UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Best-First Search: Greedy Search

Alfons Juan
Jorge Civera

DSIIC

Departament de Sistemes Informàtics i Computació

Objectives

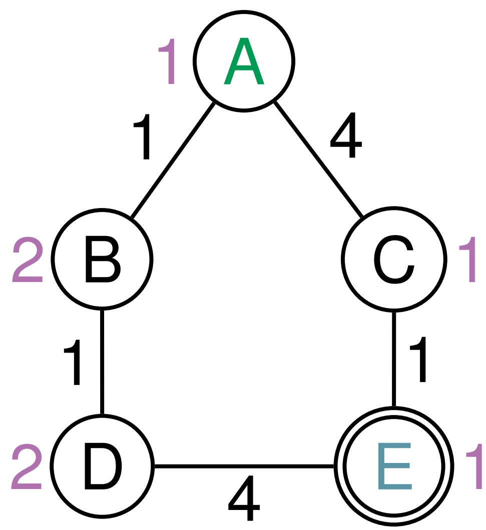
- Describe the algorithm of best-first search.
- Apply greedy search as an instance of best-first search.
- Analise the optimality and complexity of greedy search.
- Show the incompleteness of greedy tree search.

Contents

1 Introduction

Best-first search consists in enumerating paths until it finds a solution, prioritising those with lowest cost (f) and avoiding cycles:

State graph



Search

A 1

AB 2
AC 1

ACE 1 sol.!

Best-first search generalises A* allowing the use of any ***evaluation (heuristic) function*** f**,** not necessarily those in the form of $f = g+h$.

2 Best-first algorithm (graph search) [1]

BF(G, s', f) // G weighed graph, s', f ***evaluation function***

$O = InitQueue(s', f(s'))$ // ***Open: priority queue*** f

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

while not $EmptyQueue(O)$: // ***best-first:*** $s = \arg\min_{n \in O} f_n$

$s = Pop(O)$ // draws in favour of goal state

if $Goal(s)$ **return** s // solution found!

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

for all $(s, n) \in Adjacents(G, s)$: // generation: n is child of s

$x = f(n)$ // possibly new f_n

if $n \notin C \cup O$: $Push(O, n, f_n \triangleq x)$

else if $n \in O$ **and** $x < f_n$: $Update(O, n, f_n \triangleq x)$

else if $n \in C$ **and** $x < f_n$: $C = C \setminus \{n\}$; $Push(O, n, f_n \triangleq x)$

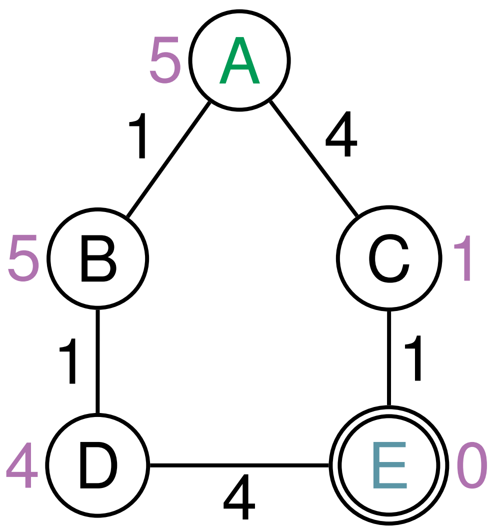
return NULL // solution not found

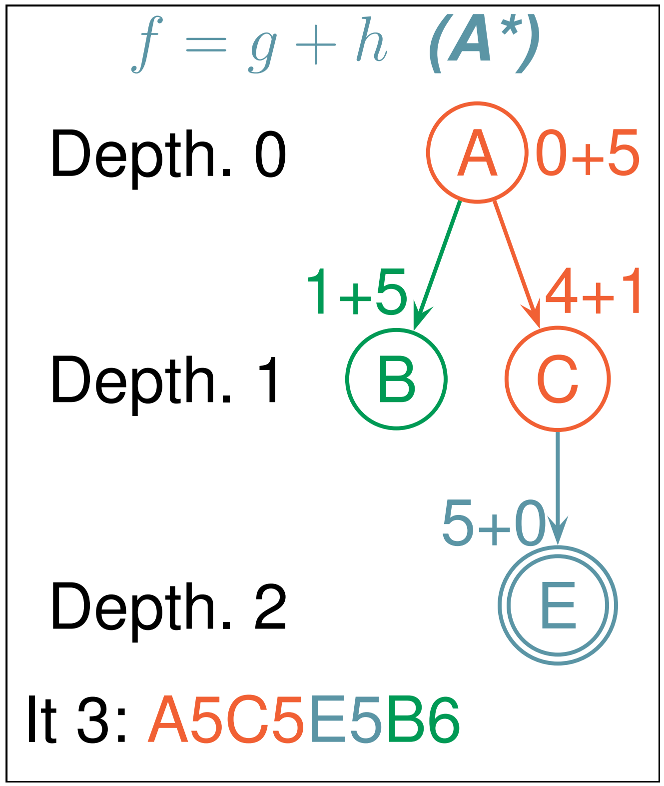
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

