UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Cerca en profunditat[1]

Albert Sanchis
Alfons Juan

DSIC

Departament de Sistemes Informàtics i Computació

[1] Per a una correcta visualització, es requereix l'Acrobat Reader v. 7.0 o superior

Objectius formatius

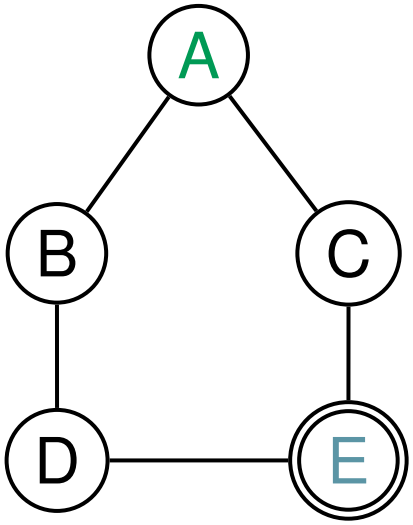
- Analitzar cerca en profunditat.
- Descriure cerca en profunditat en la variant de *backtracking*.
- Aplicar cerca en profunditat iterativa.

Índex

1 Introducció

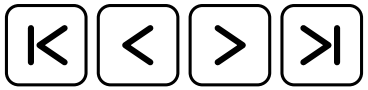
Cerca en profunditat (DFS, de Depth-first search) consisteix a enumerar camins fins a trobar una solució, prioritzant els més profunds (llargs) i limitant la llargària màxima (sols camins finits):

Graf d'estats



Cerca (limitada)

A
AB
AC
ABA
ABD
ABAB
ABAC
ABDB
ABDE sol.!



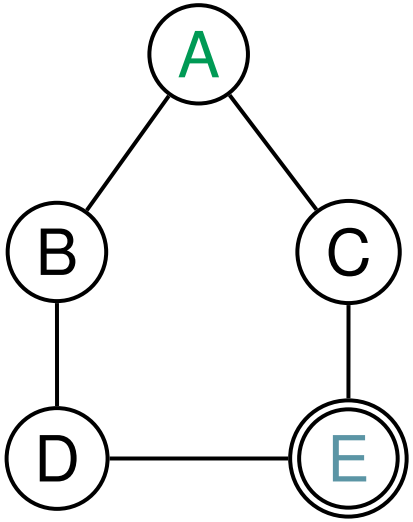
Nota: no s'emmagatzemen nodes tancats per eficiència espacial.

2 Cerca en profunditat [1, 2]

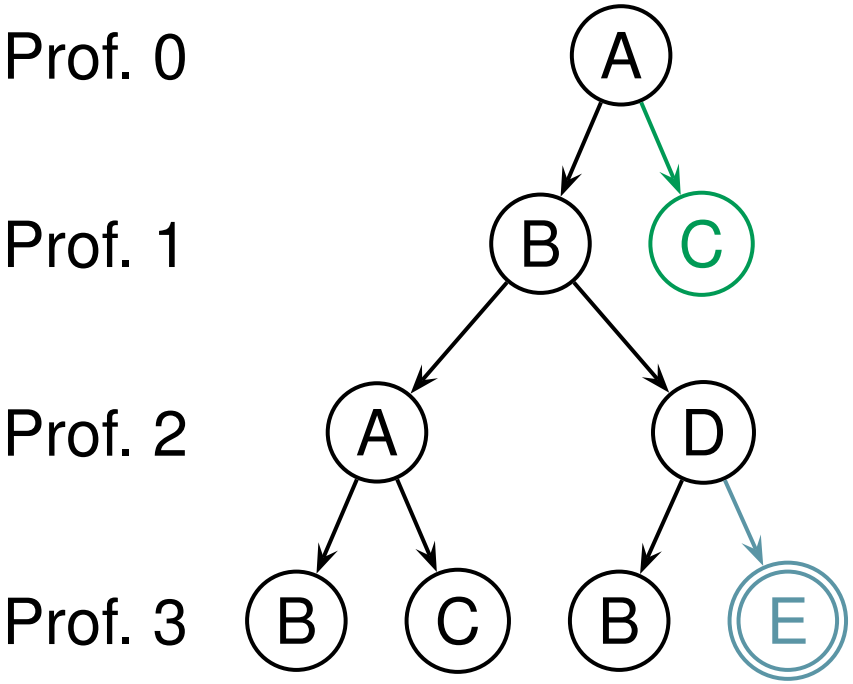
DFS(G, s', m**)** // ***Depth-first search*** amb profunditat màxima m
  $O = IniPila(s')$ // ***Open: frontera-pila*** de la cerca
  mentre no $PilaBuida(O)$:
    $s = Desapila(O)$ // selecció ***LIFO (Last in, first out)***
    si $Objectiu(s)$ **retorna** s // solució trobada!
    si $Profunditat(s) < m$: // no a profunditat màxima
      per a tota $(s, n) \in Adjacents(G, s)$: // generació: n fill d's
        $Apila(O, n)$ // afegim n a la pila
  retorna NULL // cap solució trobada

