(t)_{\_old} = g(m)_{\_new}$
    - $f(t)_{\_old} = f(m)_{\_new}$
    - Sắp xếp lại OPEN
    - FATHER(t) = n
- elseif (∃! t ∈ CLOSE ≡ m):
  - if ($f(m)_{\_new}$ < $f(t)_{\_old}$):
    - $g(t)_{\_old} = g(m)_{\_new}$
    - $f(t)_{\_old} = f(m)_{\_new}$
    - FATHER(t) = n

- A star(A*)

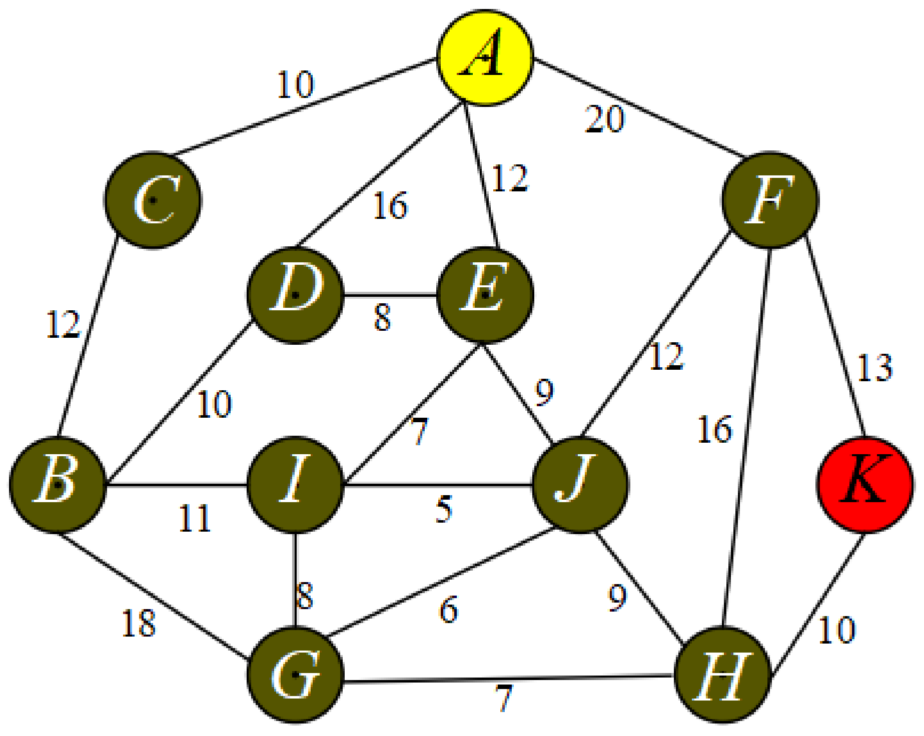


- Áp dụng giải thuật A* để tìm đường đi ngắn nhất A → K, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở bảng sau:

A	B	C	D	E	F	G	H	I	J	K
33	35	43	36	28	13	17	10	24	19	0

A star(A*)

- Start = {A}, goal = {K}, OPEN = {A}, CLOSE = ∅
- g(A) = 0, f(A) = h(A) = 33
- Bước 1. n=A, CLOSE={A}
  - g(C) = g(A)+cost(A,C) = 0+10 = 10
  - f(C) = g(C)+h(C) = 10+43 = 53
  - g(D) = g(A)+cost(A,D) = 0+16 = 16
  - f(D) = g(D)+h(D) = 16+36 = 49
  - g(E) = g(A)+cost(A,E) = 0+12 = 12
  - f(E) = g(E)+h(E) = 12+28 = 40
  - g(F) = g(A)+cost(A,F) = 0+20 = 20
  - f(F) = g(F)+h(F) = 20+13 = 33
  - {C,D,E,F}∉OPEN và {C,D,E,F} ∉ CLOSE
  - OPEN = {F(33), E(40), D(49), C(53)}

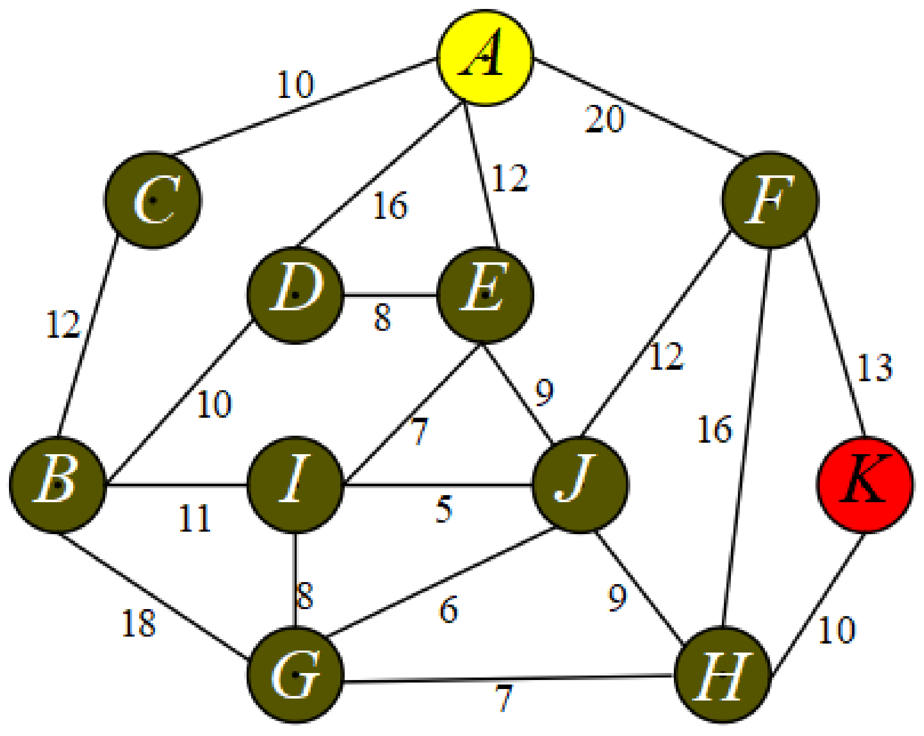


A	B	C	D	E	F	G	H	I	J	K
33	35	43	36	28	13	17	10	24	19	0

- Father(F, E, D, C) = A

▪ A star(A*)

