Ricerca esaustiva e Backtracking

A.A. 2017-2018

Ricerca esaustiva

Esistono problemi che, per essere risolti, richiedono una ricerca condotta attraverso un gran numero di soluzioni potenziali e per la cui soluzione non sembrano esistere algoritmi efficienti.

Algoritmo efficiente? → lineare, n log n , $n^{3/2}$

Vi è la tendenza a considerare pessimi gli algoritmi quadratici, per non parlare di quelli con tempi di esecuzione cubici.

Per alcuni problemi però un qualsiasi informatico sarebbe contento di conoscere, almeno da un punto di vista teorico, un algoritmo con prestazioni proporzionali a n^{20}.

Questi problemi "*sembrano*" richiedere tempo esponenziale.

Ricerca esaustiva

Esempio: ricerca su grafi.

Conosciamo metodi per visitare in modo sistematico tutti i vertici di un grafo in modo che ciascun nodo venga visitato solo una volta.

Ad esempio: una ricerca depth-first in un grafo connesso.
Quando si trova un nodo già visitato si *"torna indietro"*.

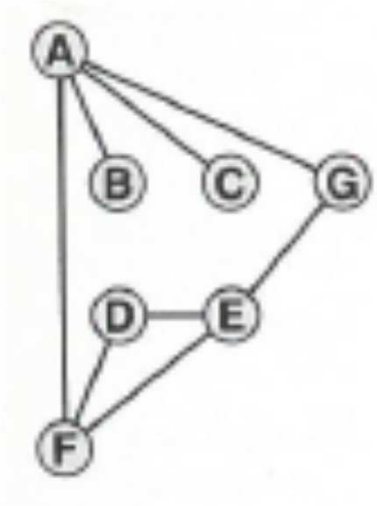
Si assumono come *globali* una *rappresentazione* del grafo, un vettore X[ ] di dimensione |V| inizializzato a 0. h è globale *con valore iniziale 0*.

