CS 5100 01/23

Last time

heuristic function

$h(n)$: estimated cost from node n to a goal node

Previously in UCS

$g(n)$: total path cost from the start node to node n

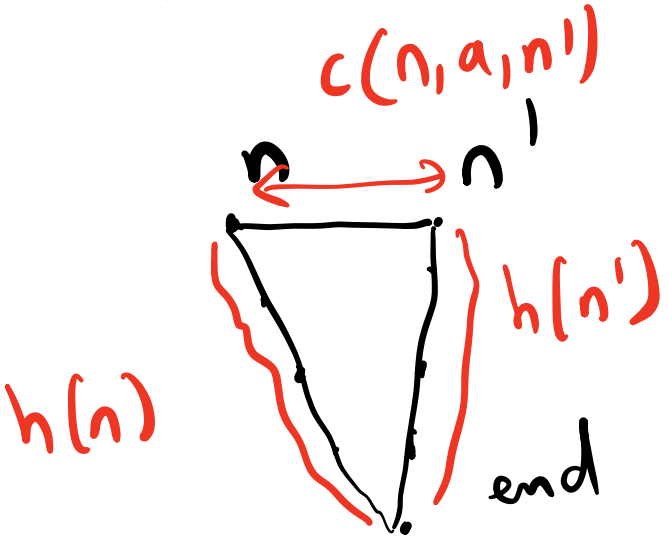
$$f(n) = g(n) + h(n)$$

total estimated cost from start node to a goal node while going through node n

Conditions on a heuristic function to get optimality

1) Admissibility: never overestimate the cost to reach a goal node

2) Consistency (triangle inequality)



$$h(n) \leq h(n') + c(n,a,n')$$

$\forall n, n'$

Consistency $\Rightarrow$ admissibility

(by induction)

* The values of $f(n)$ are non-decreasing along any path.

Take a node n' which is a successor of node n, we need to have

$$f(n') \geq f(n) = g(n) + h(n)$$

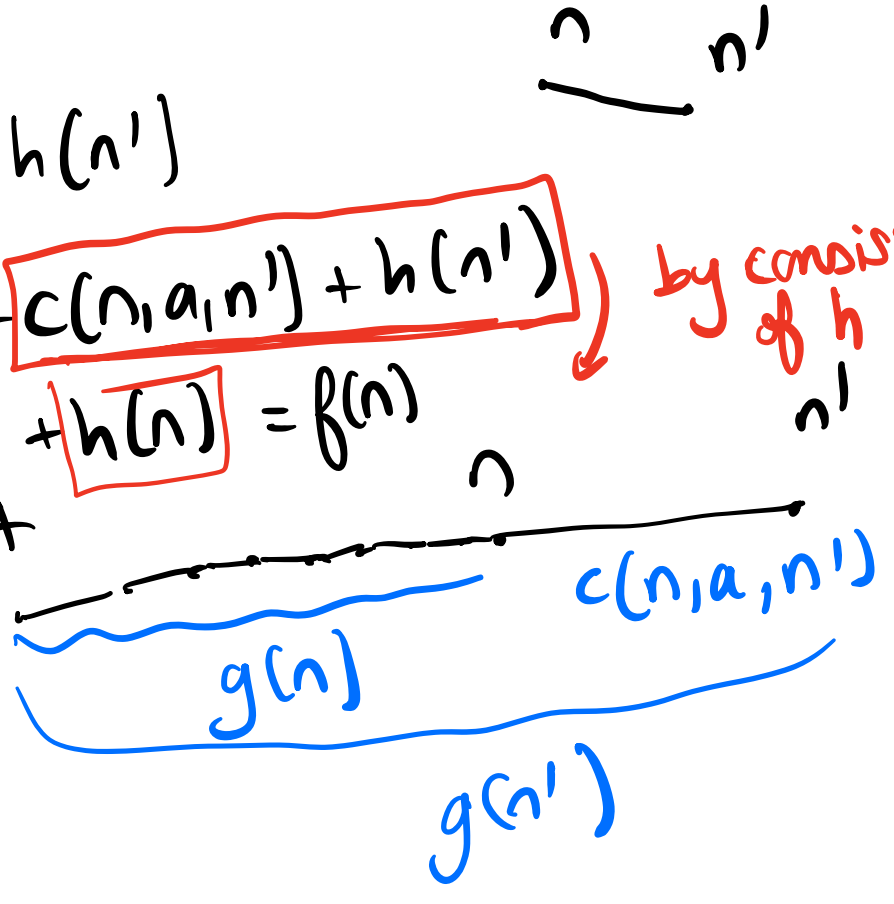
(Show this as exercise)