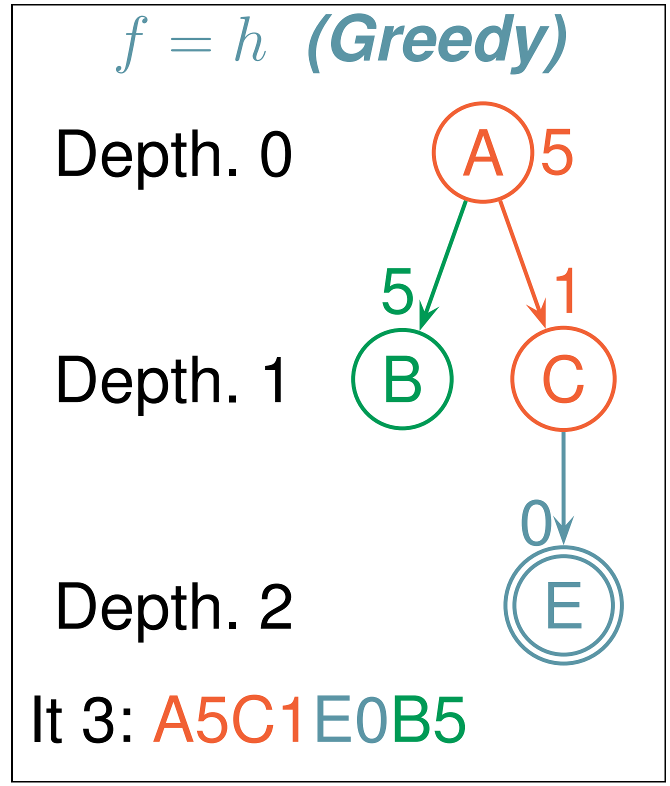
3 Greedy search (graph search)

Greedy search consists in using ***BF*** with $f = h$:

State graph







Intuition: to reach solution nodes quickly.

Optimality: complete in finite graphs and suboptimal.

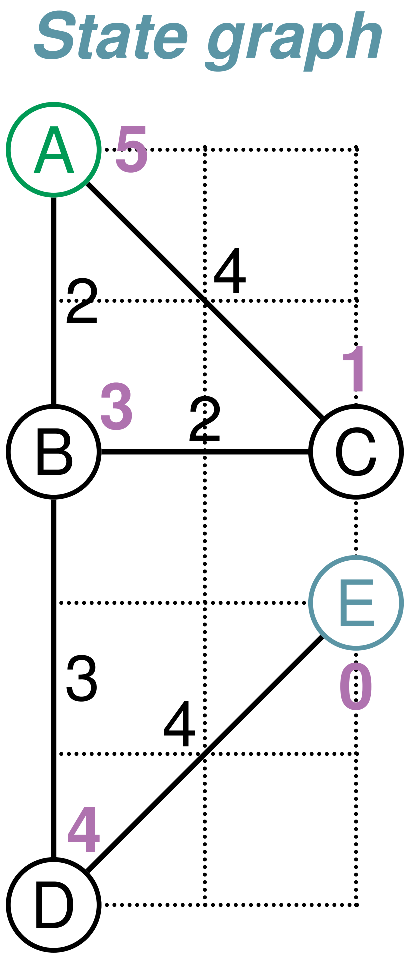
Complexity: $O(b^m)$ temporal and spatial (m maximum depth).