- Bước 2. n=F, CLOSE={A,F}, OPEN = {E,D,C}
  - g_{new}(A) = g(F)+cost(F,A) = 20+20 = 40
  - f_{new}(A) = g(A)+h(A) = 40+33 = 73
    - t=A ∈ CLOSE, f_{new}(A) < f(t) (73 < 55): FALSE
    - g(A)=0, f(A)=33
  - g(J) = g(F)+cost(F,J) = 20+12 = 32
  - f(J) = g(J)+h(J) = 32+19 = 51
  - g(H) = g(F)+cost(F,H) = 20+16 = 36
  - f(H) = g(H)+h(H) = 36+10 = 46
  - g(K) = g(F)+cost(F,K) = 20+13 = 33
  - f(K) = g(K)+h(K) = 33+0 = 33
  - {H,J,K}∉OPEN và {H,J,K} ∉ CLOSE
  - OPEN = {K(33),E(40), H(46), D(49), J(51), C(53)



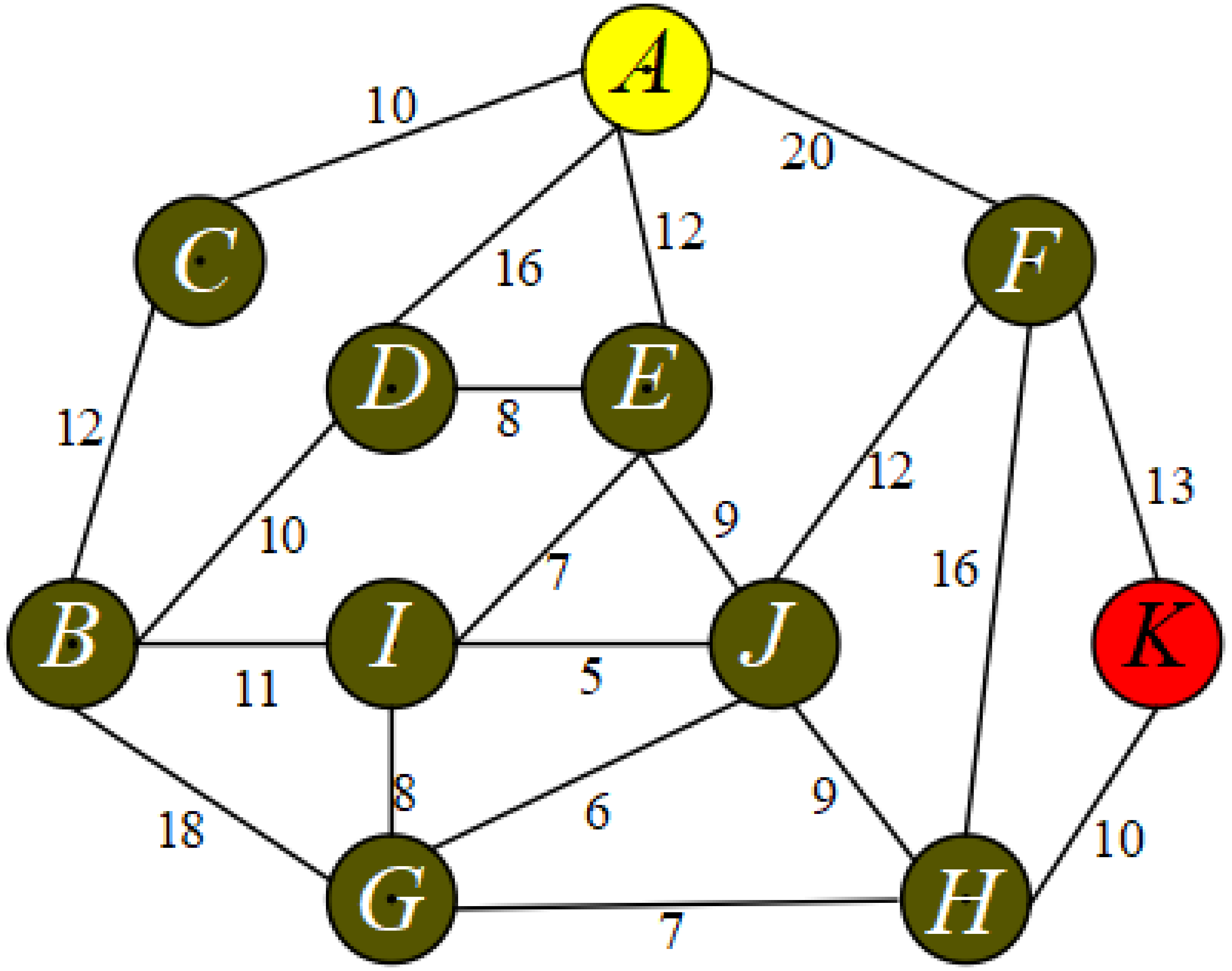
A	B	C	D	E	F	G	H	I	J	K
33	35	43	36	28	13	17	10	24	19	0

- Father(J, H, K) = F

A star(A*)

