



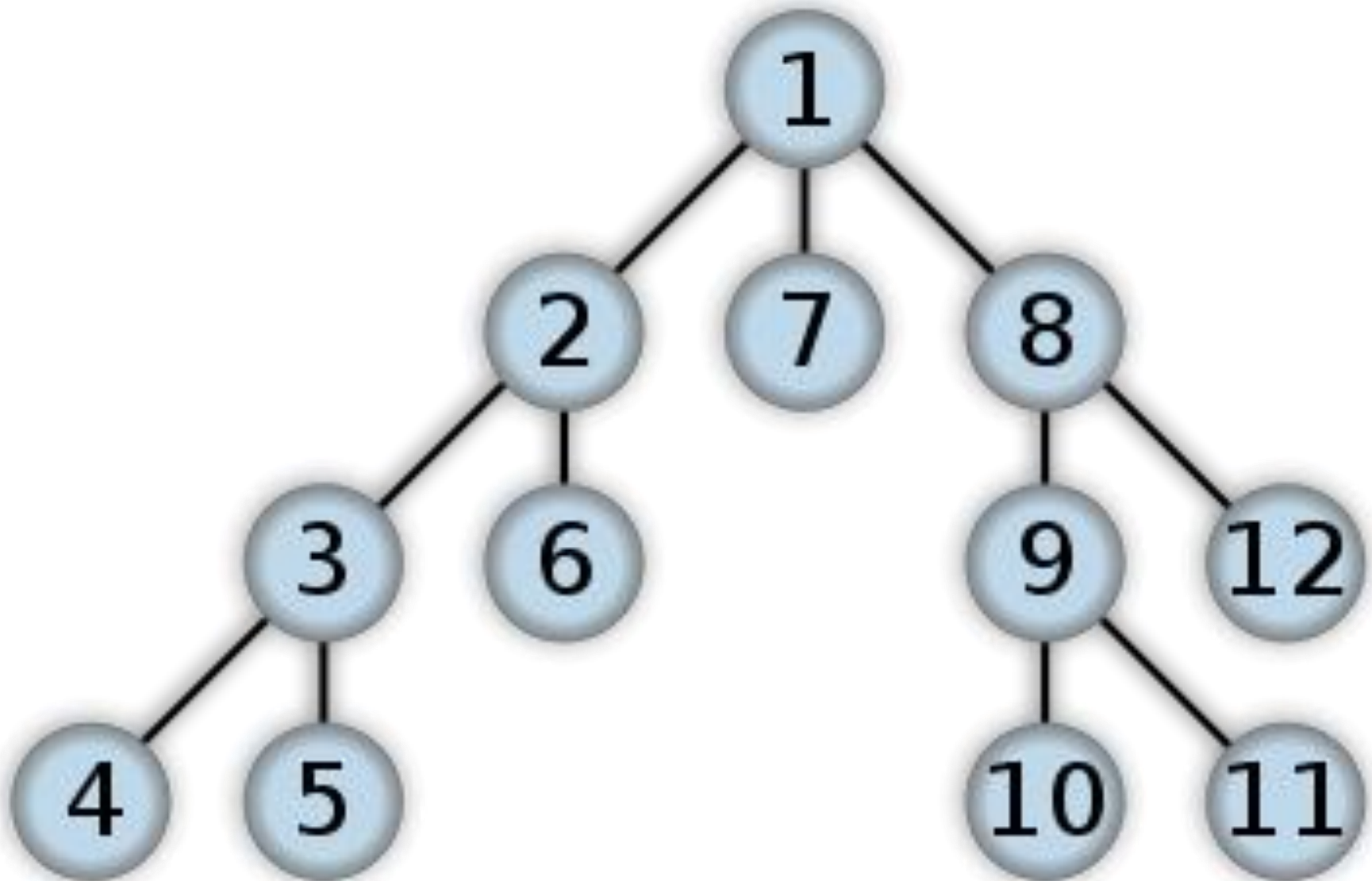
Universitatea Politehnica din București
Facultatea de Automatică și Calculatoare
Departamentul de Calculatoare



