Pytest for trading-back

main.py — trading_hub

back_storage

order_storage

back_utils

PQ

minPQ
maxPQ

Algo A4 : 8 Puzzle

Best first search

search node → prev_node

board state

# of moves made so far

oth node

init_board

0 moves

→ null_Parent

Step 1 → insert oth node into PQ

2 Delete (min) priority (manhattan or hamming), insert all neighbor nodes into PQ

neighbor → move from dequeued node

3+ Repeat until dequeued node matches goal board

parent → child

array rep ← int div

parent = i/2

child = 2i+1 (11)

2i

How to determine priority?

Hamming → # of blocks in wrong position
+
# of moves made so far

Manhattan → $\sum_{i=1}^{n}$ of manhattan distances
(vertical + horizontal)
to goal position
+
# of moves so far

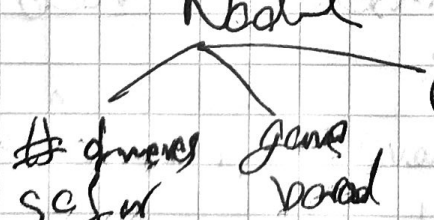
* Don't count blank

Key obs

total # of moves is at least the current priority

Optimization #1

Don't enqueue neighbor if it is the same as the predecessor



Opt 2 → Cache/pre-compute manhattan

priority value → save in an instance variable

Game Tree

root → [init_state] → null parent
→ ① moves
→ game init state

- maintain leaf node in PQ

In each step → deque min priority
→ enqueue both to game tree & PQ