UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Cerca Primer-El-Millor: Cerca voraç[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

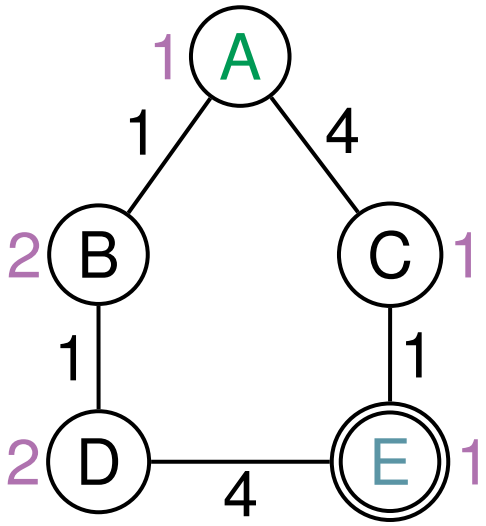
- Descriure l'algorisme de cerca Primer-El-Millor general.
- Aplicar cerca (Primer-El-Millor) voraç.
- Analitzar l'optimalitat i complexitat de cerca voraç.
- Il·lustrar la incompletesa de voraç amb cerca en arbre.

Índex

1 Introducció

Cerca Primer-El-Millor consisteix a enumerar camins fins a trobar una solució, prioritzant els de menor "cost" (f) i evitant cicles:

Graf d'estats

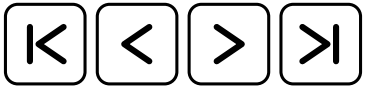


Cerca

A 1

AB 2
AC 1

ACE 1 sol.!



Primer-El-Millor generalitza A* permetent l'ús de qualsevol ***funció d'avaluació (heurística)*** f, no necessàriament del tipus $f = g + h$.

2 L'algorisme Primer-El-Millor [1]

BF(G, s', f**)** // ***Best-First;*** G, s', f ***funció d'avaluació***

$O = IniCua(s', f(s'))$ // ***Open: cua de prioritat*** f

$C = \emptyset$ // ***Closed:*** nodes explorats

mentre no $CuaBuida(O)$: // ***1r el millor:*** $s = \arg\min_{n \in O} f_n$

$s = Desencua(O)$ // desempats a favor d'objectius

si $Objectiu(s)$ **retorna** s // solució trobada!

$C = C \cup \{s\}$ // s explorat

per a tota $(s, n) \in Adjacents(G, s)$: // generació: n fill d's

$x = f(n)$ // possible f_n nou

si $n \notin C \cup O$: $Encua(O, n, f_n \triangleq x)$

si no si $n \in O$ **i** $x < f_n$: $Modcua(O, n, f_n \triangleq x)$

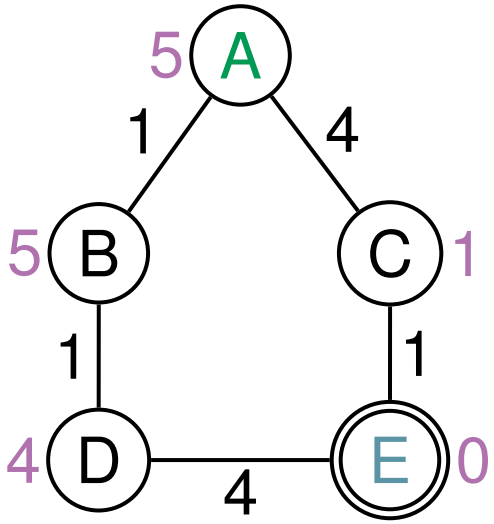
si no si $n \in C$ **i** $x < f_n$: $C = C \setminus \{n\}$; $Encua(O, n, f_n \triangleq x)$

retorna NULL // cap solució trobada

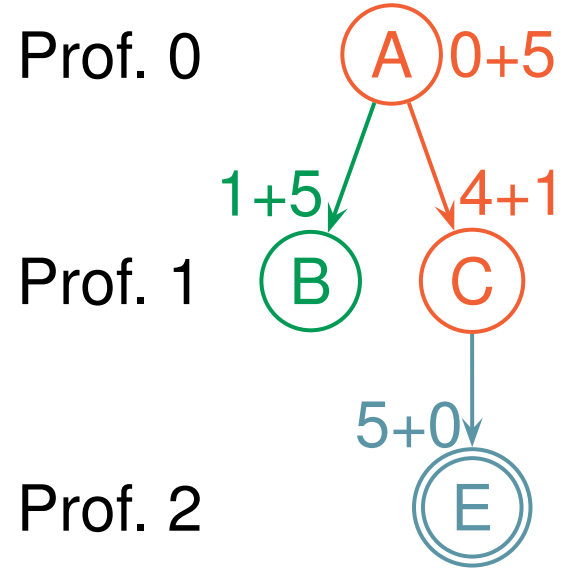
3 Cerca (Primer-El-Millor) voraç

Cerca (Primer-El-Millor) voraç consisteix a emprar $f = h$:

Graf d'estats

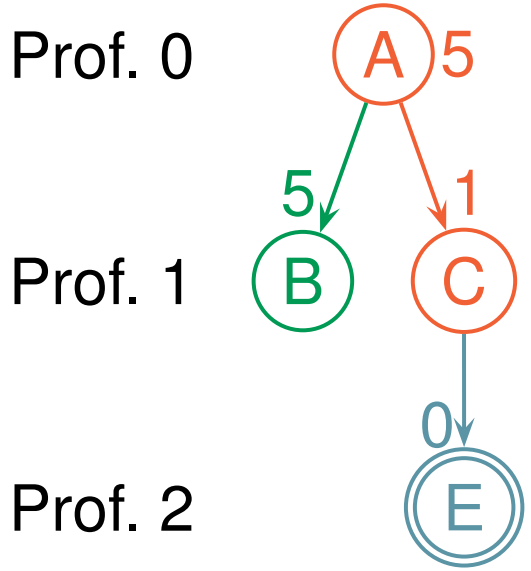


$f = g + h$ ***(A*)***

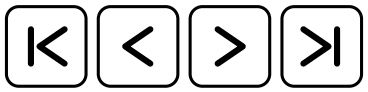


It 3: A5C5E5B6

$f = h$ ***(voraç)***



It 3: A5C1E0B5



Intuició: aproximar-se ràpidament a solucions.