PARCURGEREA ARBORILOR

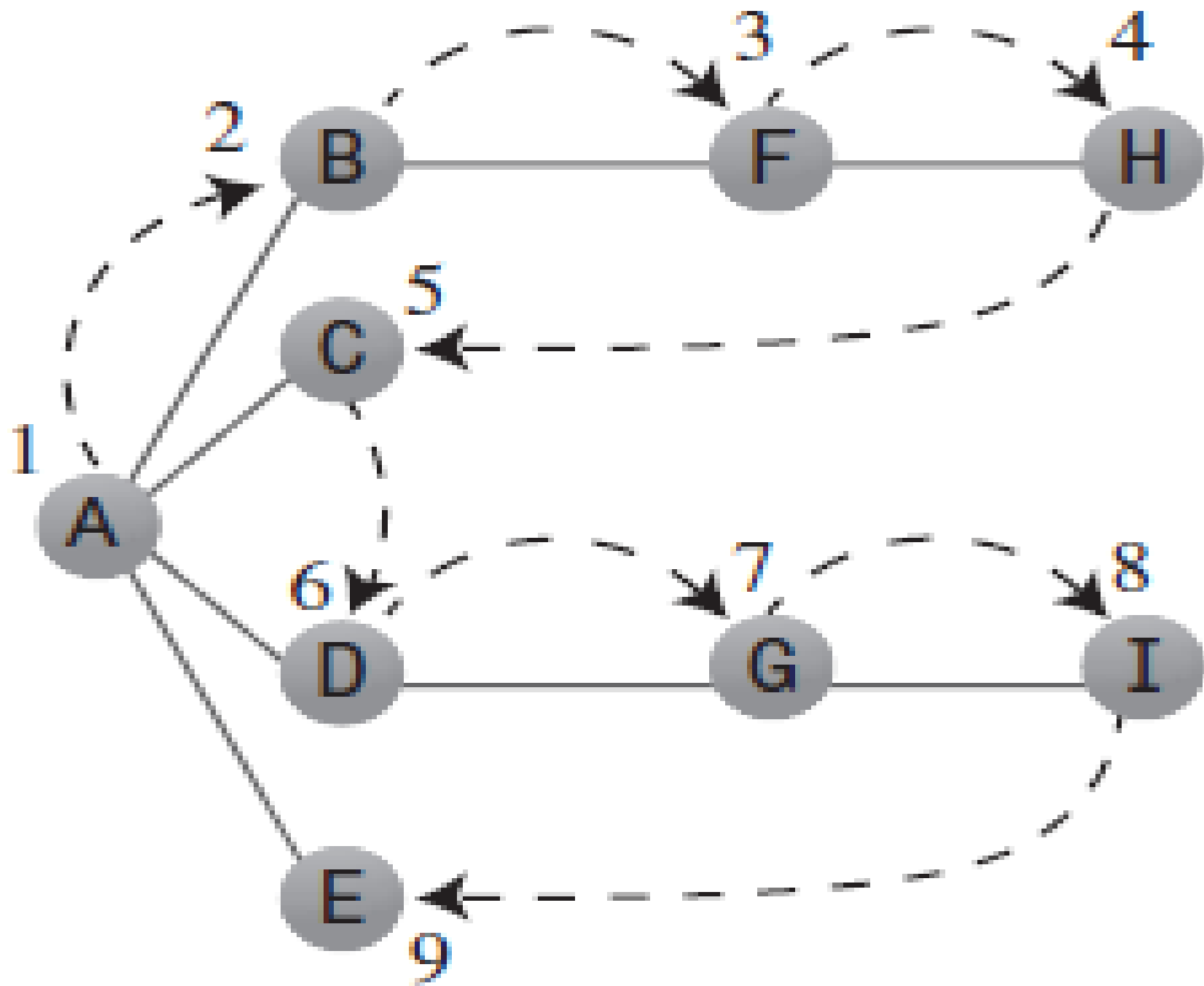
ÎN ADÂNCIME
ÎN LĂȚIME (PE NIVEL)

Parcurgerea în adâncime



Parcurgerea în adâncime

- Algoritmul de parcurgere în adâncime utilizează o **stivă** pentru a memora locul în care trebuie să se întoarcă, atunci când nu poate merge mai departe



Parcurgerea în adâncime

- Se pornește de la rădăcină - nodul **A**
- Se efectuează trei operații:
 - Se vizitează nodul
 - Se introduce nodul într-o **stivă**, pentru a-l memora
 - Se marchează nodul pentru a nu-l mai vizita încă o dată

Parcurgerea în adâncime

- Ne deplasăm la orice nod adiacent cu **A**, care nu a fost încă vizitat
- Presupunând că nodurile sunt selectate în ordine alfabetică, ajungem la **B**
- Vizităm nodul **B**, îl marcăm și îl introducem în **stivă**