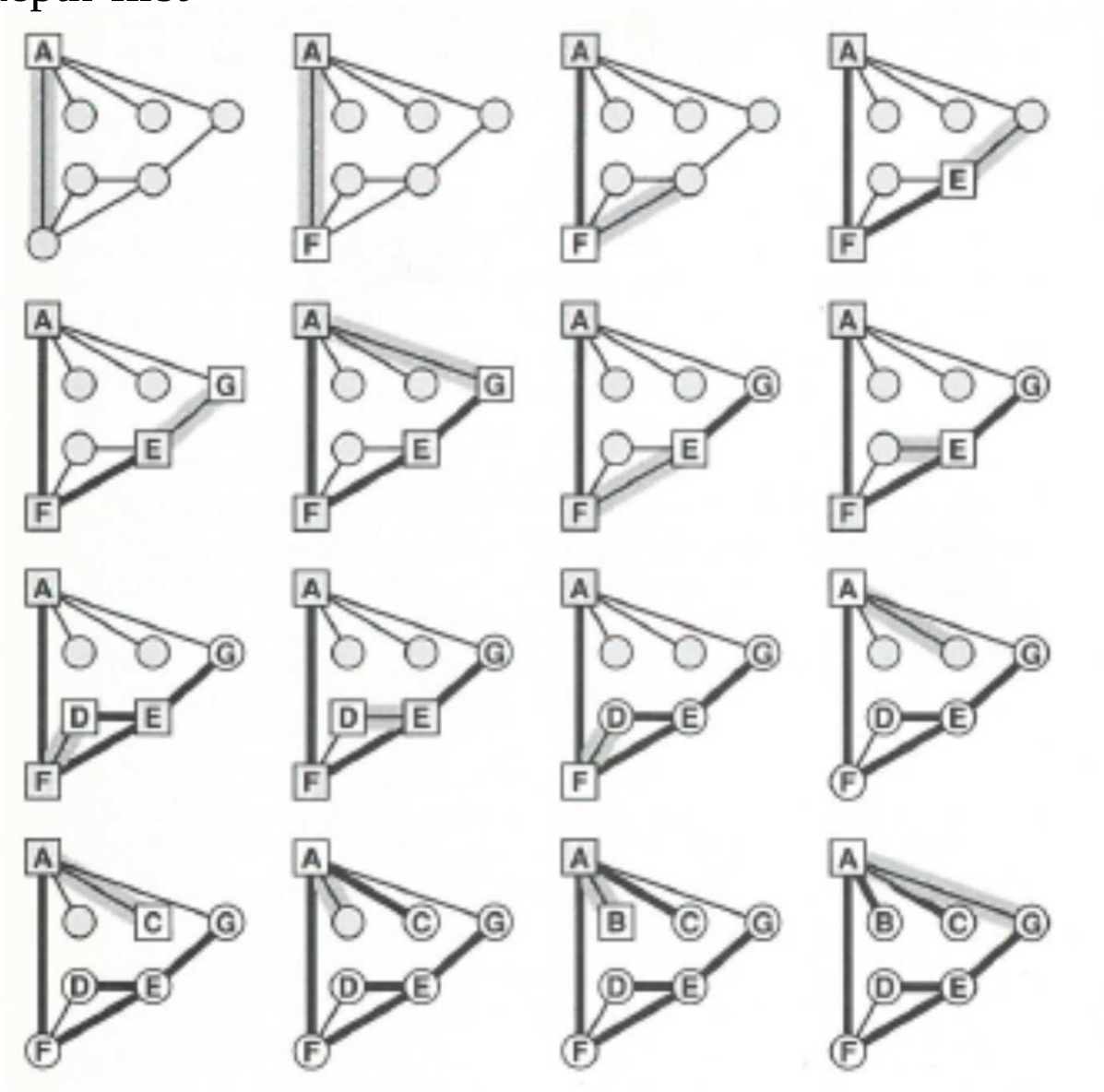
```
visit (G, i)
    h ← h + 1
    X[i] ← h
    for j ← 1 to |V|
        if ( (i,j) ∈ E)
            if (X[j] = 0)  visit (G, j)
    end
```

