CSE 318 05/06/2023

Number of states will be high, 9! for 3×3 grid. So, no need to maintain "visited" state since memory requirement will be very high. Thus, we will visit a node multiple times if necessary. DFS will not give optimal soln. On the other hand, BFS is too costly. Thus, we will use A* search.

If we can design a proper heuristic function then we can reach the goal node by exploring least nodes compared to Dijkstra and BFS. This search algorithm is called Best First search.

h(n) must be admissible, that is,

$$h(n) \leq g(n, t)$$

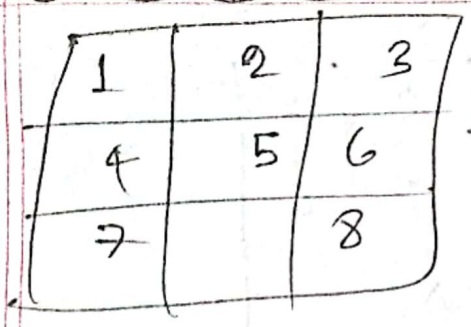
h(n) = estimated cost from n to goal

g(n, t) = actual cost from n to goal

Search space pruning

The closer h(n) is to g(n, t) the better.

Hamming Distance



without blank → $h(n) = 1$

with blank → $h(n) = 2$, overestimates, we can reach goal with just 1 move.

∴ "$h(n)$ with blank" is not admissible.

Manhattan distance is better; #explored-nodes is less.

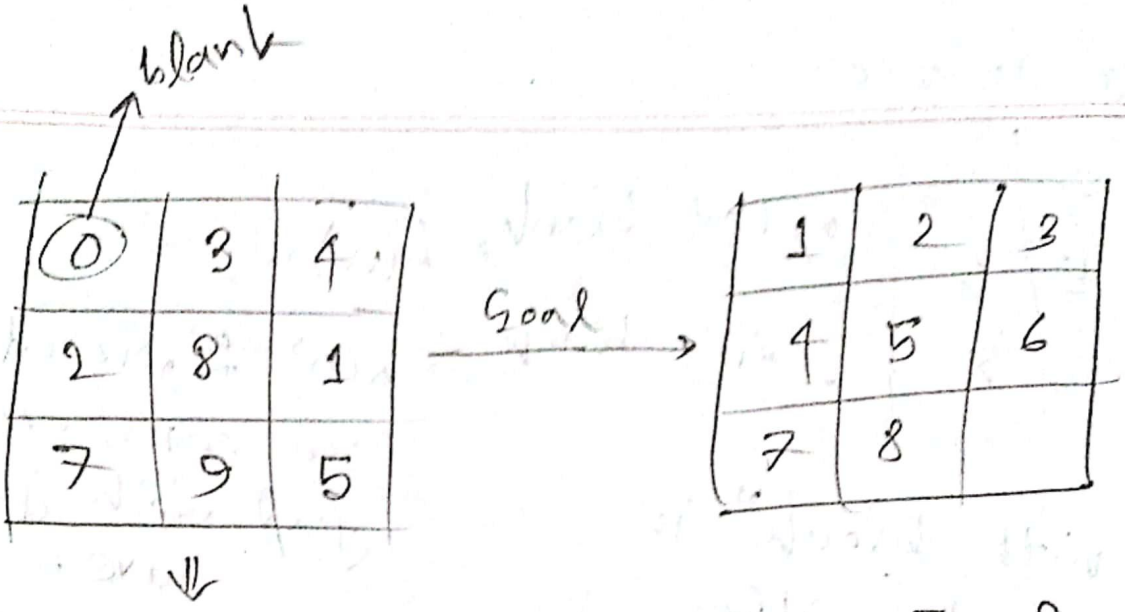
$\forall n,\ h_1(n) > h_2(n) \rightarrow$ $h_1(n)$ is better than $h_2(n)$.