$$f(n') = g(n') + h(n')$$

$$= g(n) + c(n, a, n') + h(n')$$

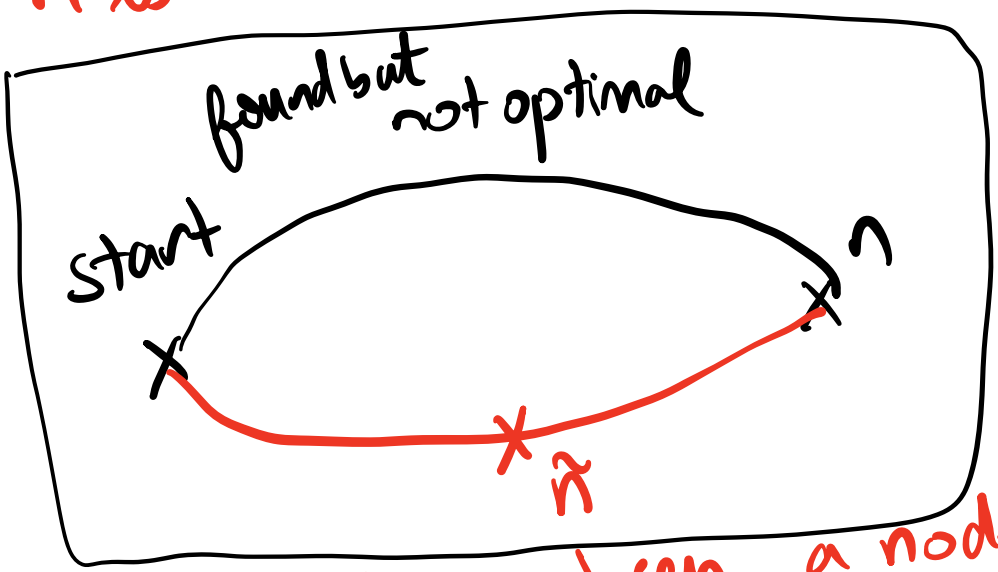
$$\geq g(n) + h(n) = f(n)$$

by consistency of h



* Show that whenever a node is selected for expansion (using priority queue with $f(n) = g(n) + h(n)$), the optimal path to node n has been found

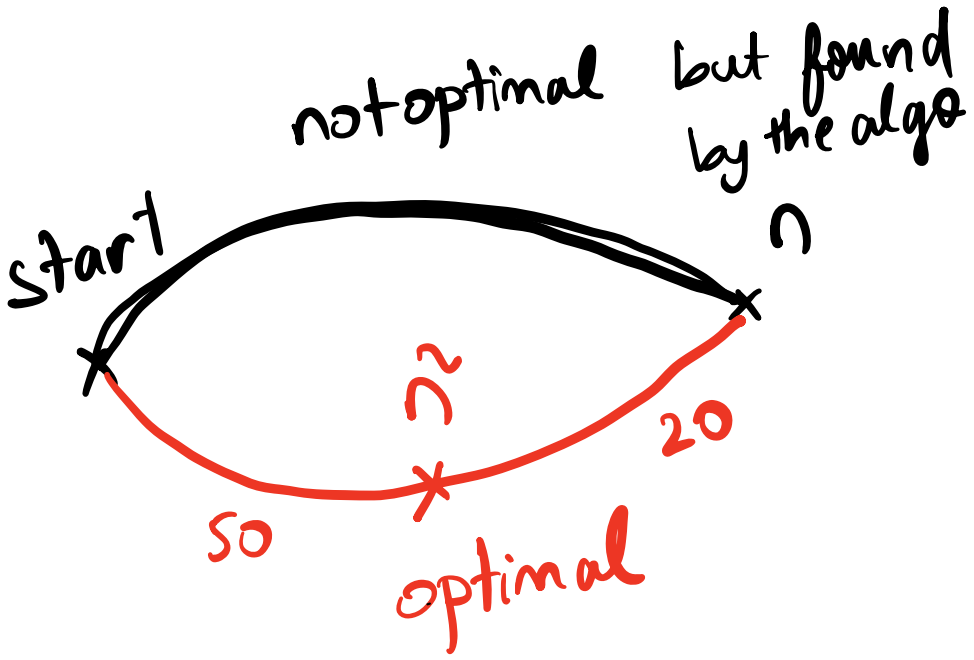
Assume that the path found to node n is not the optimal path