- Bước 3. n=K, TRUE
- Father(J, H, K) = F
- Father(F, E, D, C) = A

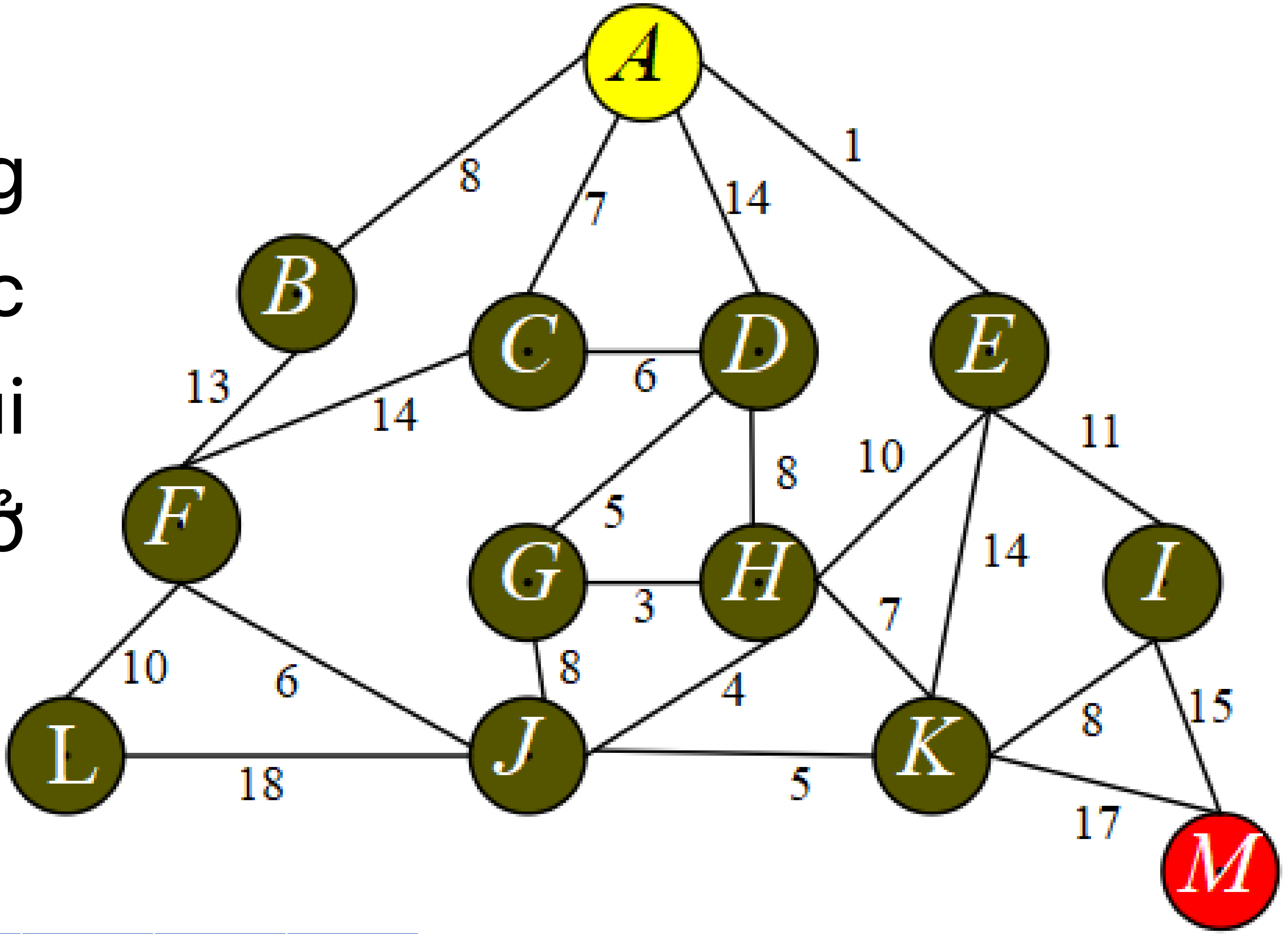
- A ⇨F ⇨K



A	B	C	D	E	F	G	H	I	J	K
33	35	43	36	28	13	17	10	24	19	0

- A star(A*)

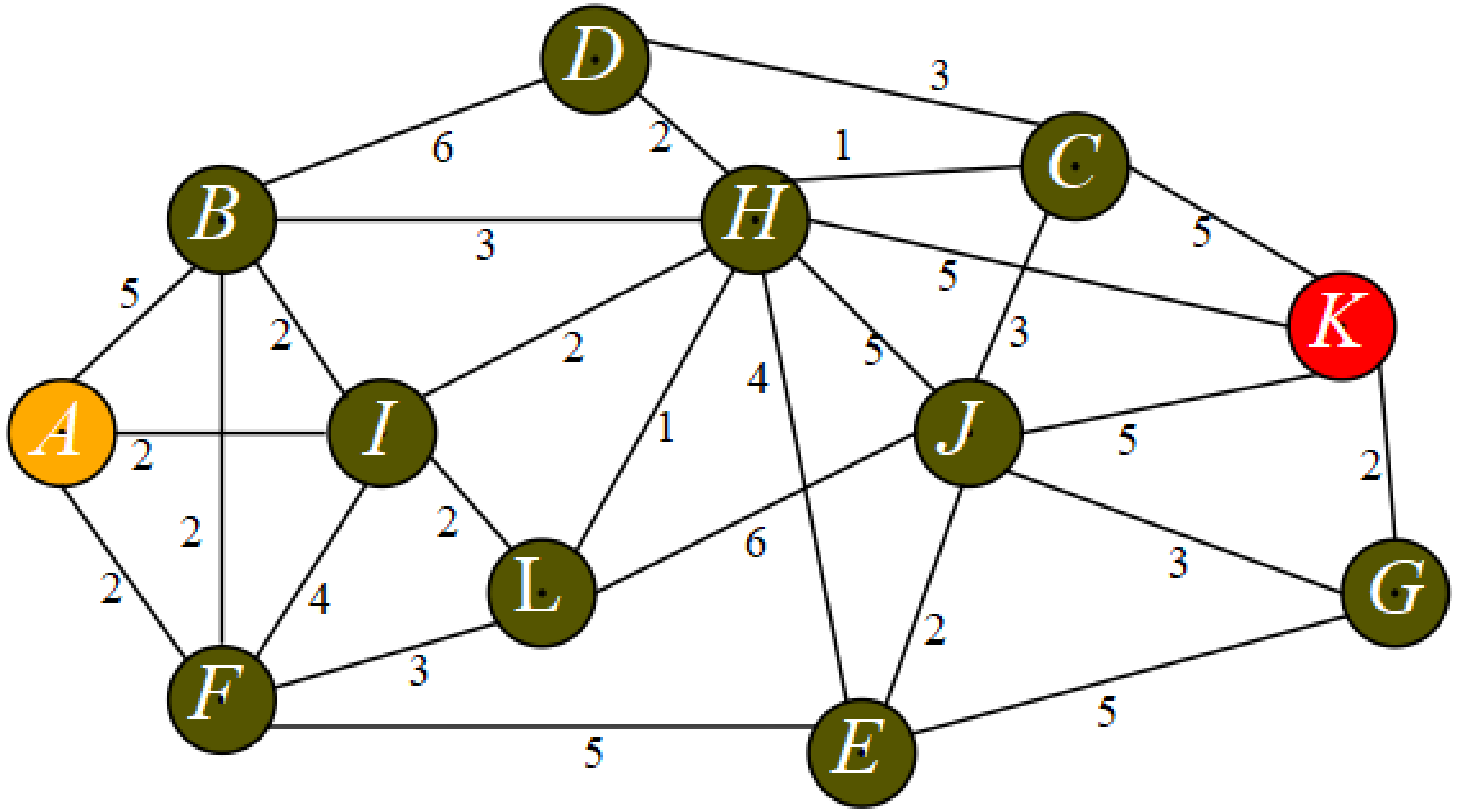
- Áp dụng giải thuật A* để tìm đường đi ngắn nhất A → M, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở bảng sau:



A	B	C	D	E	F	G	H	I	J	K	L	M
27	34	33	32	26	28	27	24	15	22	17	15	0

- A star(A*)

- Áp dụng giải thuật A* để tìm đường đi ngắn nhất A → K, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở bảng sau:



A	B	C	D	E	F	G	H	I	J	K	L
9	8	5	7	7	9	2	5	7	5	0	6

- Nhánh – cận (Branch-and-Bound)

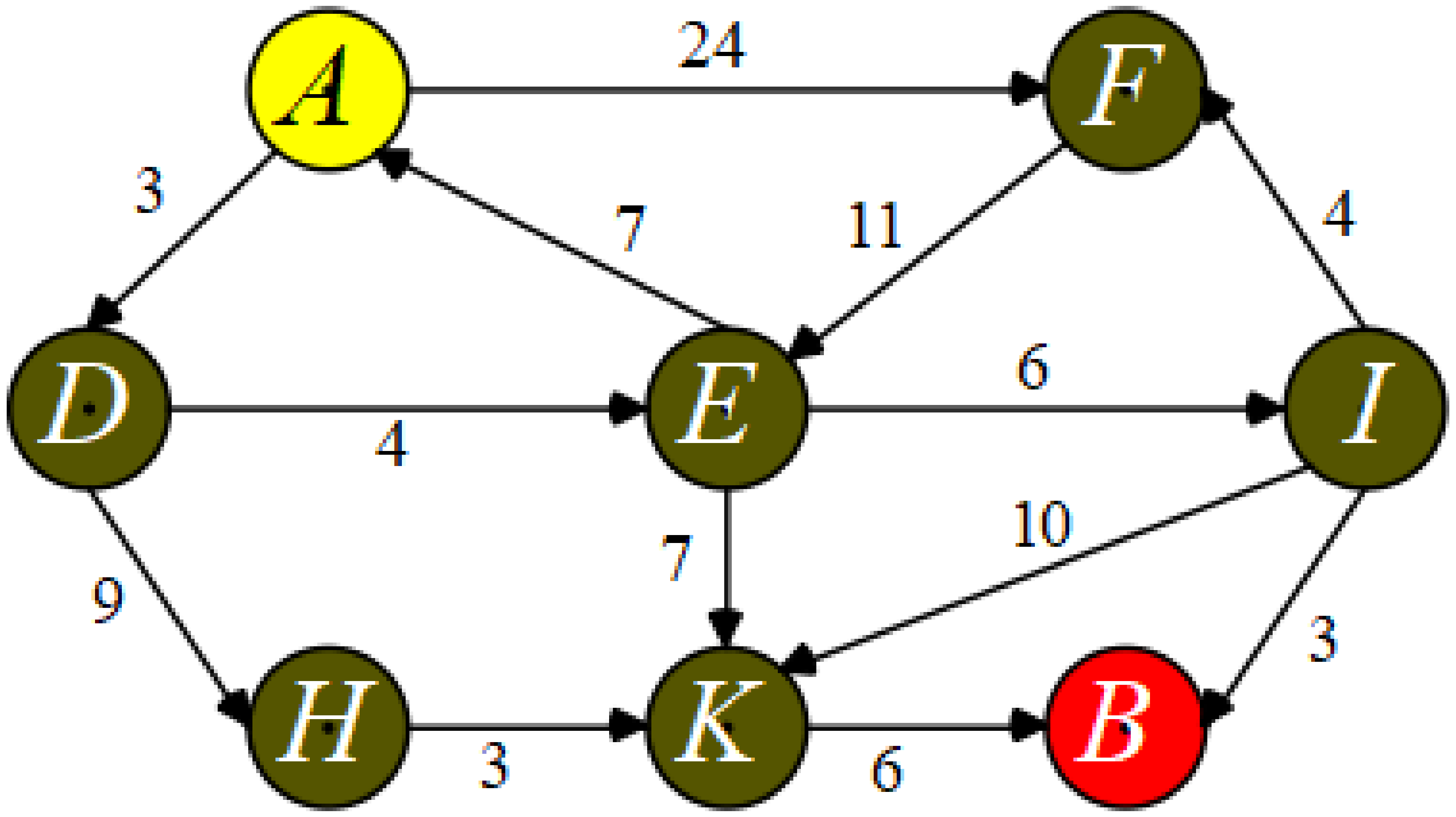
- Thuật toán tìm kiếm leo đồi kết hợp với hàm đánh giá f(u). Tại mỗi bước, khi phát triển trạng thái u, chọn trạng thái con v tốt nhất (f(v) nhỏ nhất) của u để phát triển ở bước sau.
- Quá trình tiếp tục như vậy cho đến khi gặp trạng thái w là đích, hoặc w không có đỉnh kề, hoặc w có f(w) lớn hơn độ dài đường đi tối ưu tạm thời. Trong các trường hợp này, chúng ta không phát triển đỉnh w nữa, tức là cắt bỏ những nhánh xuất phát từ w, và quay lên cha của w để tiếp tục đi xuống trạng thái tốt nhất trong số những trạng thái còn lại chưa được phát triển

Nhánh – cận (Branch-and-Bound)