Linear Conflict, NOT NEEDED FOR LAB

Expand → When a node is out of the queue and its neighbours are pushed

Explored → The neighbors just pushed are explored, not expanded

One child of a node will be identical to its ~~the~~ grandparent. So, we can optimize a little by checking this equality and pushing to the queue only if this is not the case.



row-major: $0\ \frac{3}{2}\ \frac{4}{2}\ \frac{2}{1}\ \frac{8}{3}\ \frac{1}{0}\ \frac{7}{1}\ \frac{9}{1}\ \frac{5}{0}$

$$\#\text{inversions} = 2+2+1+3+0+1+1+0$$
$$= 16$$

Left-right movement ⟶ No change in inversion

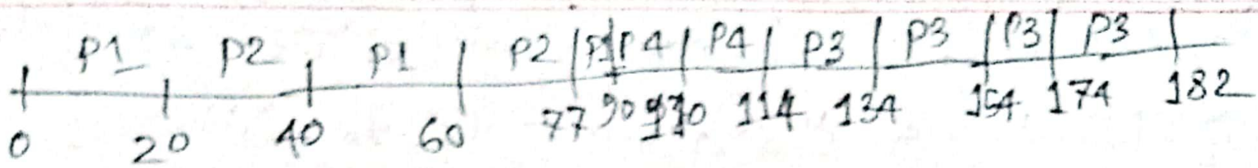
Up-Down " ⟶ Change in inversion

~~Applicable only if~~

~~"n" is odd~~

When "n" is odd, invariant → Inversion Count is always even

When "n" is even, invariant → IC + row-distance (blank) is always even



1 P1 → P2

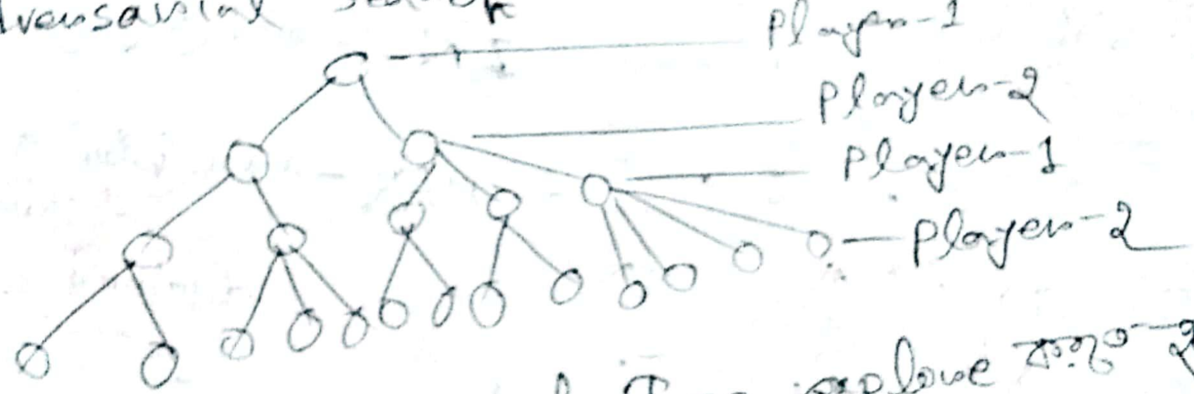
2 P2 → P1

19/06/2023

CSE318

Gain Tree [প্রত্যেক depth পর্যন্ত যেতে পারে]