there must have been a node $\hat{n}$ on the optimal path from start to n

$$f(\hat{n}) \leq f(n)$$ (From the previous exercise)

$\hat{n}$ must be popped first from the priority queue

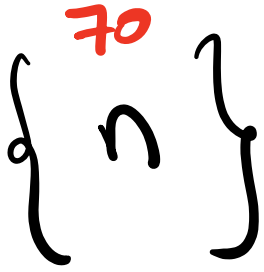


$$f(\tilde{n}) \leq f(n)$$

priority queue

70

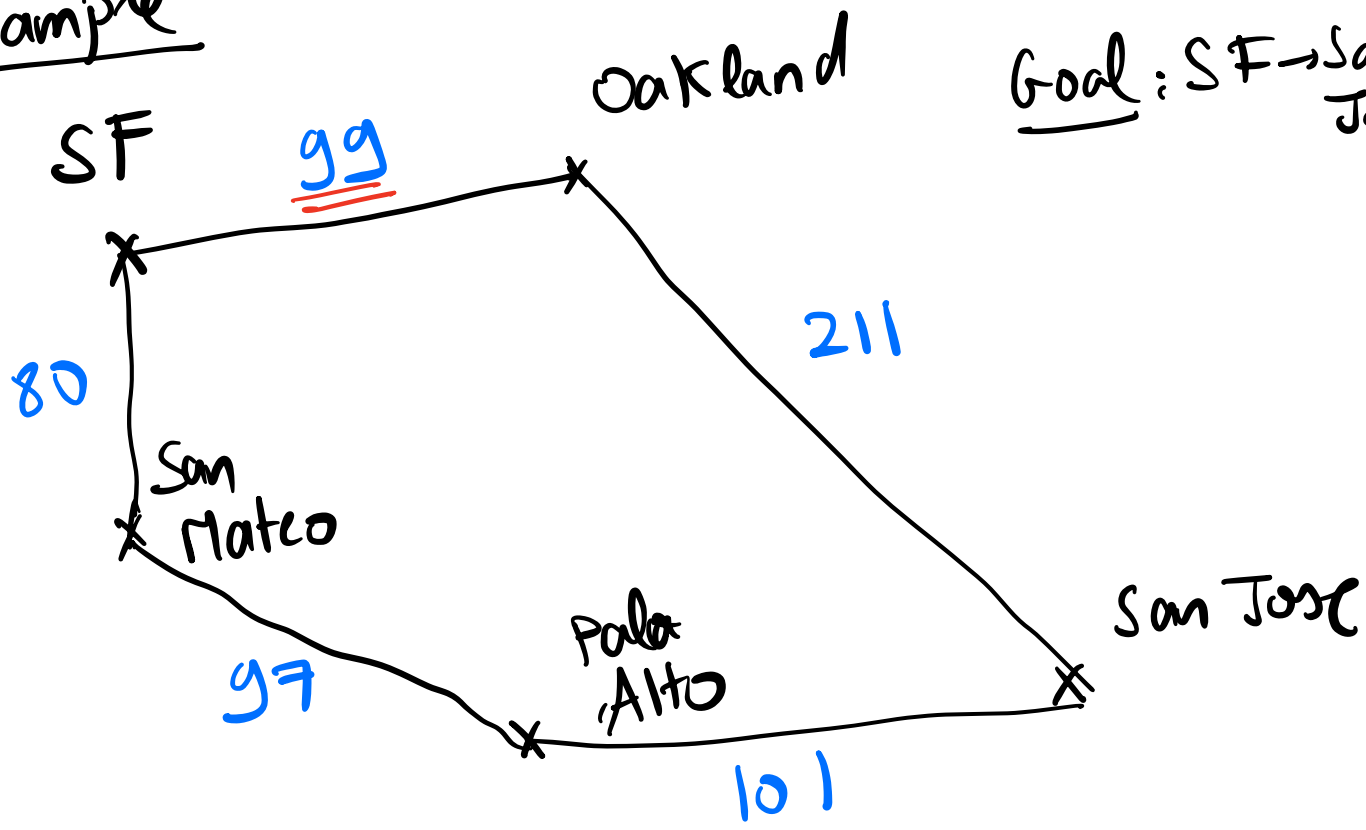
* pop $\tilde{n}$ before n
* explore successors of $\tilde{n}$ → find n