- Bước 1. OPEN = {start}, min = $+\infty$
- Bước 2. Loop:
  - 2.1. if(OPEN $\equiv\emptyset$): STOP
  - 2.2. n:= lấy phần tử đầu danh sách OPEN
  - 2.3. if(n$\equiv$goal):
    - if (f(n) < min): {min=f(n); quay lại 2.1 //cập nhật lại min}
  - 2.4. if (f(n) > min): quay lại 2.1 //cắt bỏ nhánh con
  - 2.5. for ($\forall$m: đỉnh kề n):
    - g(m) = g(n) + cost(m,n)
    - f(m) = g(m)+h(m)
    - Đưa m vào danh sách Γ(n)
  - 2.6. Sắp xếp Γ(n) theo thứ tự tăng dần của f
  - 2.7. Chèn Γ(n) vào đầu OPEN

▪ Nhánh – cận (Branch-and-Bound)

- Áp dụng thuật toán nhánh cận để tìm đường đi ngắn nhất từ đỉnh A → I, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở Bảng.



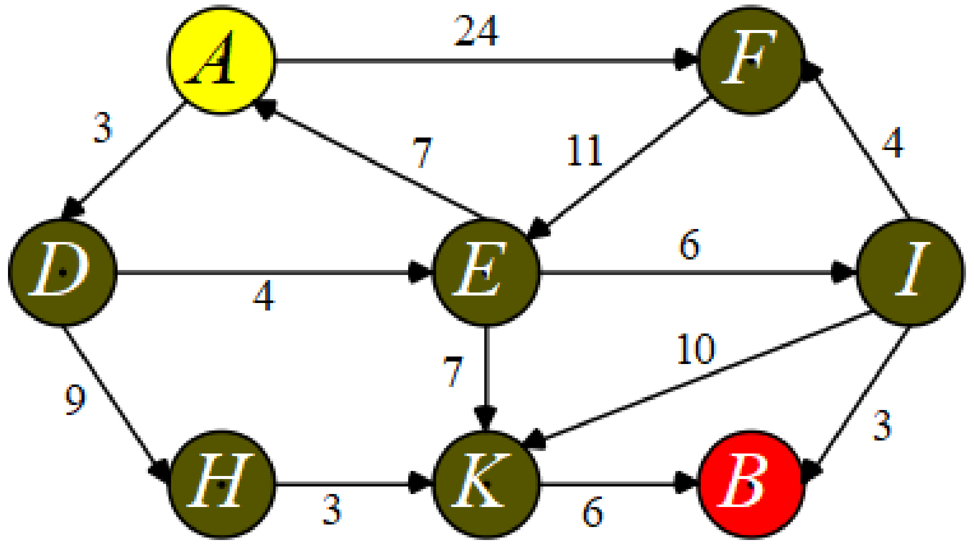
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

▪ Nhánh – cận (Branch-and-Bound)

Bước	n	m	g(m)=g(n) + cost(n,m)	f(m)=g(m) + h(m)	$\Gamma(n)\uparrow$	OPEN	min
0			g(A)=0	f(A) = h(A) = 13	$\varnothing$	A	$+\infty$
1	A f(A)<min	D F	g(D) = g(A) + cost(A,D) = 0 + 3 = 3 g(F) = g(A) + cost(A,F) = 0 + 24 = 24	f(D) = g(D) + h(D) = 3 + 2 = 5 f(F) = g(F) + h(F) = 24 + 9 = 33	$D^{4} \rightarrow F^{33}$	D^{4}, F^{33}	$+\infty$

- Father(D,F) = A

VET							
A	B	D	E	F	H	I	K
1	-	-	-	-	-	-	-

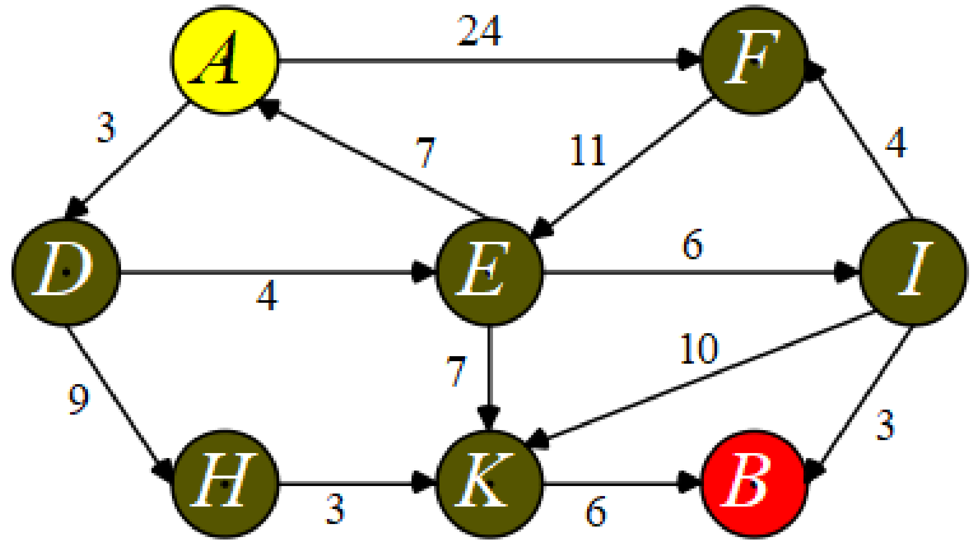


HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

▪ Nhánh – cận (Branch-and-Bound)

Bước	n	m	g(m)=g(n) + cost(n,m)	f(m)=g(m) + h(m)	Γ(n)↑	OPEN	min
2	D f(D)<min	E H	g(E) = g(D) + cost(D,E) = 3 + 4 = 7 g(H) = g(D) + cost(D,H) = 3 + 9 = 12	f(E) = g(E) + h(E) = 7 + 8 = 15 f(H) = g(H) + h(H) = 12 + 7 = 19	$E^{15}\rightarrow$ H^{19}	E^{15}, H^{19}, F^{33}	$+\infty$