h rappresenta l'ordine in cui i nodi vengono visitati

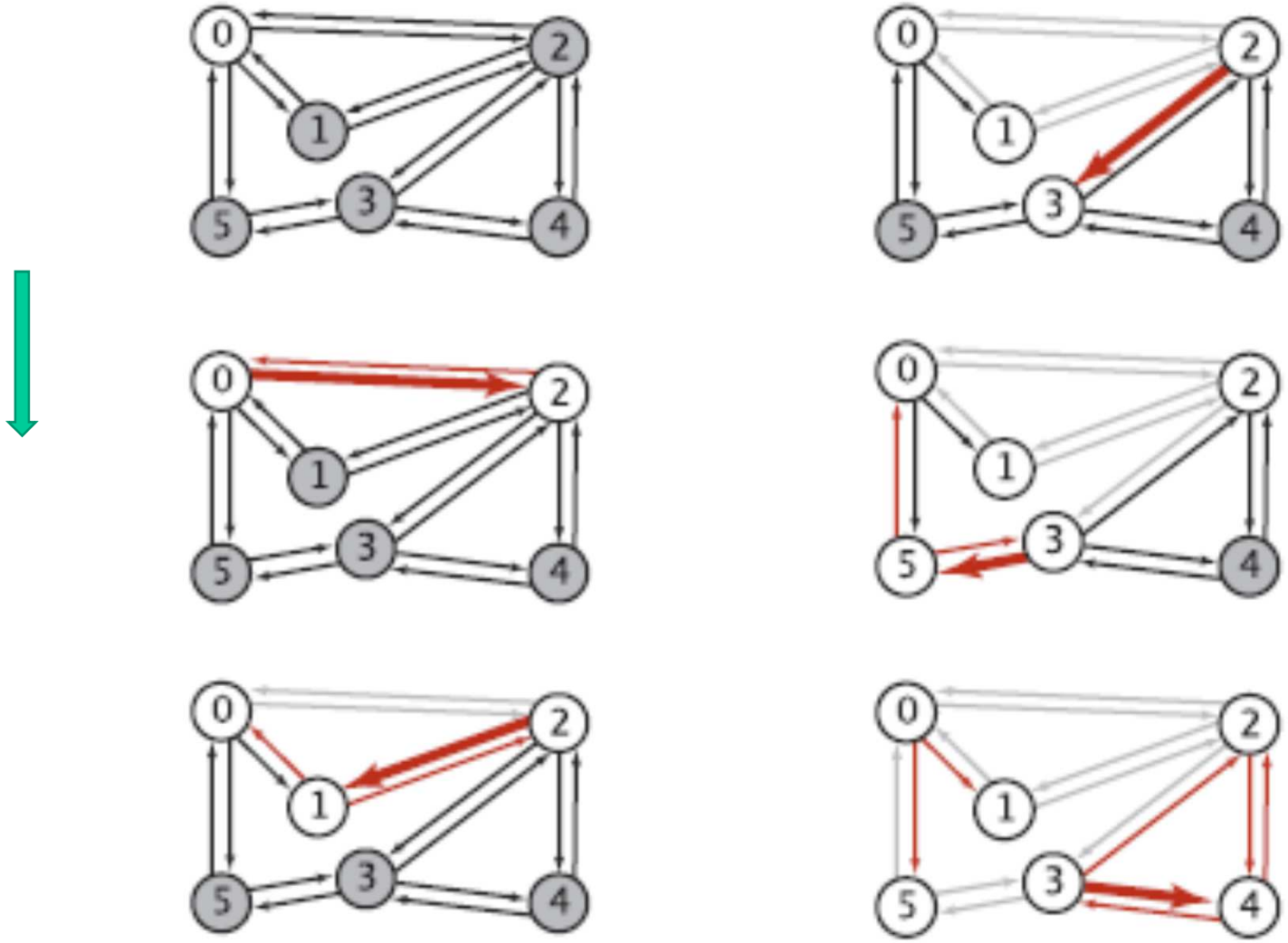
Chiamata iniziale: *visit*(i)
(i è un nodo qualunque)

Esempio di esecuzione per depth-first





Esempio di esecuzione per depth-first



Ricerca esaustiva

La tecnica algoritmica che consiste nel risolvere un problema cercando (più o meno alla ceca) tutte le possibili soluzioni esplorando sistematicamente lo spazio degli stati è detta "***backtracking***".

Uno dei problemi più noti che corrisponde a questa descrizione è quello del **ciclo di Hamilton**: dato un grafo non orientato, esiste un ciclo semplice che contiene tutti i nodi?

In altre parole, partendo da un nodo qualunque, è possibile visitare tutti gli altri nodi per poi tornare a quello di partenza considerando ciascun nodo esattamente una volta?

Per quanto riguarda il ciclo Hamiltoniano una soluzione di questo tipo non è evidente perché sembra necessario visitare ogni nodo più di una volta.

Consideriamo il grafo in figura:

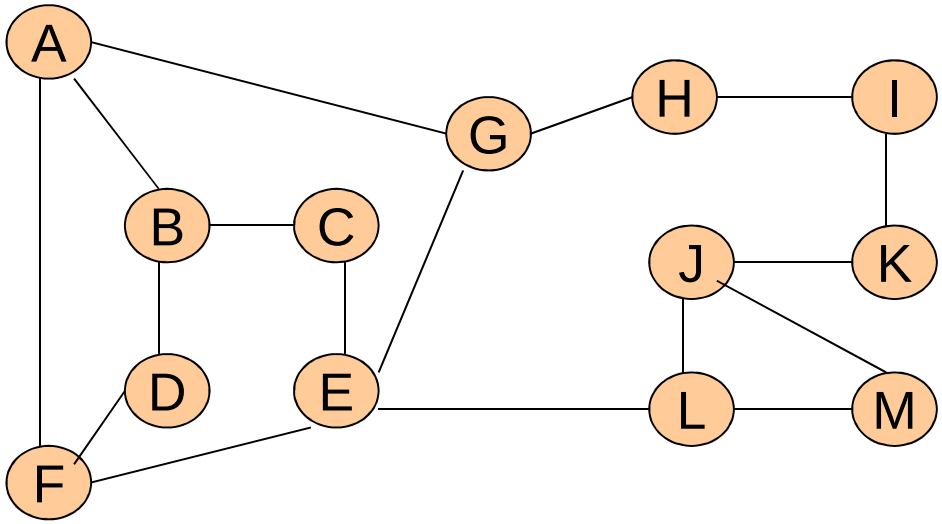
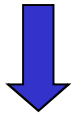