L'arbre de cerca en profunditat ($m = 3$)

Graf d'estats



Arbre de cerca



It. 8: ABABCDBEC E sol.!

|< < > >|

Qualitat: incompleta i subòptima.

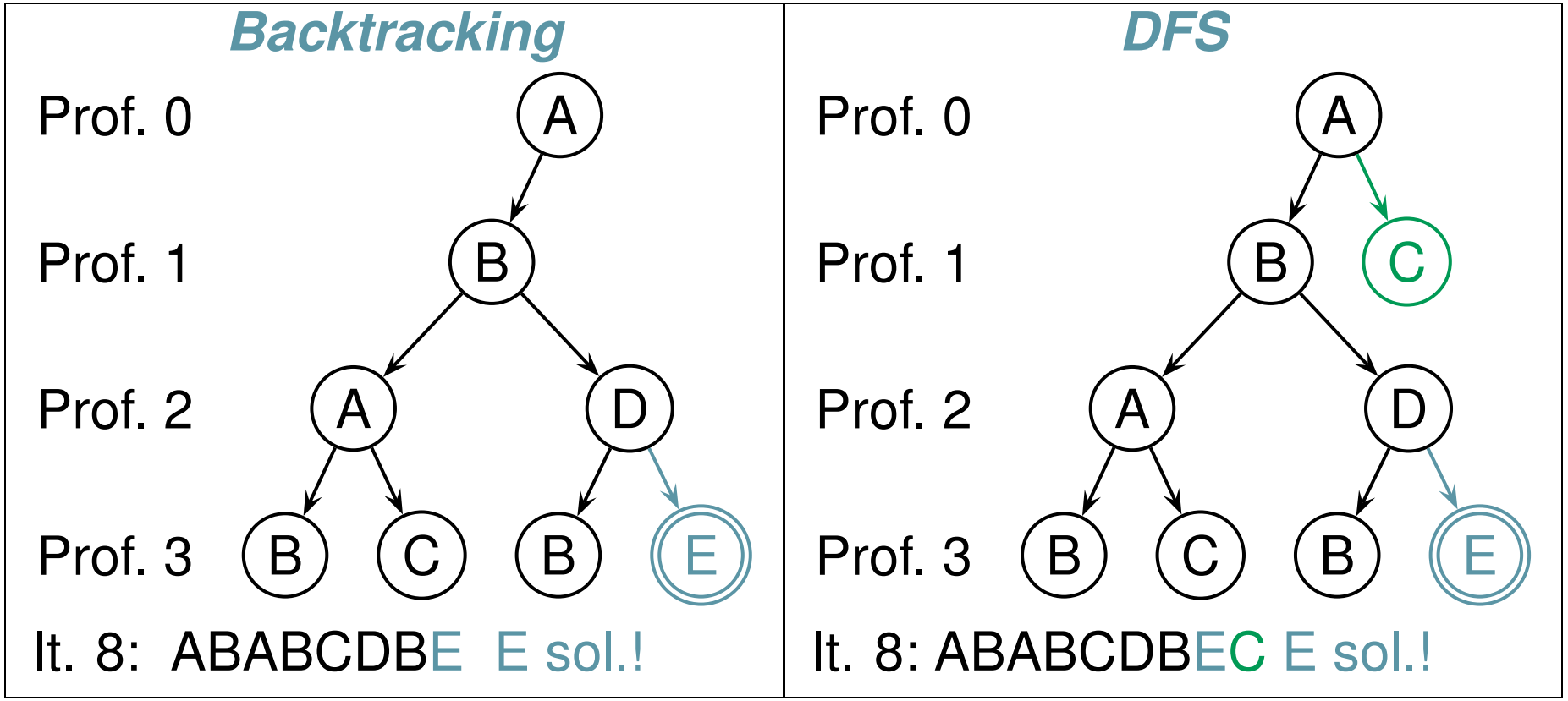
Complexitat: $O(b^m)$ temporal i $O(bm)$ espacial.