Algorithm : A* algorithm

(identical to UCS but we're using the Function $f(n) = g(n) + h(n)$ when considering the priority queue instead of only using $g(n)$

Example



Use straight line distance heuristic

$h_{SF \to SJ} = 200$

$h_{OA \to SJ} = 150$

→ $h_{SM \to SJ} = 120$

$h_{PA \to SJ} = 70$

node = SF

frontier = {SF} 200

explored = {}

loop

node popped = SF (not goal)

explored = {SF}

loop actions

child = San Mateo

frontier = {San Mateo} 80+120=200

child = Oakland

frontier = {San Mateo, Oakland} 200

99+150 = 249

node popped = San Mateo (not goal)

explored = {SF, San Mateo}

loop actions

child = Palo Alto

frontier = {Oakland, Palo Alto} 249

80+97+70 = 247

node popped = Palo Alto (not goal)

explored = {SF, San Mateo, Palo Alto}

loop actions

child = San Jose (not goal)

24g 278+0

frontier = {Oakland, San Jose}

node popped = Oakland

explored = {SF, San Mateo, PA, Oakland}

loop actions

child = San Jose

don't do anything

node popped = San Jose (goal node)

Exercise (code on canvas)

start at value 1

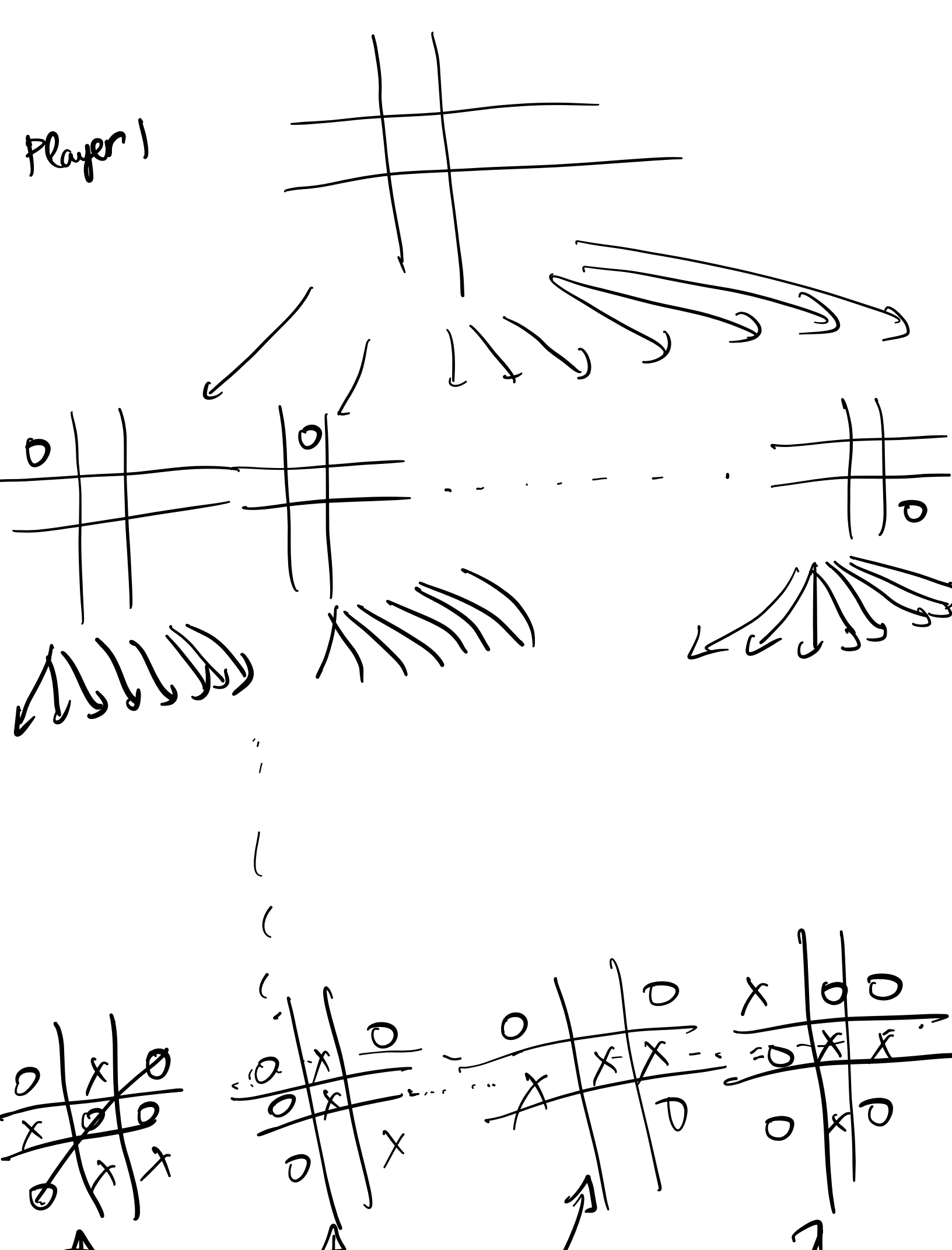
actions → either add 1
→ multiply by 2

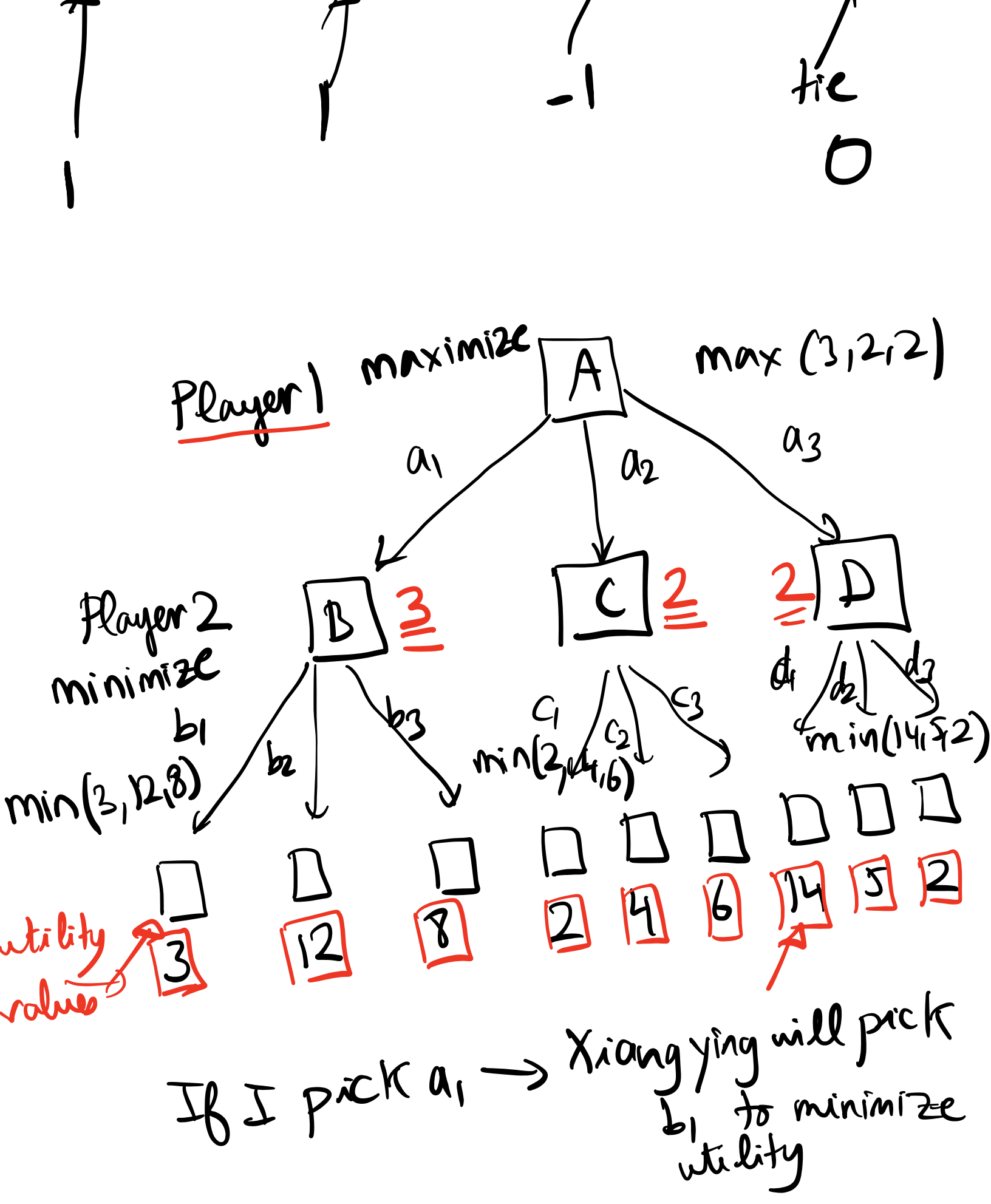
Goal: value n

Games like chess, whenever I am at some state, my action will really depend on what my opponent will also be doing. (the action that I choose will give me an advantage, some positive utility)