- Father(D,F) = A
- Father(E,H) = D



VET							
A	B	D	E	F	H	I	K
1	-	2	-	-	-	-	-

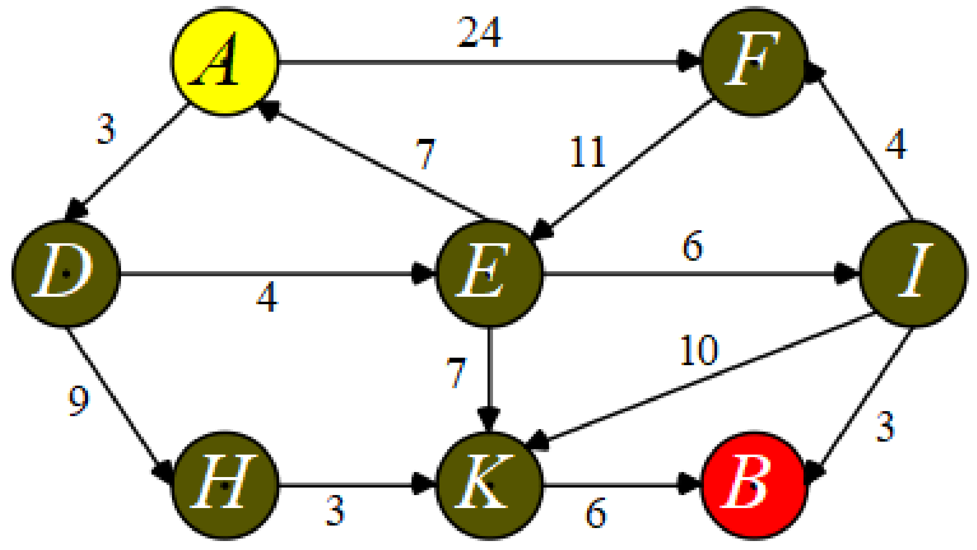
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Nhánh – cận (Branch-and-Bound)

Bước	n	m	g(m)=g(n) + cost(n,m)	f(m)=g(m) + h(m)	Γ(n)↑	OPEN	min
3	E f(E)<min	I K A	g(I) = g(E) + cost(E,I) = 7 + 6 = 13 g(K) = g(E) + cost(E,K) = 7 + 7 = 14 g(A) = g(E) + cost(E,A) = 7 + 7 = 14	f(I) = g(I) + h(I) = 13 + 6 = 19 f(K) = g(K) + h(K) = 14 + 4 = 18 f(A) = g(A) + h(A) = 14 + 13 = 26	K^{18}→ I^{19}→ A^{26}	K^{18}, I^{19}, A^{26}, H^{19}, F^{33}	+∞

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	-

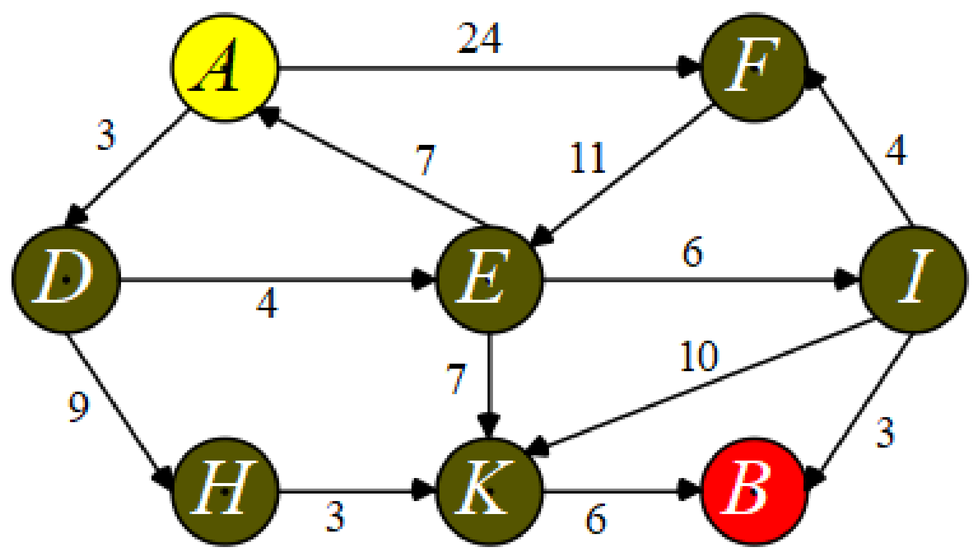
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Nhánh – cận (Branch-and-Bound)

Bước	n	m	g(m)=g(n) + cost(n,m)	f(m)=g(m) + h(m)	Γ(n)↑	OPEN	min
4	K f(K)<min	B	g(B) = g(K) + cost(K,B) = 14 + 6 = 20	f(B) = g(B) + h(B) = 20 + 0 = 20	B^{20}	B^{20}, I^{19}, A^{26}, H^{19}, F^{33}	+∞

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



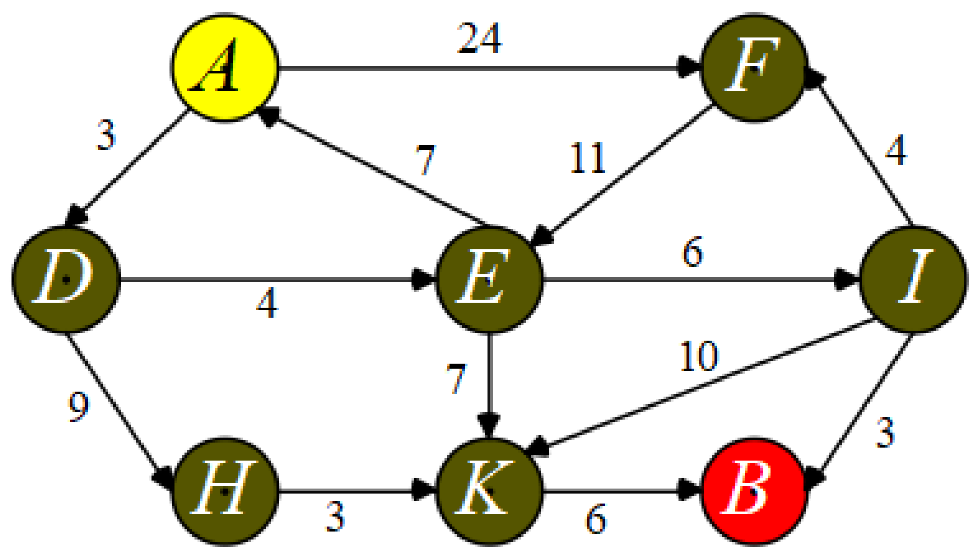
VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