Parcurgerea în adâncime

- Suntem în **B** și efectuăm aceeași operație ca mai înainte: mergem într-un nod adiacent care nu a mai fost vizitat
- Ajungem astfel în **F**
- Acest mod de deplasare de la un nod la următorul îl vom numi **Regula 1**
- **Regula 1** : dacă este posibil, vizităm un nod adiacent încă nevizitat, îl marcăm și îl introducem în **stivă**

Parcurgerea în adâncime

- Prin aplicarea repetată a regulii **1**, ajungem în **H**
- În acest punct, trebuie să ne modificăm strategia, întrucât nu mai există noduri nevizitate, adiacente cu **H**
- Aici va interveni Regula **2**
- **Regula 2** : dacă nu putem aplica regula **1**, extragem un nod din stivă

Parcurgerea în adâncime

- Respectând această regulă, extragem nodul **H** din **stivă**, ajungând înapoi în **F**
- **F** nu are însă noduri adiacente nevizitate, deci va fi extras la rândul său
- La fel și **B**
- În acest moment, doar **A** a mai rămas în stivă

Parcurgerea în adâncime

- Nodul **A** are noduri adiacente nevizitate, deci vizităm următorul nod **C**
- Vizităm apoi **D**, **G** și **I**, după care le extragem pe toate din **stivă**, atunci când ne blocăm în **I**
- Am ajuns din nou înapoi în **A**
- Vizităm nodul **E** și ne întoarcem încă o dată în **A**

Parcurgerea în adâncime

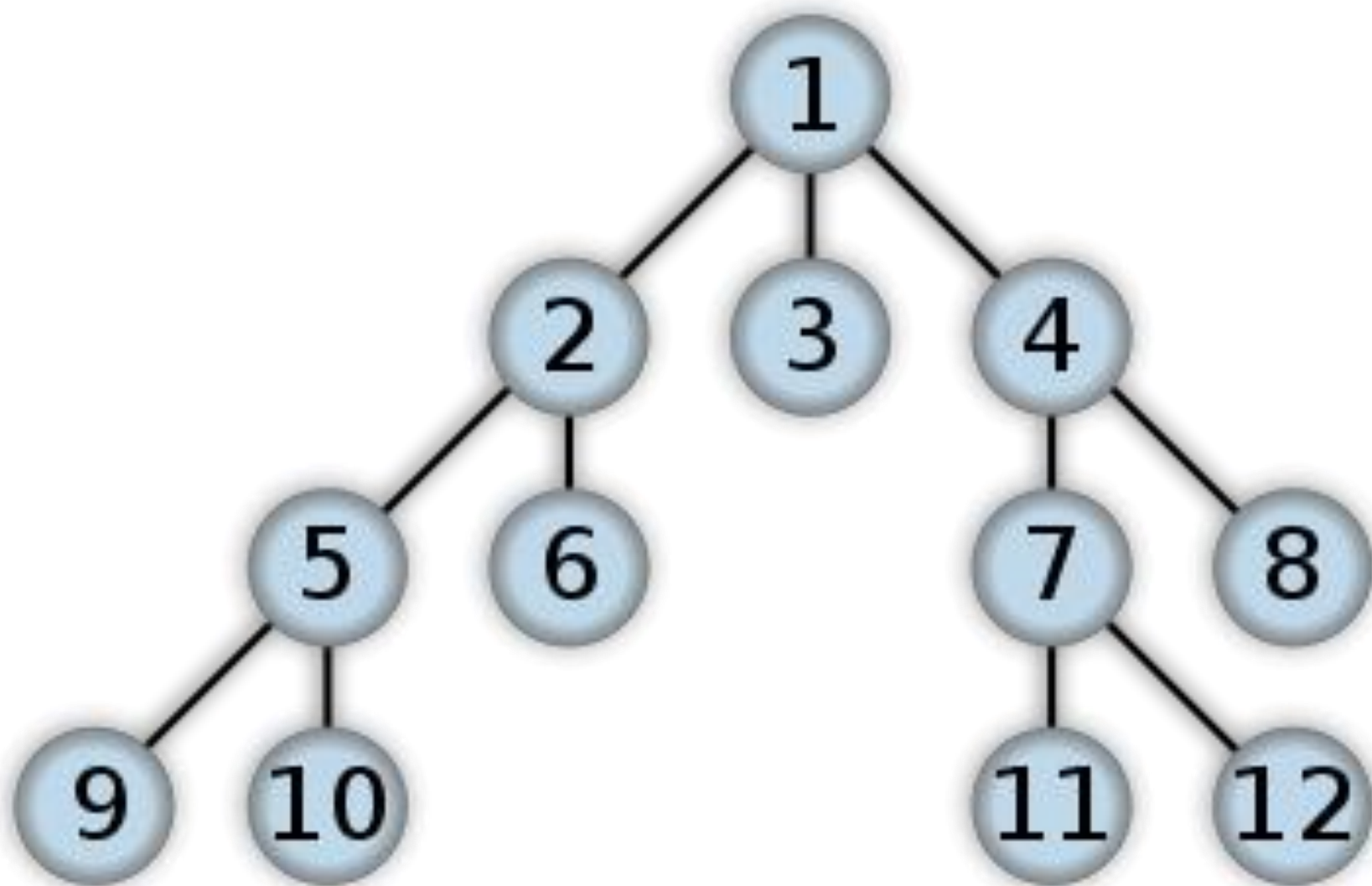
- De data aceasta, **A** nu mai are vecini nevizitați, deci îl extragem din **stivă**
- Nu a mai rămas nimic de extras din **stivă**
- Aici intră în acțiune **Regula 3**
- **Regula 3** : atunci când nu mai putem aplica regulile **1** sau **2**, parcurgerea s-a terminat

