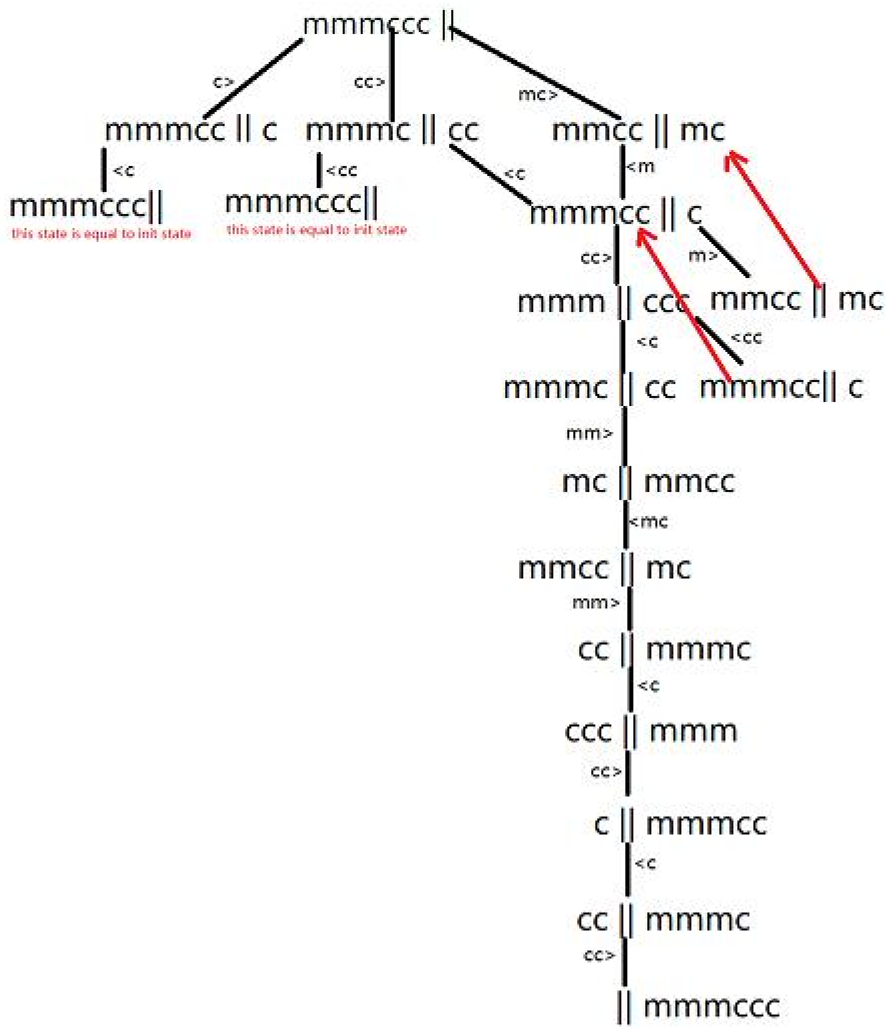
Problem 1: Formulating a Search Problem (20 marks)

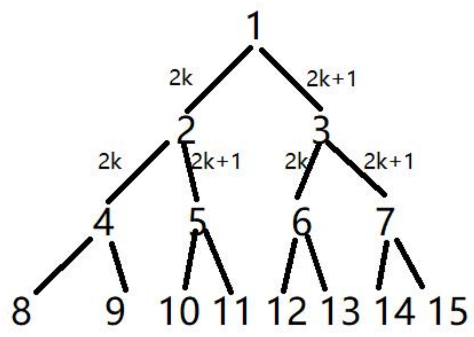
Q1: Say m for missionary, c for for cannibal and || stands for the river.
The initial state equals to mmmccc || , the goal state is || mmmccc.
The all possible actions are: 1. m> missionary the cross the river (<m cross back the river, these two action in my opinion are the same actions), 2. c> cannibal cross the river alone. 3. mc> missionary & cannibal cross the river together. 4. mm> two missionaries cross the river. 5. cc> two cannibals cross the river together.
Q2: One approach is on the below:
constraint: the number of cannibal cannot outnumber the number of missionary in the both river side



Q3: In my opinion, DFS is the best choice for this problem. Because, if we programme it, DFS is a memory saving approach compare to BFS and uniform search and as for the greedy and A* search, we don't need to set h(n) heuristic function and g(n).