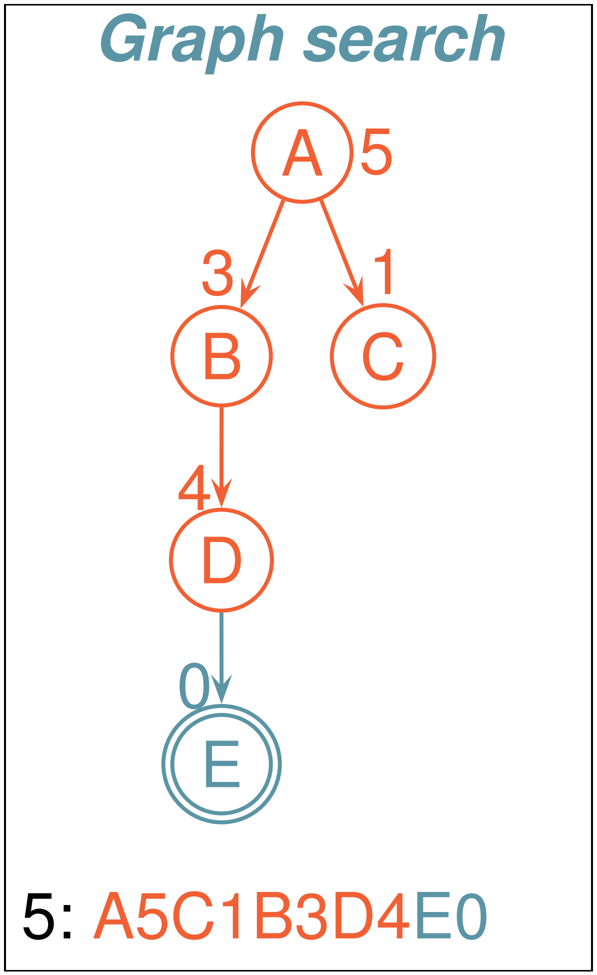
4 Best-first algorithm (tree search) [1]

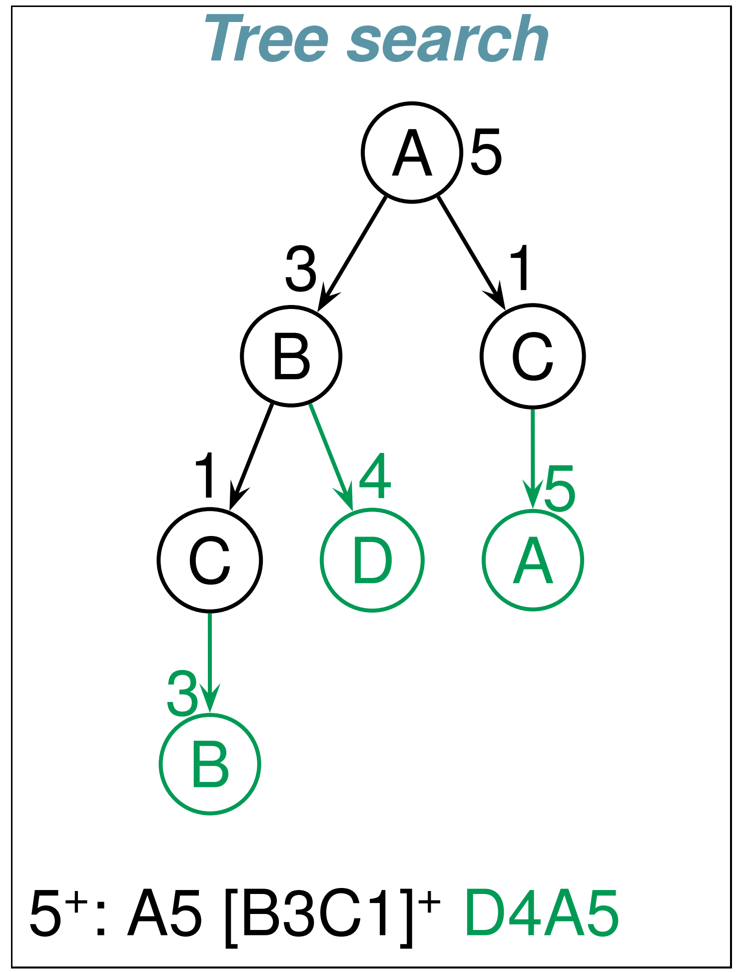
```
BF(G, s', f)                                   // G weighed graph, s', f evaluation function
  O = InitQueue(s', f(s'))                     // Open: priority queue f
  while not EmptyQueue(O):                     // best-first: s = arg min_{n∈O} f_n
    s = Pop(O)                                 // draws in favour of goal state
    if Goal(s) return s                        // solution found!
    for all (s, n) ∈ Adjacents(G, s):          // generation: n is child of s
      x = f(n)                                 // possibly new f_n
      if n ∉ O: Push(O, n, f_n ≜ x)
      else if n ∈ O and x < f_n: Update(O, n, f_n ≜ x)
  return NULL                                  // solution not found
```

5 Greedy search (tree search)

Greedy search with tree search [2] ($C = \emptyset$) is incomplete:







6 Conclusions

We have studied:

- The algorithm of best-first search, both graph and tree search.
- Greedy search with graph and tree search.
- Optimality and complexity of greedy search.
- Incompleteness of greedy tree search.

References

[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

```python
#!/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'])
```