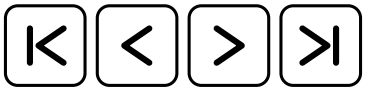
3 Backtracking

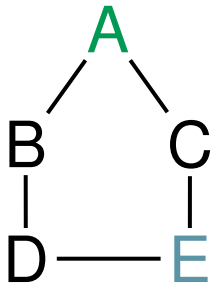
Variant de DFS (recursiva) amb generació individual de fills:

```
BT(G, s, m)                          // Backtracking amb profunditat màxima m
  si Objectiu(s) retorna s                          // solució trobada!
  si m = 0 retorna NULL                             // profunditat màxima
  n = PrimerAdjacent(G, s)                          // generació: n primer fill d's
  mentre n ≠ NULL:
    r = BT(G, n, m − 1)                             // resultat del fill actual
    si r ≠ NULL: retorna r                          // si r és solució, acabem
    n = SegüentAdjacent(G, s, n)                    // generació: n següent fill d's
  retorna NULL                                      // cap solució trobada
```

L'arbre de cerca amb backtracking ($m = 3$)







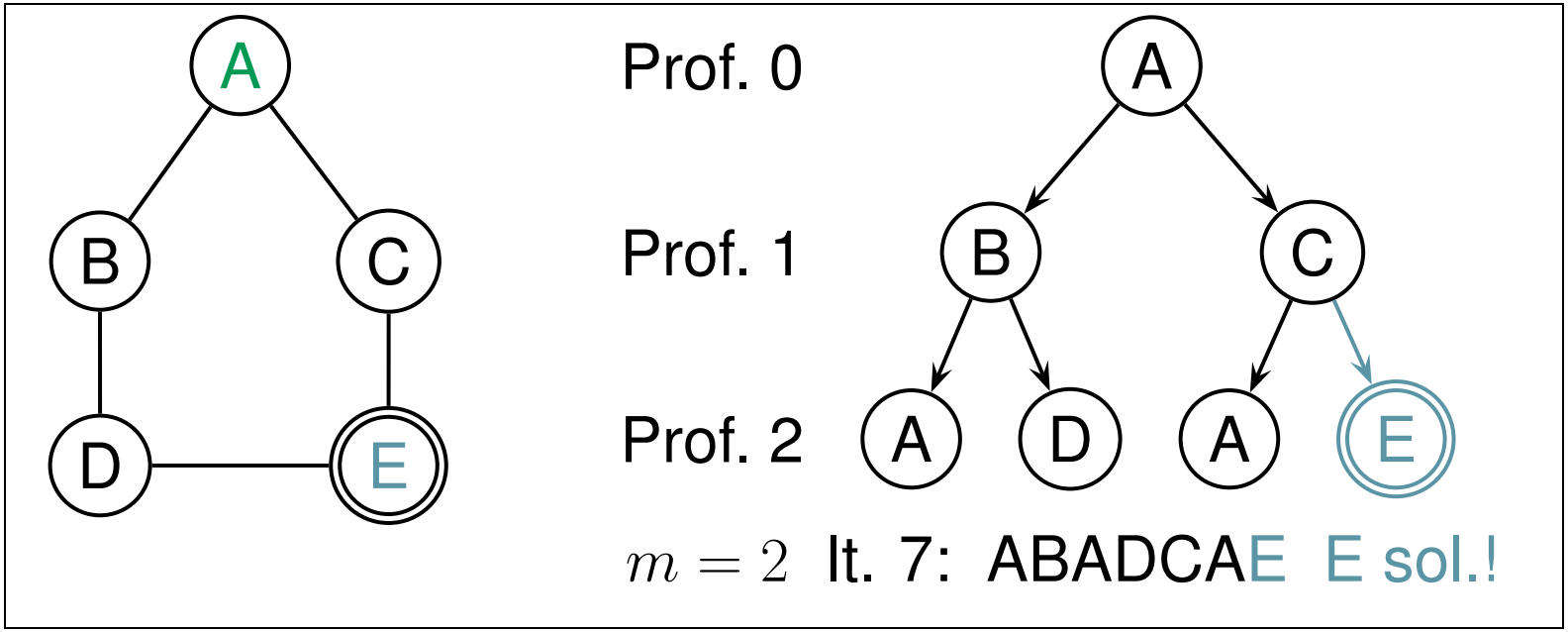
Qualitat: incompleta i subòptima.

Cost temporal: $O(b^m)$.

Cost espacial: $O(m)$, millor que el $O(bm)$ de DFS.

4 Cerca en profunditat iterativa [3]

PI(G, s) // ***Profunditat Iterativa***