Fig. 1

Nell'ipotesi che il grafo sia memorizzato con una matrice o con liste di adiacenza in ordine alfabetico, una ricerca depth-first visita i nodi nell'ordine ABCEFDGHIKJLM: non è un ciclo semplice

Ricerca esaustiva

Per trovare un ciclo di Hamilton, è necessario cercare un altro modo di visitare i nodi.



Tentiamo sistematicamente tutti le possibilità.
La tecnica di verificare ognuno dei possibili percorsi prende il nome di *ricerca esaustiva*.

Semplice *modifica* alla procedura *visit*:

"***pulire dove passa***"

```
visit (G, i)
    h ← h + 1
    X[i] ← h
    if h = |V| then
        if (i, X[1]) ∈ E then return
        else for j ← 1 to |V|
                if ( (i,j) ∈ E) then
                    if (X[j] = 0) visit (G, j)
        h ← h - 1
        X[i] ← 0
        return
```

Chiamata iniziale *visit*(G,1)

I nodi con X diverso da 0 sono quelli per i quali *visit* non ha ancora terminato l'analisi: corrispondono a un cammino semplice di lunghezza *h* con partenza dal nodo iniziale e arrivo nel nodo in esame. La procedura controlla tutti i percorsi semplici del grafo che partono dal nodo iniziale.

Nella ricerca esaustiva ciascun nodo può essere visitato più volte.

La Fig. 2 mostra l'ordine nel quale vengono verificati i percorsi applicando l'algoritmo al grafo di Fig. 1.
Il primo percorso controllato è ABCEFD. Il secondo ABCEGHIKJLM.

Per fare in modo che *visit* controlli l'esistenza di un ciclo Hamiltoniano, tutte le volte che X[i] = |V| è necessario controllare l'esistenza di un arco che colleghi i al vertice iniziale.
Nell'esempio esiste solo un ciclo di questo tipo, che compare 2 volte nell'albero di Fig. 2, attraversato nei due sensi (cammini formati dai nodi bianchi).

