AI algorithms: Optimal Search

A*
SMA*

Input:
A graph of nodes with costs for all connections, a start node S and a goal node G
A heuristic estimate h for each node (of distance to G)
A queue Q of possible paths
Output:
Cheapest path from S to G
Algorithm:

```
Q ← S
memsize ← meminit
while Q not empty AND G not reached by first path do
  P ← get (and remove) first path from Q
  while P has more children do
    if Total number of nodes in Q = memsize then
      Remove leaf with highest f from its path(s) in Q
      Remember (f of best forgotten child) in parent node
    end if
    p ← next child path of P
    Remove p if it contains a loop
    Calculate cost c
    Calculate f (= c + h of last node)
    if length(p) = memsize, not ending in G then
      f = infinity
    end if
    Add p to Q
  end while
  f of parent P ← minimum of f values of children (if > than current f)
  Sort Q according to f
  for all path P in Q do
    n ← last node of P
    if another path P2 contains n AND c of P ≥ c of P2 then
      Remove P from Q
    end if
  end for
end while
if G reached then
  Succes
else
  Failure, return best path that fits in memory
end if
```

AI algorithms: Optimal Search (2)

IDA*

Algorithm:

$Q \leftarrow S$
$fbound \leftarrow f$ of S
$fnew \leftarrow$ infinity
while Q not empty AND <u>G not reached</u> **do**
  $p \leftarrow$ get (and remove) first path from Q
  $P \leftarrow$ all paths to children of p
  Remove all paths from P containing loops
  Calculate cost c for each path in P
  Calculate f (= $c + h$ of last node) for each path in P
  $fnew \leftarrow$ <u>minimum of $fnew$ and smallest new f-value larger than $fbound$</u>
  Remove paths with $f > fbound$ from P
  Add paths of P to front of Q
end while
if G reached **then**
  Succes
else
  <u>Start over with $fbound \leftarrow fnew$</u>
end if