Uniform Cost Search

Wednesday, September 7, 2016 9:11 AM

Actions w/ costs

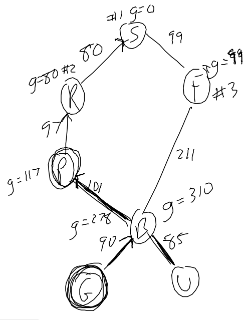
Instead of shortest # of action → lowest cost solutions

- optimal from cost perspective
- complete

g(n) - cost of state n from init state to n via the shortest known path

g(init) = 0

init: S
Goal: G



Action cost

Uniform cost search

g(n) – actual cost from init to state n ~~a~~ along the best known path.

Sort the open list with g(n)

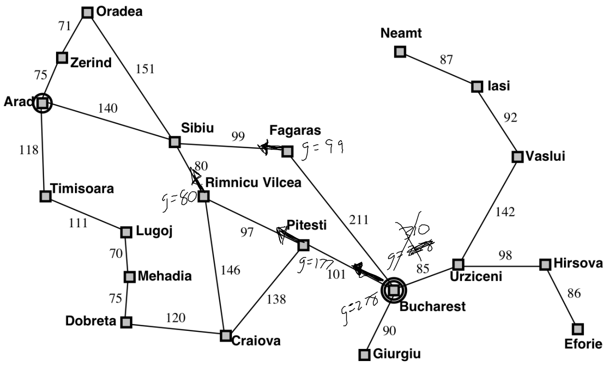
open list → priority Queue

Generic Search Algorithm

```
ops <— {…}
closed <— nil
open <— {initial-state}
current <— initial-state
WHILE (NOT isgoal(current) AND open ≠ NIL) DO:
    closed <— closed + {current}
    open <— open - {current} ⊕ ( successors(current, ops) - closed )
    current <— first(open)
END WHILE
IF isgoal(current) THEN report success!
ELSE report failure
```

insert sort based on g()
O(n)

Example: Romania



Chapter 3 6

Handling shortcuts

```
WHILE (NOT isgoal(current) AND open ≠ NIL) DO:
    closed <— closed + {current}
    FOREACH n ∊ successors(current, ops) DO:
        IF n is not on open or closed THEN DO:
            compute g(n)
            Insert n into open (ranked on g())
        ELSE IF n is on open AND n is reached by a shorter path THEN DO:
            n.parent <— current
            update g(n)
            resort open
    END FOREACH
    current <— pop(open)
END WHILE
```

visit

smaller g fn n

Delete + reinsert

O(n)

New

UCS is optimal & complete

→ but you cant jump straight to goal state
when successor function generates it.