Ricerca esaustiva

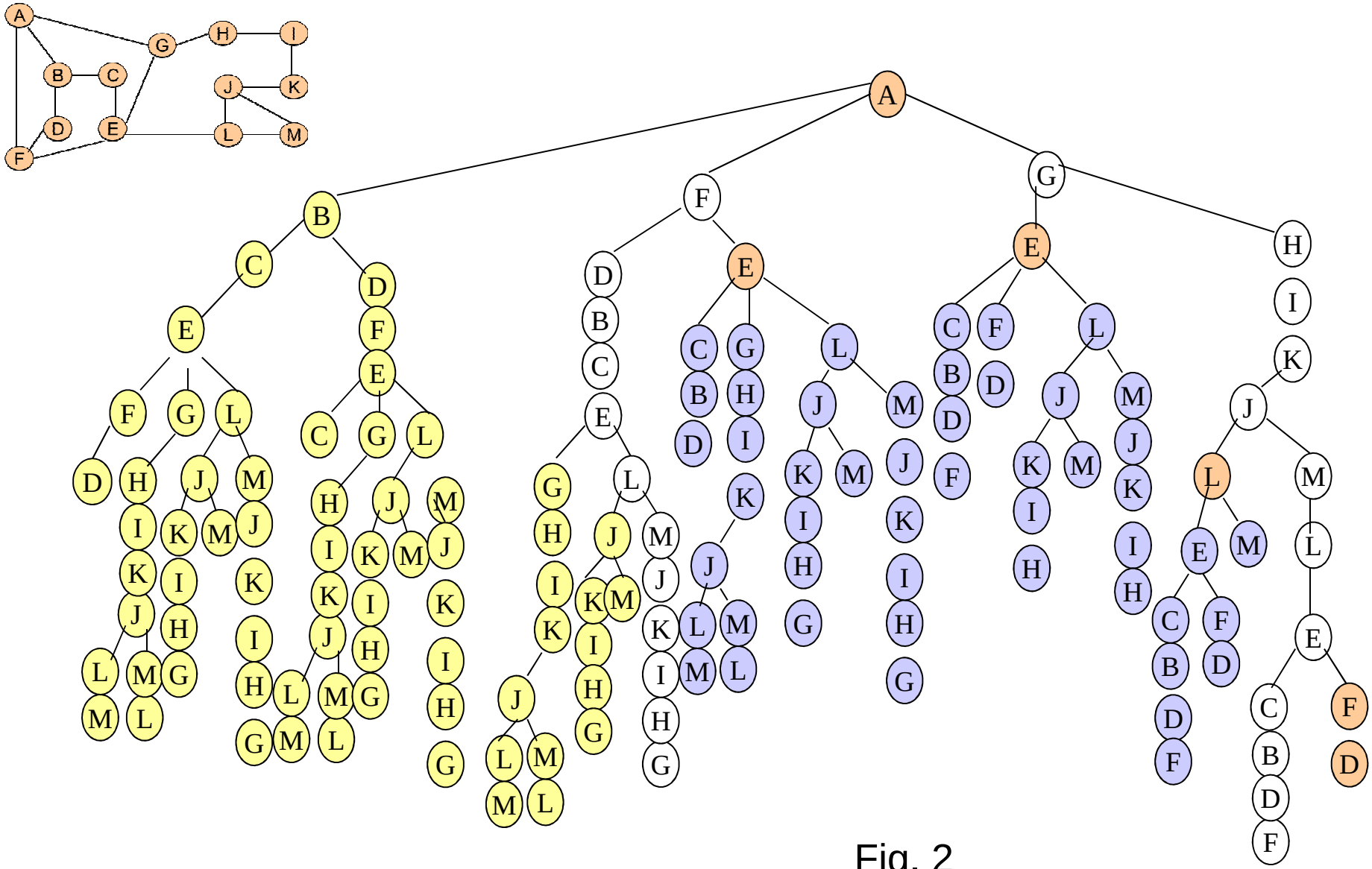
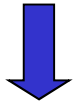


Fig. 2

Ricerca esaustiva

Il tempo impiegato dalla procedura per la ricerca esaustiva è proporzionale al numero di chiamate a *visit*, cioè al numero dei nodi dell'albero. Limitato da |V|!

È necessario trovare tecniche che consentano di ridurre il numero di possibilità da considerare



Aggiungere dei controlli per evitare di effettuare chiamate ricorsive in corrispondenza di alcuni nodi del grafo.

potare l'albero

Potare l'albero

1. Rimozione delle simmetrie
Nell'esempio precedente ogni ciclo è attraversato in entrambe le direzioni, cioè viene percorso due volte.
È possibile assicurarsi di trovare ciascun ciclo una volta sola imponendo che tre nodi debbano comparire in un determinato ordine.
Ad esempio se si impone che *C debba comparire dopo A e prima di B* si ottiene un albero decisamente più piccolo del precedente, mostrato in Fig. 3.