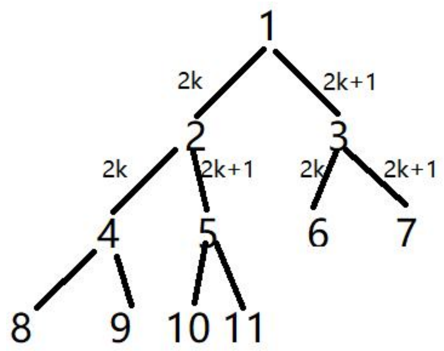
Problem 2: Uninformed Search (40 marks)

Q1: The portion is shown as below:

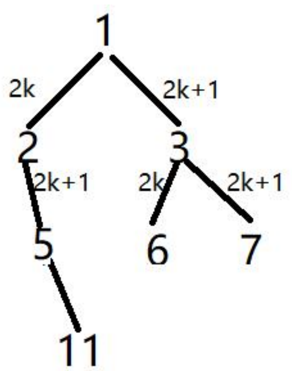


Q2:

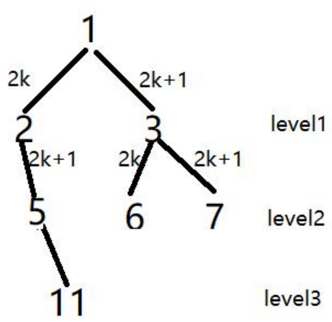
BFS:



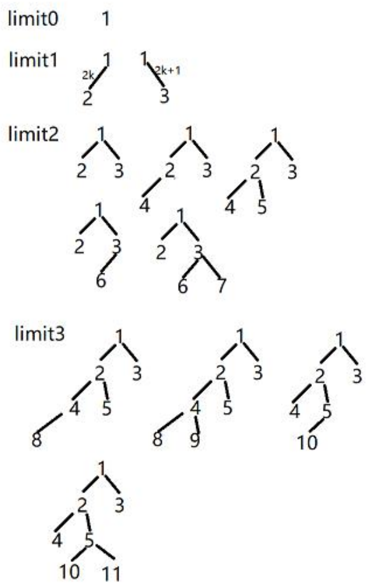
depth-first search:



depth-limited search with limit 3:



iterative deepening search:



Q3:

If the goal state is a large number, for example, 32768 or pow(2,50) , and we only have limited memory. Tree-like DFS will be used to solve this problem. Because, for BFS and graph-DFS we need to save the visited nodes into memory, but in our case, it is hard to save such a huge number into our memory, and it is an acyclic problem, so I vote for the tree-like DFS which discard visited nodes that have no descendants and save memory.

Problem 3: Heuristic Search (40 marks)

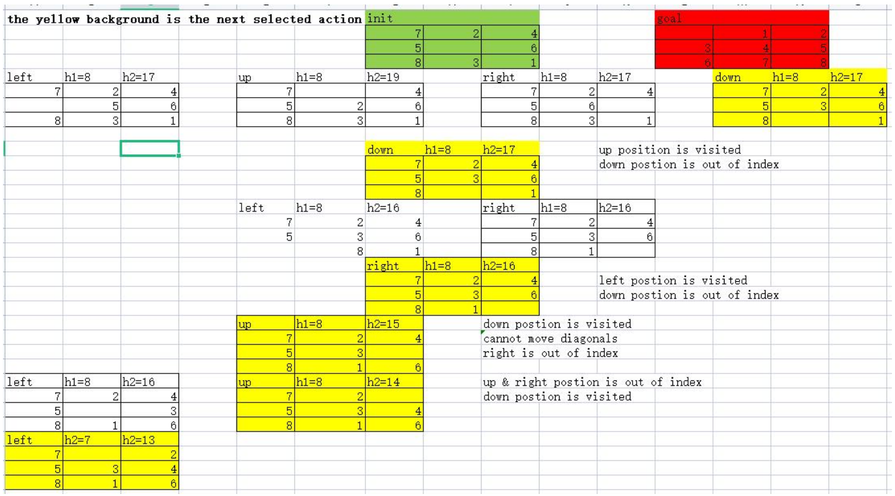
Q1:

In my opinion, both h1 and h2 are admissible. Because the heuristic function is to estimate the cost of the cheapest path. h1 can tell us the state of tiles, are they reached their goal states. On the other hand, h2 is also an admissible way, it shows the closest route to their goal state.

Q2:

Comparing with h1 and h2, h2 is a better function, since it generate the fewer search nodes. For example, for the level one, the h2 in four nodes are same(8), but h2 returns two same results(17 and 19). So, in the every search loop, we can find the fewer nodes.

Q3: greedy search with the heuristic function h2 to solve for this problem, if h2 is same, I always choose the last one action.



Q4: A* search with the heuristic function h2 to solve for this problem, in this case, gn is the depth of the search tree, every time we expand one node, then gn = gn+1. I pushed the python solution for this 8-puzzle problem into the github: https://github.com/3egg/HKBU-HW/blob/main/7015/WrittenAssignment.py Feel free to check it