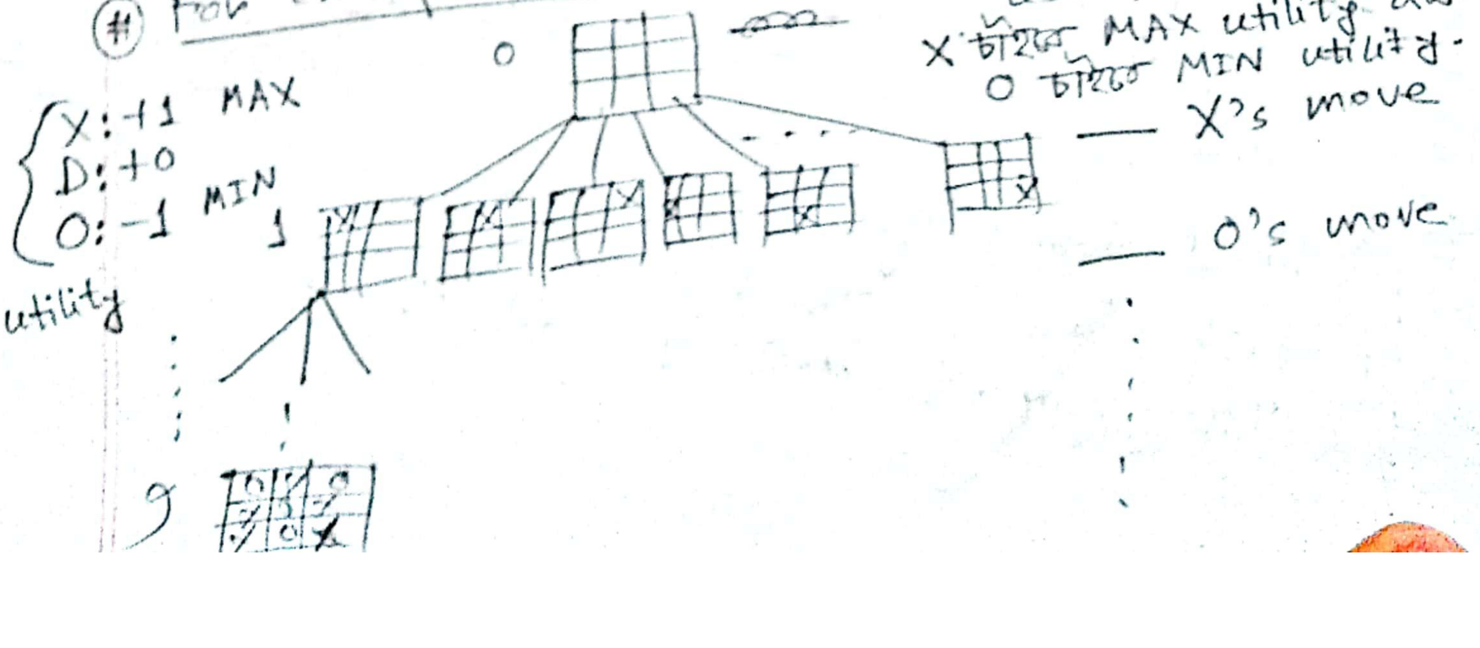
Adversarial Search



যেখানে complete Search Tree explore করতে হয় না!

For example, Tic-tac-toe:

Both player's will use best utility.
X চাইবে MAX utility and
O চাইবে MIN utility.

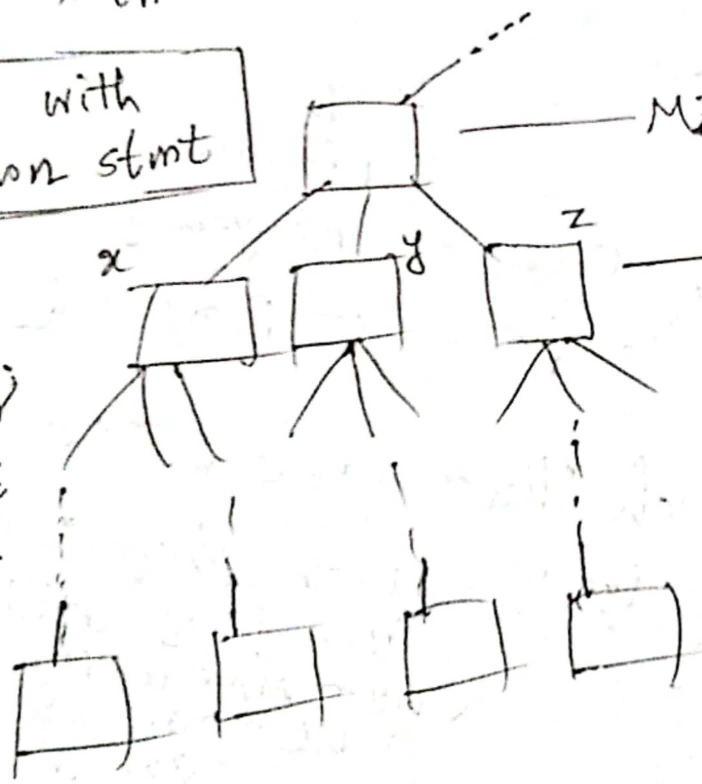


→ X এমন একটি terminal node-এ গেছে, জিতেছে, সেটার value MAX. Similarly, for 'O'.