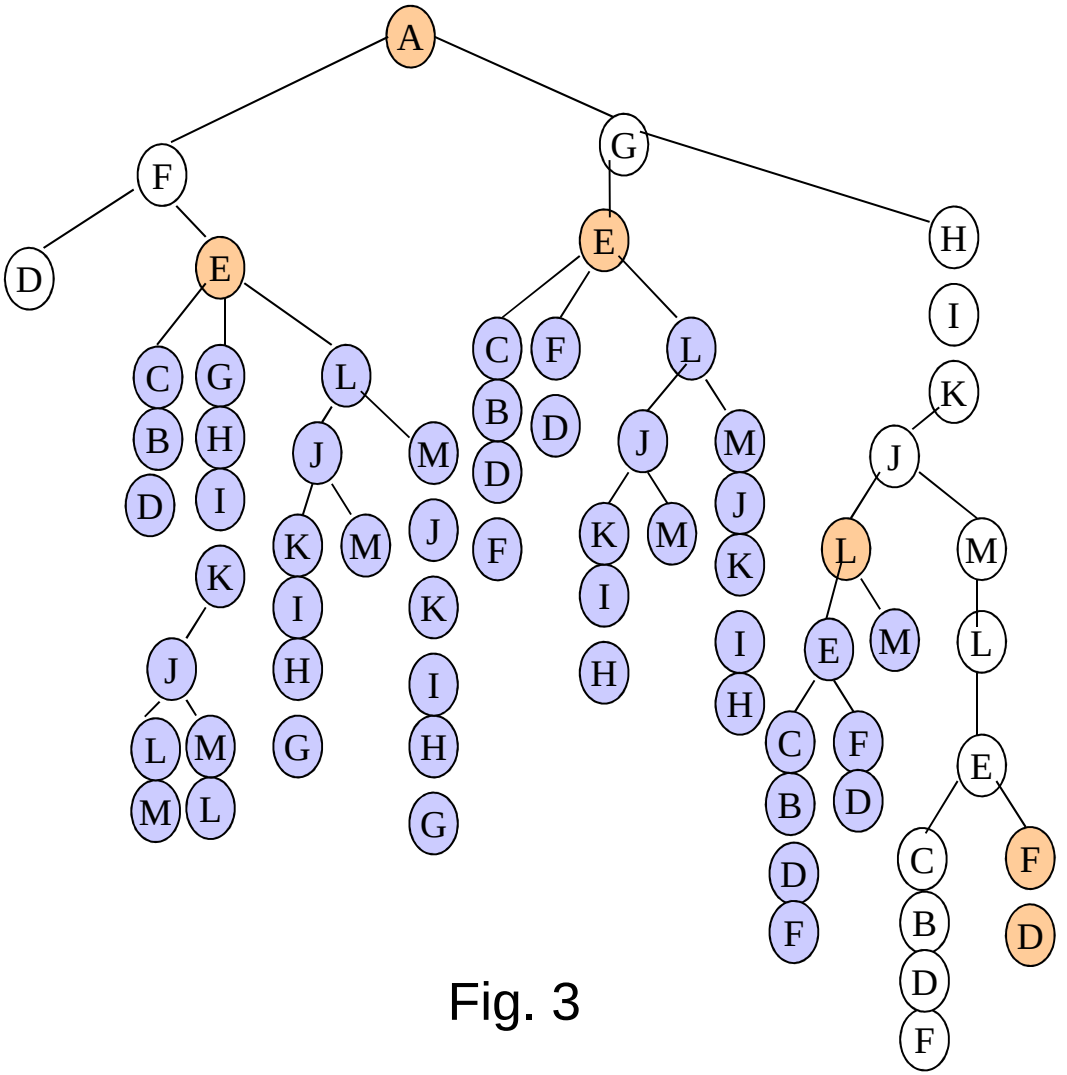


Fig. 3

Potare l’albero

2. Un altro modo per ridurre il numero di chiamate ricorsive deriva dall’osservazione che alcuni percorsi potrebbero dividere il grafo in modo tale che i nodi non visitati non siano connessi, in tal caso ovviamente è impossibile trovare un ciclo.

