lec#7

[1] Breadth First Search

(input → problem; outputs → Solution, Failure)

Function BFS (problem) returns Solution or Failure

node ← node with state = initial state (root), PathCost = 0

if problem. Goal-test (node.state)
  then return (solution. node).

else frontier ← FIFO queue with node as only element
  loop do
    iF frontier is Empty
    then return Failure.
    else لو ال frontier مش empty يبقى نبدأ اشيل ال node
      node ← pop (frontier) ← Shallowest node
      add node. state to explored.
      (علشان ماترجعلهاش تاني)
      بعد كده هعمل Expand ليها ↓

هعمل childnode لكل action. ← **For** each action **in** problem.actions **do**
        Child ← ChildNode (problem, node, action)
        if Child.State **not in** explored set **or** Frontier.
        then
          check if it's goal then **return** goal.
        else
          Frontier ← Insert (child)

- BFS is Complete ← بيمسكن الحلول الممكنة
  - BFS will find the shallowes one first (Shortest path)
- BFS is optimal (if the cost is increasing)
  عشان يجيب أقرب solution فلازم الـ cost متزايد مش هيبقى الـ cost

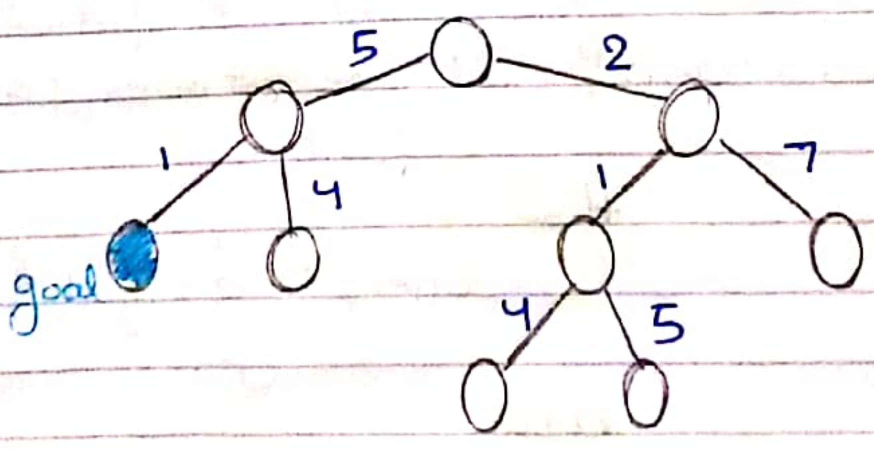
- b = branching factor
- Time complexity $O(b^d)$
- space complexity $O(b^d)$