⇒ এই game-টি কখনো মাঝে শেষ হবে না। শেষ পর্যন্ত যেতে হবে। সুতরাং, Terminal Node ছাড়া অন্য কোনো Node-এর কোনো Utility থাকবে না। Leaf পর্যন্ত গিয়ে utility check

→ then decide which child to expand.

DFS with return stmt
↓
Terminal Node: যদি কেউ একজন জিতে যায় / draw হয়

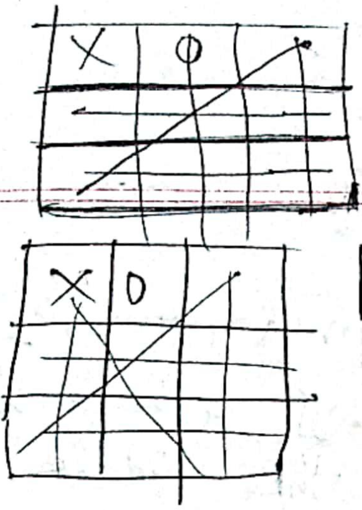


MIN – min (x, y, z)

MAX – max (value of all children obtained recursively)

[The origin of all values are these leafs]

Optimal Gameplay
- Both players are best
- The complete path to reach the goal



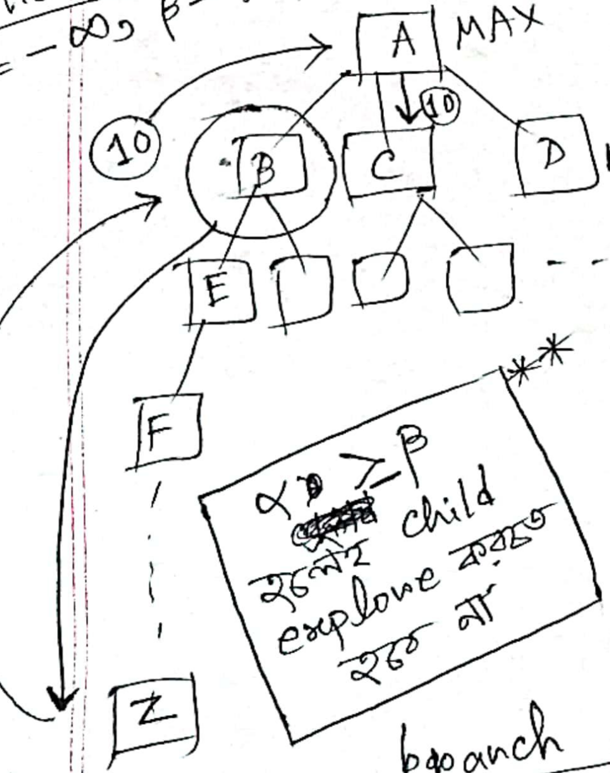
heuristic → Non-leaf Node-এর Score calculate করার জন্যে needed.

favourable for X

যখন আমরা মাঝখানেই cutoff করে দিব।

X : 6
O : 5 } +1

Initially, $\alpha = -\infty$, $\beta = +\infty$



A-B-E-F....-Z Path দিয়ে visit করে $\alpha = 10$ হলো। A ~~এখন~~ C-কে recursively α pass করছে। C, α-র value use করে Pruning করবে। এখন, C যদি ~~তার~~ একটি child traverse করে $\beta = 5$ পায়, ~~তাহলে~~ C ~~তার~~ অন্য কোনো child create করবে না। ∴ A will Prune ~~all of its children~~

branch A-C....

* Parent যদি MIN/MAX হয় → β/α-তে value দিবে
* Comparison হবে C and A-এর value-র মধ্যে

কারণ A, ~~এর~~ অন্য ~~আরেকটি~~ branch থেকে ≥ 10 পায়। যেহেতু A is MAX, তাই A-C branch নিবে না।

Assignment:

- Implement Minimax using given heuristics for cutting off a branch in the middle

- α/β Pruning ~~with~~
- NO NEED TO ~~WE~~ VISIT COMPLETE SEARCH TREE