▪ Nhánh – cận (Branch-and-Bound)

Bước	n	m	g(m)=g(n) + cost(n,m)	f(m)=g(m) + h(m)	Γ(n)↑	OPEN	min
5	B	B		$f(B) = 20 < min$		I^{19}, A^{19}, H^{19}, F^{33}	20
6	I f(I)<min	B F K	g(B) = g(I) + cost(I,B) = 13 + 3 = 16 g(F) = g(I) + cost(I,F) = 13 + 4 = 17 g(K) = g(I) + cost(I,K) = 13 + 10 = 23	f(B) = g(B) + h(B) = 16 + 0 = 16 f(F) = g(F) + h(F) = 17 + 19 = 26 f(K) = g(K) + h(K) = 23 + 4 = 27	$B^{16}\rightarrow$ $F^{26}\rightarrow$ K^{27}	B^{16}, F^{26}, K^{27}, A^{19}, H^{19}, F^{33}	20

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



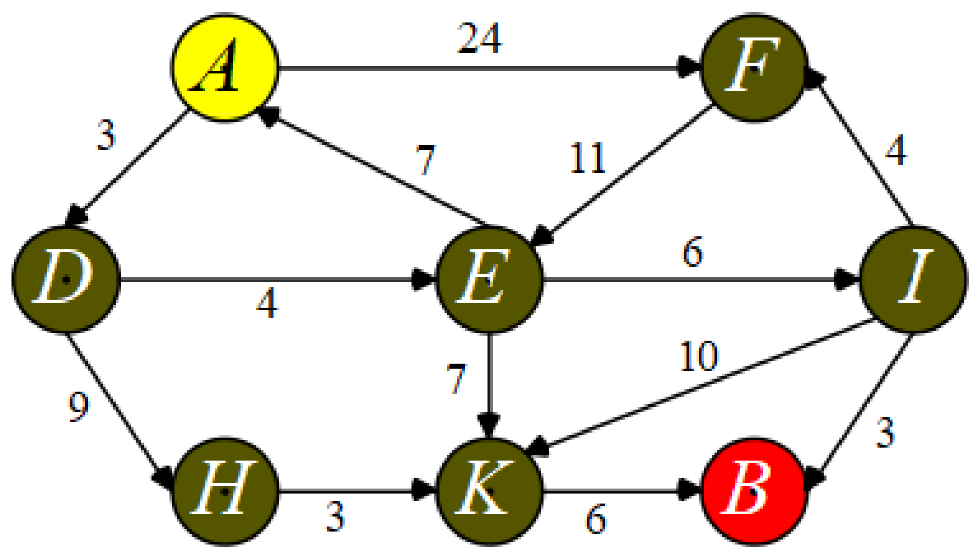
VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

▪ Nhánh – cận (Branch-and-Bound)

Bước	N	m	g(m)=g(n) + cost(n,m)	f(m)=g(m) + h(m)	Γ(n)↑	OPEN	min
7	B	B		$f(B) = 16 < min$		$F^{26}, K^{27}, A^{19}, H^{19}, F^{33}$	16
8	F		quay lại 2.1 (cắt bỏ nhánh con)	$f(F) = 26 > min$		$F^{26}, K^{27}, A^{19}, H^{19}, F^{33}$	16

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

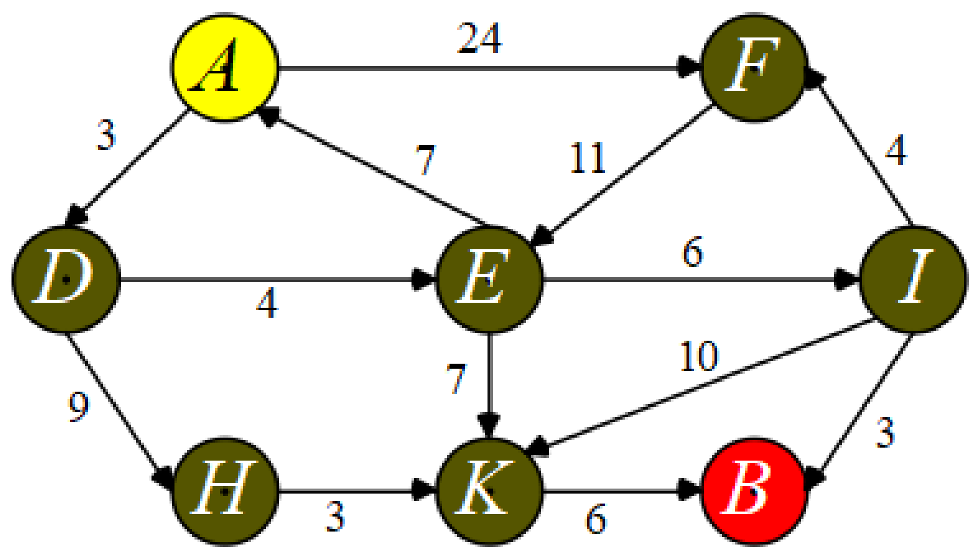
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Nhánh – cận (Branch-and-Bound)

Bước	N	m	g(m)=g(n) + cost(n,m)	f(m)=g(m) + h(m)	Γ(n)↑	OPEN	min
9	K		quay lại 2.1 (cắt bỏ nhánh con)	$f(K) = 27 > min$		K^{27}, A^{19}, H^{19}, F^{33}	16
10	A		quay lại 2.1 (cắt bỏ nhánh con)	$f(A) = 19 > min$		A^{19}, H^{19}, F^{33}	16