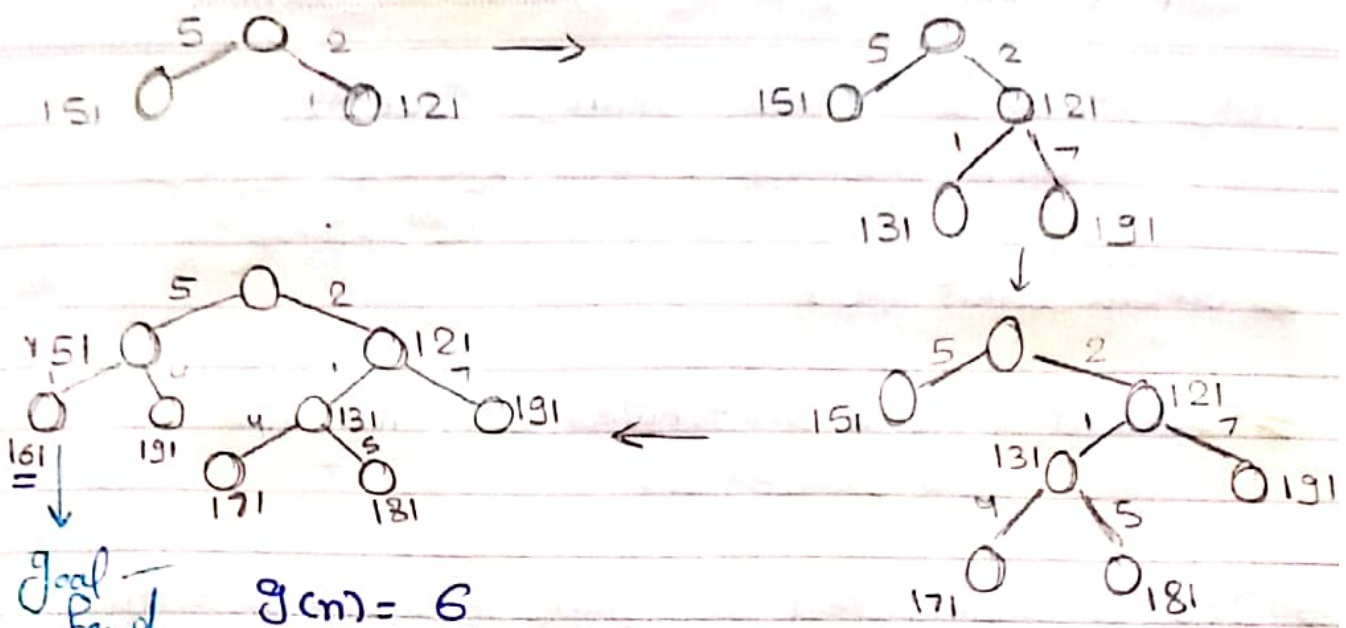
[E] Uniform Cost Search

بلما هختار الطريق الأقصر هاخد في اعتباري الـ minimum Cost

- UCS expands with **lowest cost path** $g(n)$
- frontier: priority queue ↘
  لما بيسحب ليس بيسحب أقل قيمة في القيم اللي عنده
- if all cost steps are equall then UCS = BFS

Example





Goal found g(n) = 6

← رجعلى ال code في ال Slides اهم.

- UCS is complete and optimal.
- Time Complexity = space complexity = $O(b^{d+1})$

3 Depth-first Search (DFS)

- expands the deepest node in the Frontier
- Frontier: LIFO stack

- incomplete and non-optimal

اول ما بيلاقي حل في المسار اللي هو فيه مش بيبص على باقي المسارات

- time Complexity = $O(b^m)$ in worst case ⚠
- Space complexity = $O(bm)$ linear space

it stores only a single path from root to leaf

لما الـ depth ممكن يكون لبرا ان ومعقوش
من غير النهاية يعني ما عيلا دارة ؟

[4] Depth-limited search (DLS)

اقدر يوصل لحد معينه وبعدها له ملقا الحل هناك
يعمل Backtrack

- depth limit (ℓ)
- nodes at depth ℓ are treated as if they have no successors.
- DLS is implemented as simple recursive algorithm
- DLS is incomplete and non optimal.

terminate → Standard Failure: لما يكون مفيش حل
terminate → Cutoff value: لما يوصل الـ depth عند الـ limit

Time Complexity: $O(b^{\ell})$
Space Complexity: $O(b\ell)$

⚠ ← ركز على الكود في الـ slides

كده اصبح عندي مشكلة، هي اني مش عارف الـ Solution هيجي قبل ولا بعد الـ limit.

لو فتبا الـ limit واحدة واحدة ونزود لو رجع بـ cutoff
هنزود لو رجع بـ Failure هسيبه زي ما هو

[5] Iterative Deeping Search (IDS)

- Combines the Benefits of BFS and DFS

Bfs

- Complete
- finite b

جاب الأقرب

dfs

- memory requirants

جاب الأسرع

! ← ليه ده الأسود في ال Slides.

- IDS is complete and optimal

Time Complexity: $O(b^d)$

Space Complexity: $O(bd)$

وده بيشتغل عليه أحسن technique

للBlind Search