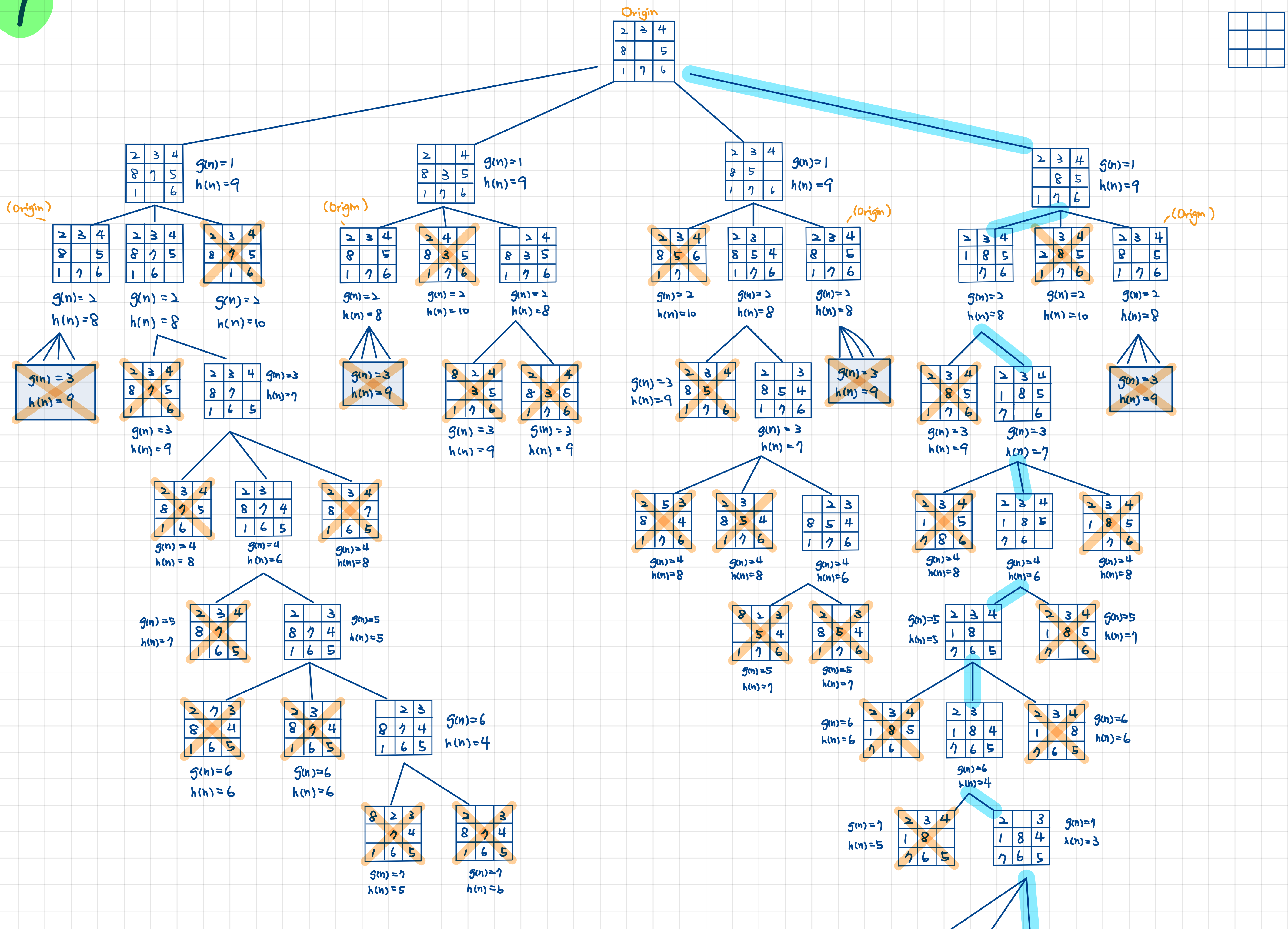
Theory of Computer Games

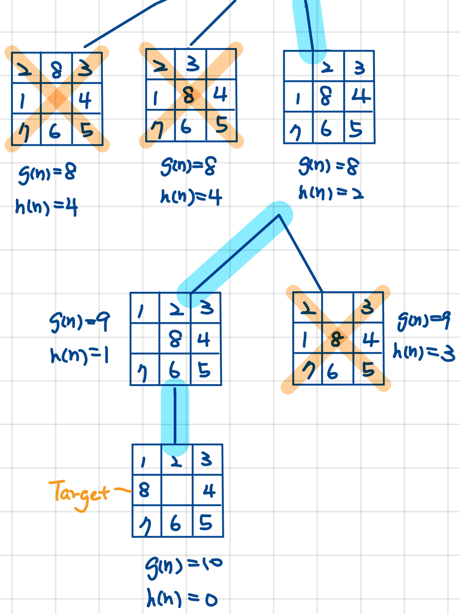
Homework #2

311605004
劉子齊

1



Origin
g(n)=1
h(n)=9
g(n)=1
h(n)=9
g(n)=1
h(n)=9
g(n)=1
h(n)=9
(Origin)
g(n)=2
h(n)=8
g(n)=2
h(n)=8
g(n)=2
h(n)=10
(Origin)
g(n)=2
h(n)=8
g(n)=2
h(n)=10
g(n)=2
h(n)=8
g(n)=2
h(n)=10
g(n)=2
h(n)=8
(Origin)
g(n)=2
h(n)=8
g(n)=2
h(n)=8
g(n)=2
h(n)=10
(Origin)
g(n)=2
h(n)=8
g(n)=3
h(n)=9
g(n)=3
h(n)=9
g(n)=3
h(n)=7
g(n)=3
h(n)=9
g(n)=3
h(n)=9
g(n)=3
h(n)=9
g(n)=3
h(n)=9
g(n)=3
h(n)=7
g(n)=3
h(n)=9
g(n)=3
h(n)=9
g(n)=3
h(n)=7
g(n)=3
h(n)=9
g(n)=4
h(n)=8
g(n)=4
h(n)=6
g(n)=4
h(n)=8
g(n)=4
h(n)=8
g(n)=4
h(n)=8
g(n)=4
h(n)=6
g(n)=4
h(n)=8
g(n)=4
h(n)=6
g(n)=4
h(n)=8
g(n)=5
h(n)=7
g(n)=5
h(n)=5
g(n)=5
h(n)=7
g(n)=5
h(n)=7
g(n)=5
h(n)=5
g(n)=5
h(n)=7
g(n)=6
h(n)=6
g(n)=6
h(n)=6
g(n)=6
h(n)=4
g(n)=6
h(n)=6
g(n)=6
h(n)=4
g(n)=6
h(n)=6
g(n)=7
h(n)=5
g(n)=7
h(n)=5
g(n)=7
h(n)=5
g(n)=7
h(n)=3



⇒ Whenever we meet the situation which node with $f(n) = g(n) + h(n) > 10$, we cut the path off.

Then we can get the result with 10 steps to the target. #

2

When working on a bi-directional search with limited memory, the search tree and the hash table will share the same limited memory. Hence, before the search, we should allocate the memory for the search routine to maintain the search tree. Besides, we also also need to allocate the memory for the hash table for storing intermediate results.