Ad esempio nel grafo considerato non può esistere alcun percorso semplice che inizia per AFE perché questo percorso separa B, C e D dal resto del grafo. La scoperta di questo fatto, al costo di una ricerca depth-first, consente di evitare ben 25 chiamate a *visit*.
Considerazioni analoghe si possono fare per i percorsi che iniziano per AGE e per quelli che presentano K, J e L in quest’ordine.

Potare l'albero

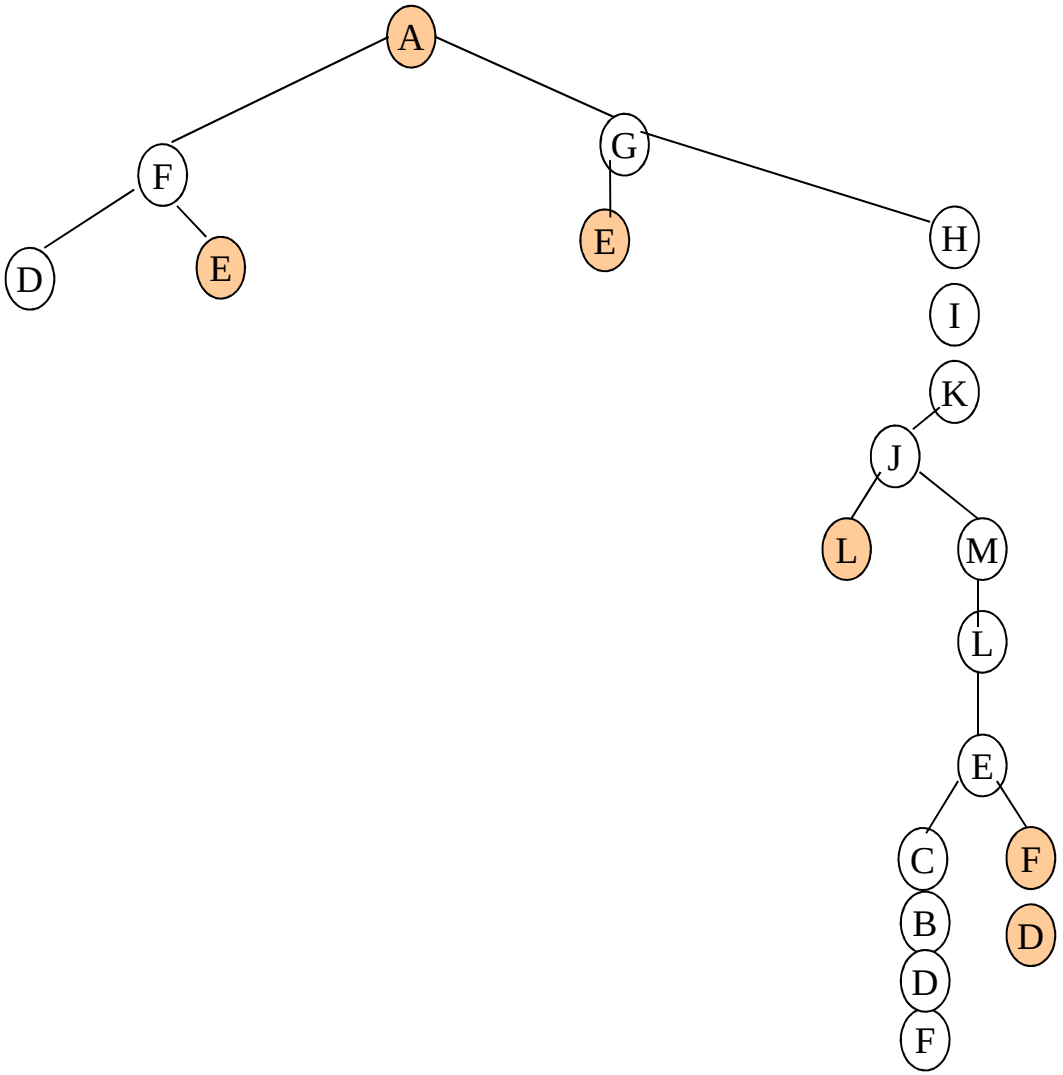


Fig. 4