  per a $m = 0, 1, 2, \ldots$: **si** $(r = \text{DFS}(G, s, m)) \neq \text{NULL}$: **retorna** r



Qualitat: completa i òptima amb accions de cost positiu idèntic.

Complexitat: $O(b^d)$ temporal i $O(bd)$ espacial.

5 Conclusions

Hem vist:

- Cerca en profunditat.
- La variant amb *backtracking* de cerca en profunditat recursiva.
- L'extensió anomenada cerca en profunditat iterativa.

Alguns aspectes a destacar sobre DFS:

- Incompleta i subòptima.
- Cost espacial raonable, sobretot amb *backtracking*.
- Pot ser una bona opció per a cercar solucions profundes, especialment si la llargària (cost) del camí no és molt rellevant.
- L'extensió de cerca en profunditat iterativa és completa, òptima amb arestes de cost idèntic, i manté un cost espacial raonable, per la qual cosa és generalment preferible a BFS.

Referències

[1] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2010.

[2] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2018.

[3] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 1985.

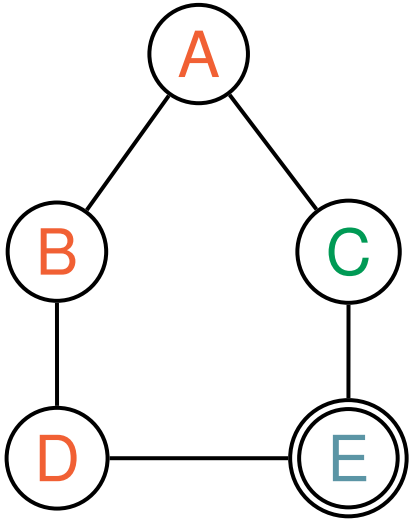
Cerca en profunditat (amb graf) [1]

Cerca en graf: es manté un conjunt de nodes explorats C.