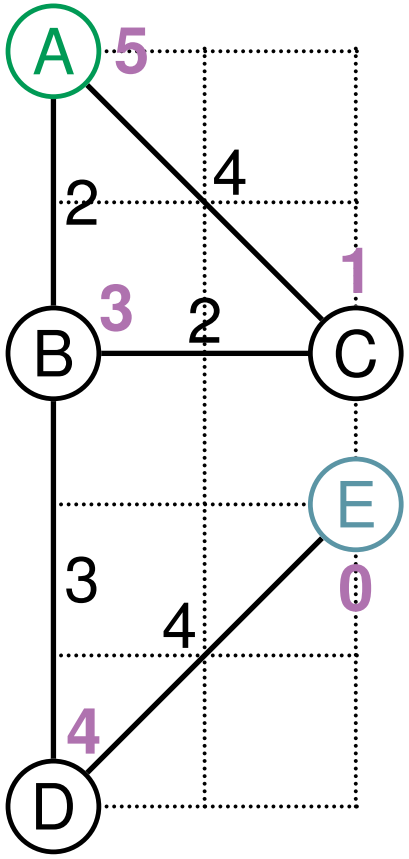
Optimalitat: completa en grafs finits i subòptima.

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

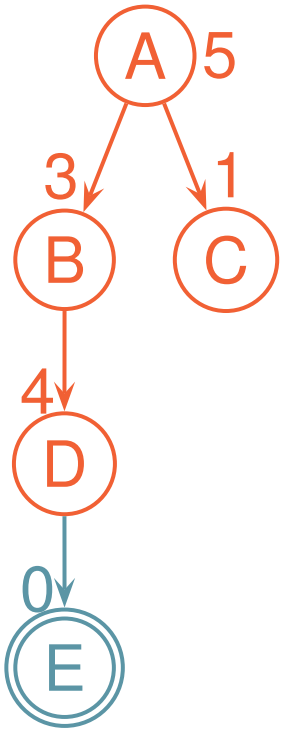
4 Voraç amb cerca en arbre

Voraç amb cerca en arbre [2] ($C = \emptyset$) és incompleta:

Graf d'estats

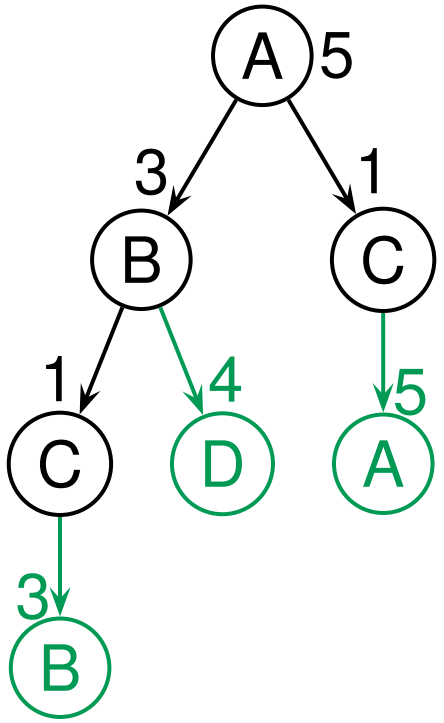


Cerca en graf



5: A5C1B3D4E0

Cerca en arbre



5^+: A5 $[B3C1]^+$ D4A5

5 Conclusions

Hem vist:

- L'algorisme de cerca Primer-El-Millor general.
- L'arbre de cerca (Primer-El-Millor) voraç.
- L'optimalitat i complexitat de cerca voraç.
- La incompletesa de voraç amb cerca en arbre.

Referències

[1] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

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

gbfs.py

```
#!/usr/bin/env python3
import heapq
G={'A':[('B',1),('C',4)],'B':[('A',1),('D',1)],
→ 'C':[('A',4),('E',1)],'D':[('B',1),('E',4)],
→ 'E':[('C',1),('D',4)]}
hstar={'A':5,'B':5,'C':1,'D':4,'E':0}
def bf(G,s,t,f):
→fs=f[s]; Od={s:(0,fs)}; Cd={} # Open and Closed g,f dict
→Oh=[]; heapq.heappush(Oh,(fs,s,[s])) # Open heap
→while Od:
→→s=None
→→while s not in Od: fs,s,path=heapq.heappop(Oh) # delete-min
→→gs,fs=Od[s]
→→if s==t: return gs,path
→→del Od[s]; Cd[s]=gs,fs
→→for n,wsn in G[s]:
→→→gn=gs+wsn; fn=f[n]
→→→if n in Cd:
→→→→if fn<Cd[n][1]: del Cd[n] # for variable f (eg with path)
→→→→else: continue
→→→elif n in Od and fn>=Od[n][0]: continue
→→→Od[n]=gn,fn; heapq.heappush(Oh,(fn,n,path+[n]))
print(bf(G,'A','E',hstar))
```

gbfs.py.out

```
(5, ['A', 'C', 'E'])
```