Then we will use the iterative deepening DFS from goal and store all the leaves and their shortest paths to the goal in the hash table. Before searching a new level, we need to calculate the required memory for storing all the entries of this level. When the free memory is not enough for starting a new level, we need to use iterative deepening DFS from root. When it encounters the node in the hash table, it gets the shortest path. In other words, we need to try to fill the hash table with the deepest result as much as possible when the free memory is not enough for starting a new level. However, when there is not enough memory for the search routine, we will need to return to the beginning step which allocates the memory and allocate more memory for it, then we will conduct the further steps, such as DFS.

3

As mentioned in class, for additive pattern databases, patterns of different sets must be disjoint sets.

⇒ When using the patterns, the size will correspondingly be:

for $(1\sim7) \Rightarrow \frac{16!}{9!}$ for $(8\sim15) = \frac{16!}{8!}$

Hence when using the two patterns the size will be $\frac{16!}{9!} + \frac{16!}{8!}$

For pattern $(1\sim7)$, there are $\frac{16!}{9!}$ possible states without considering spaces and $8\sim15$.

For pattern $(8\sim15)$, there are $\frac{16!}{8!}$ possible states without considering spaces and $1\sim7$.

When using two patterns (space, $1\sim7$) & $(8\sim15)$ the size is $\frac{16!}{8!} + \frac{16!}{8!}$ #