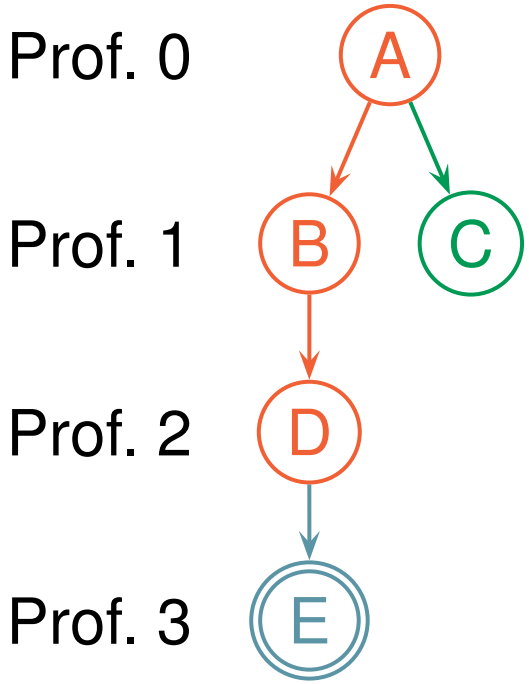
DFS(G, s'**)** // ***Depth-first search***; G graf i s node inicial
  $O = IniPila(s')$ // ***Open: frontera-pila*** de la cerca
  $C = \emptyset$ // ***Closed:*** conjunt de nodes explorats
  mentre no $PilaBuida(O)$:
    $s = Desapila(O)$ // selecció ***LIFO (Last in, first out)***
    si $Objectiu(s)$ **retorna** n // solució trobada!
    $C = C \cup \{s\}$ // s ja explorat
    per a tota $(s, n) \in Adjacents(G, s)$: // generació: n fill d's
      si $n \notin C \cup O$: // n no descobert fins ara
        $Apila(O, n)$ // afegim n a la pila
  retorna NULL // cap solució trobada

Cerca en profunditat (amb graf): arbre de cerca

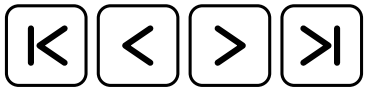
Graf d'estats



Arbre de cerca



It. 4: ABDEC E sol.!



Propietats: completa, subòptima, $O(|V|+|E|)$ temporal i espacial.

En general, pitjor que cerca en amplària!

dfs.py

```
#!/usr/bin/env python3
from collections import deque
G={'A':['B','C'],'B':['A','D'],'C':['A','E'],
→ 'D':['B','E'],'E':['C','D']}
def dfsi(G,s,m,t):
→O=deque(); O.append((s,[s]))
→while O:
→→s,path=O.pop()
→→if s==t: return path
→→if len(path)<=m:
→→→for n in list(reversed(G[s])):
→→→→O.append((n,path+[n]))
print(dfsi(G,'A',3,'E'))
```

dfs.py.out

```
['A', 'B', 'D', 'E']
```

bt.py

```
#!/usr/bin/env python3
G={'A':['B','C'],'B':['A','D'],'C':['A','E'],
→ 'D':['B','E'],'E':['C','D']}
def bt(G,s,m,t):
→if s==t: return [s]
→if m==0: return None
→for i in [0,1]:
→→n=G[s][i]; path=bt(G,n,m-1,t)
→→if path!=None: return [s]+path
print(bt(G,'A',3,'E'))
```

bt.py.out

```
['A', 'B', 'D', 'E']
```

pi.py

```
#!/usr/bin/env python3
G={'A':['B','C'],'B':['A','D'],'C':['A','E'],
→ 'D':['B','E'],'E':['C','D']}
def dfs(G,s,m,t):
→if s==t: return [s]
→if m==0: return None
→for n in G[s]:
→→path=dfs(G,n,m-1,t)
→→if path!=None: return [s]+path
def pi(G,s,t):
→for m in range(len(G)):
→→path=dfs(G,s,m,t)
→→if path!=None: return path
print(pi(G,'A','E'))
```

pi.py.out

```
['A', 'C', 'E']
```