Event	Stack
Visit A	A
Visit B	AB
Visit F	ABF
Visit H	ABFH
Pop H	ABF
Pop F	AB
Pop B	A
Visit C	AC
Pop C	A
Visit D	AD
Visit G	ADG
Visit I	ADGI
Pop I	ADG
Pop G	AD
Pop D	A
Visit E	AE
Pop E	A
Pop A	
Done	

Observații

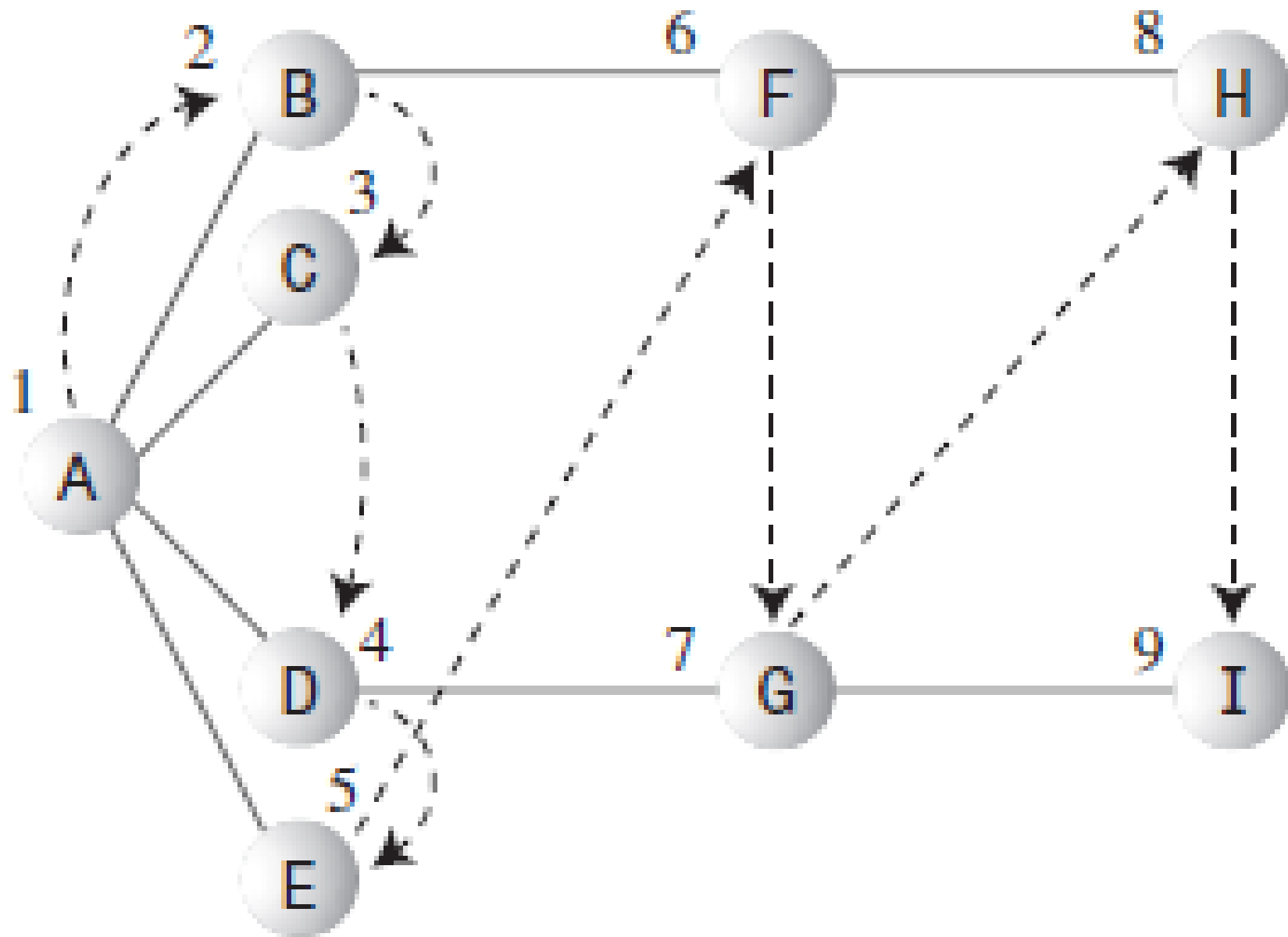
- Conținutul **stivei** reprezintă calea de la rădăcină până în nodul curent
- Pe măsură ce ne depărtăm de rădăcină, vom introduce noduri noi în **stivă**
- Atunci când ne deplasăm înapoi spre rădăcină, extragem nodurile din **stivă**
- Ordinea vizitării nodurilor este **ABFHCDGIE**