* Initial state
* Actions(s)
* Result(s, a) (transition model)
  - s → state
* Player(s)
* Utility(s, p) (positive if great for me and negative if bad for me)
* terminal_test(s)
  - allowing me to check whether the game is done

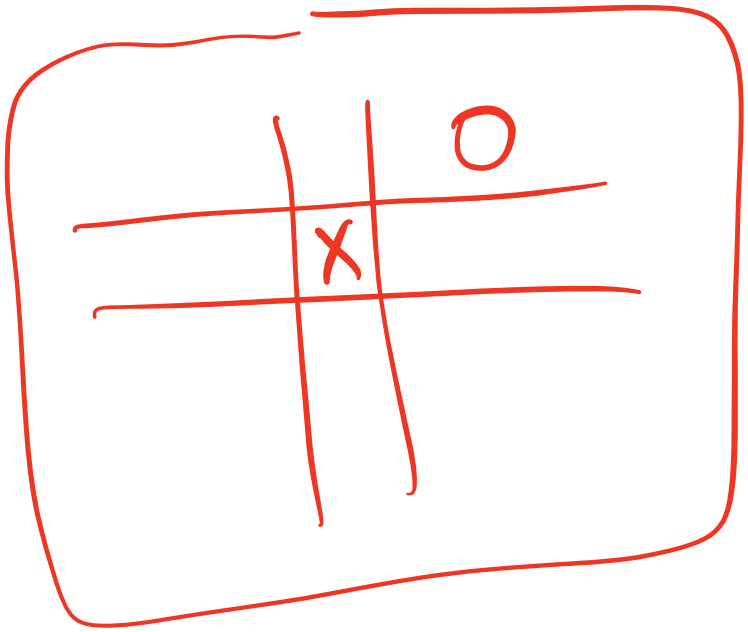
(good for my opponent)


Player 1


1
1
-1
tie
0
Player 1
maximize
A
max (3,2,2)
a_1
a_2
a_3
Player 2
minimize
B
3
C
2
2
D
b_1
b_2
b_3
c_1
c_2
c_3
d_1
d_2
d_3
min(3,12,8)
min(2,4,6)
min(14,5,2)
3
12
8
2
4
6
14
5
2
utility
values
If I pick a_1 → Xiangying will pick b_1 to minimize utility

If I pick a_2 → Xiangying will pick c_1

If I pick a_3 → Xiangying pick d_3



a_1
b_1

→ 3