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



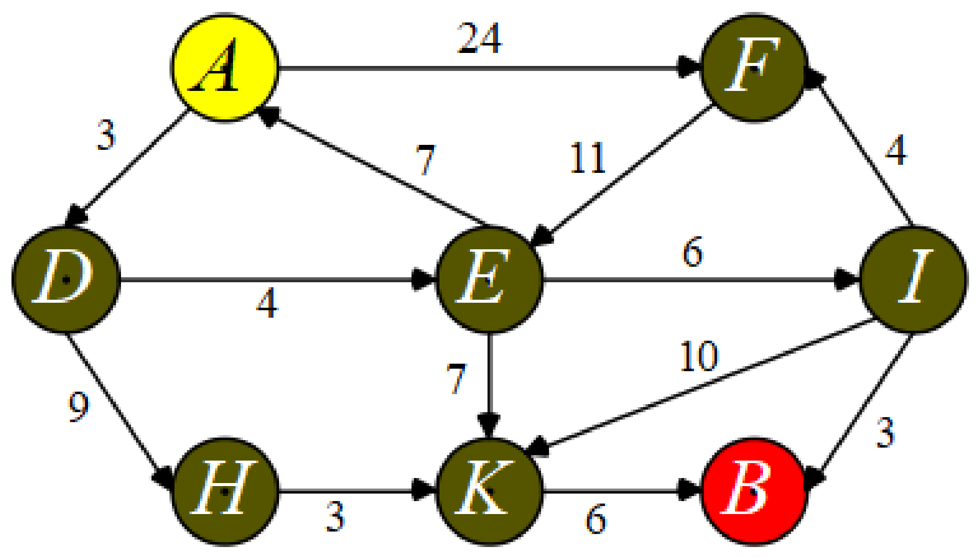
VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

▪ Nhánh – cận (Branch-and-Bound)

Bước	N	m	g(m)=g(n) + cost(n,m)	f(m)=g(m) + h(m)	Γ(n)↑	OPEN	min
11	H		quay lại 2.1 (cắt bỏ nhánh con)	$f(H) = 19 > min$		F^{33}	16
12	F		quay lại 2.1 (cắt bỏ nhánh con)	$f(F) = 33 > min$		$\varnothing$	16
13	TRUE						

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E

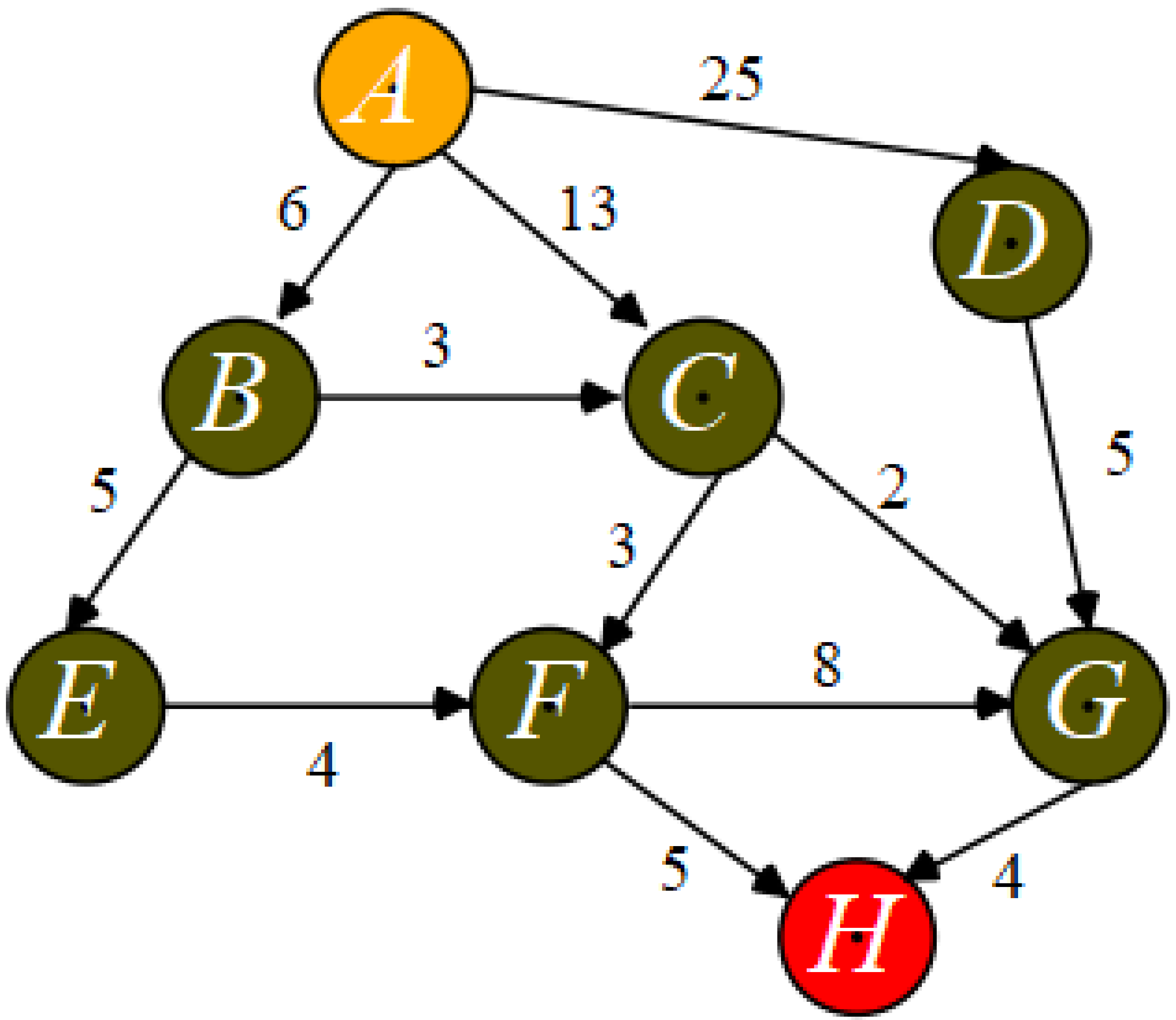


VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

- Nhánh – cận (Branch-and-Bound)

  - Áp dụng thuật toán nhánh cận để tìm đường đi ngắn nhất từ đỉnh A → I, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở Bảng.



A	B	C	D	E	F	G	H
20	5	7	6	9	1	3	0

HẾT CHƯƠNG 4