Parcurgerea în lățime (pe nivel)



Parcurgerea în lățime (pe nivel)

- La parcurgerea în lățime, algoritmul va rămâne cât mai aproape de punctul de pornire
- Algoritmul vizitează toate nodurile adiacente cu rădăcina și numai după aceea trece cu un pas mai departe
- Acest algoritm este implementat cu ajutorul unei **cozi**



Parcurgerea în lățime

- Se pornește de la rădăcina **A** și se aplică următoarele reguli:
- **Regula 1** : vizităm următorul nod nevizitat (dacă există un astfel de nod), adiacent cu nodul curent, îl marcăm și îl introducem în **coadă**

Parcurgerea în lățime

- **Regula 2** : dacă nu putem aplica regula **1**, din cauză că nu mai există noduri nevizitate, ștergem un nod din **coadă** (dacă este posibil), nodul șters devenind nodul curent
- **Regula 3** : dacă nu putem aplica regula **2**, din cauză că în **coadă** nu mai există noduri, am terminat

Observații

- Vom vizita toate nodurile adiacente cu **A**, inserându-le pe toate în **coadă**, pe măsură ce le vizităm
- Astfel, vom vizita **A**, **B**, **C**, **D** și **E**
- În acest moment, **coada** conține nodurile **BCDE**

Observații

- Deoarece nu mai există noduri nevizitate adiacente cu **A**, ștergem **B** din **coadă** și căutăm noduri adiacente cu el
- Găsim nodul **F**, pe care îl inserăm în **coadă**
- Nici **B** nu mai are vecini nevizitați, deci vom șterge nodul **C** din **coadă**

Observații

- Nodul **C** nu are vecini nevizitați, deci vom șterge **D**, vizitând nodul **G**
- Nici **D** nu mai are noduri adiacente nevizitate, deci vom șterge și **E** din **coadă**
- **Coadă** conține acum nodurile **FG**
- Ștergem **F** și vizităm **H**, după care ștergem **G** și vizităm **I**

Observații

- **Coadă** conține acum nodurile **HI**
- După ștergerea acestor noduri, care nu au vecini nevizitați, **coada** devine vidă, deci parcurgerea se încheie

Event	Queue (Front to Rear)
Visit A	
Visit B	B
Visit C	BC
Visit D	BCD
Visit E	BCDE
Remove B	CDE
Visit F	CDEF
Remove C	DEF
Remove D	EF
Visit G	EFG
Remove E	FG
Remove F	G
Visit H	GH
Remove G	H
Visit I	HI
Remove H	I
Remove I	
Done	

Observații

- În fiecare moment, **coada** conține nodurile care au fost vizitate, dar ai căror vecini nu au fost complet explorați
- Nodurile sunt vizitate în ordinea **ABCDEFGHI**