Homework 1

by Mher Movsisyan

Problem 1

Initial state: 3 cats, 2 dogs, boat - 0 cats, 0 dogs

State space upper limit: Two places where the boat can be, dogs can be 2-0,1-1,0-2, and cats can be 3-0,2-1,1-2,0-3 so upper limit is 2^1 + 3^2 + 4^2 = 288

Actions(initial_state) = { 1 cat goes (pointless) 1 dog goes (pointless) 1 cat and 1 dog go 2 dogs go (termination in 2 steps) 2 cats go (termination right after this step) } every action moves the boat to the opposite bank branching factor of at most 5

State representation: `cdbdc` where `c` is the number of cats `d` is the number of dogs `b` is 1 if the boat is on the left bank, 0 otherwise

Goal state: `00123`

Action representation: `cd`

Transition model:

```
take_action(state, action)
    if state.b == 0:
        state.b = 1
        state.left_c = state.left_c - action.c
        state.left_d = state.left_d - action.d
        state.right_c = state.right_c + action.c
        state.right_d = state.right_d + action.d
    else:
        state.b = 0
        state.right_c = state.right_c - action.c
        state.right_d = state.right_d - action.d
        state.left_c = state.left_c + action.c
        state.left_d = state.left_d + action.d
```

c. It is better to use a graph search algorithm since the number of nodes aren't that many (memory won't be a problem) and there are loops in the graph.

Problem 2

Password: `***`

$$\in \{X, Y, Z\}$$

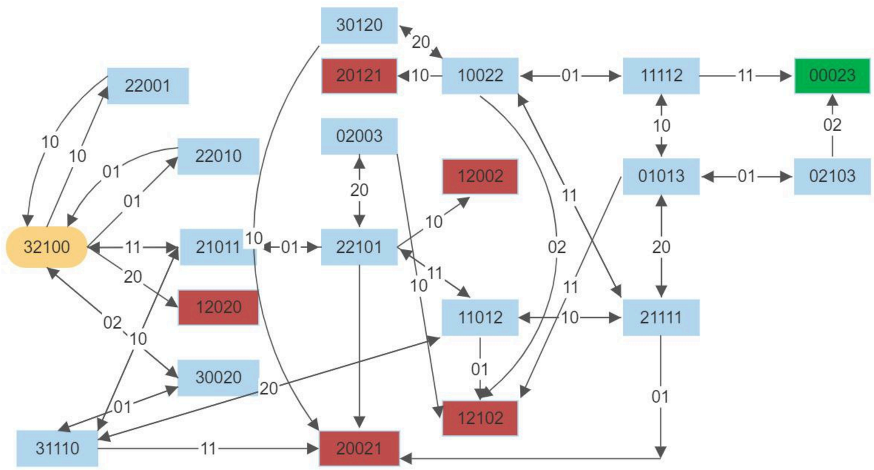


Figure 1: Transition map

Initial state = ‘`action = append`*‘
goal state =

`password in {`ZYX`, `YXY`, `XYY`, `ZY`, `ZZZ`, `YZZ`, `XXX`}`

branching factor = 3
shallowest solution depth = 2
upper limit on state space = 3 _ 3 _ 3 = 27

We will use tree search since we can’t end up exploring the same state twice.

DFS:

1. `X`
2. `XX`
3. `XXX`
   Since we are going alphabetically we will reach the `XXX` goal state without exploring other branches.

BFS:

1. `X`
2. `Y`
3. `Z`
   done with layer one, onto layer two
4. `XX`
5. `XY`
6. `XZ`

7. `YX`
8. `YY`
9. `YZ`
10. `ZZ`
11. `ZY`
    reached shallowest goal state `ZY`

UCS:

0. Expanded `""`, got {`X` (10), `Y` (2), `Z` (1)}
1. `Z` (1), frontier = {`X` (10), `Y` (2), `ZZ` (2), `ZY` (3), `ZX` (11)}
2. `Y` (eventho this state and `ZZ` have the same cost of 2, we respect the alphabet), frontier = {`X` (10), `ZZ` (2), `ZY` (3), `ZX` (11), `YZ` (3), `YY` (4), `YX` (10)}
3. `ZZ` (2), frontier = {`X` (10), `ZY` (3), `ZX` (11), `YZ` (3), `YY` (4), `YX` (10), `ZZZ` (3), `ZZY` (4), `ZZX` (12)}
4. `YZ` (3), frontier = {`X` (10), `ZY` (3), `ZX` (11), `YY` (4), `YX` (10), `ZZZ` (3), `ZZY` (4), `ZZX` (12), `YZZ` (4), `YZY` (5), `YZX` (13)}
5. `ZY` (3)
   Reached shallowest goal state `ZY`

If it weren't for the alphabetical tie breaker, we might have reached the `ZZZ` goal state

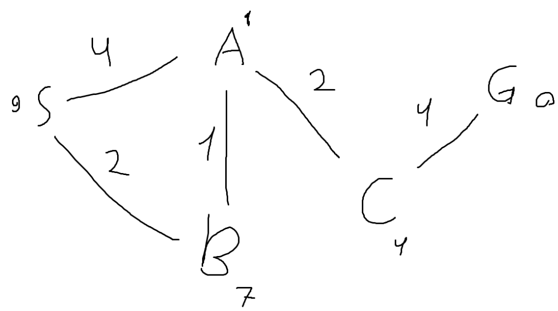
Problem 3

a.

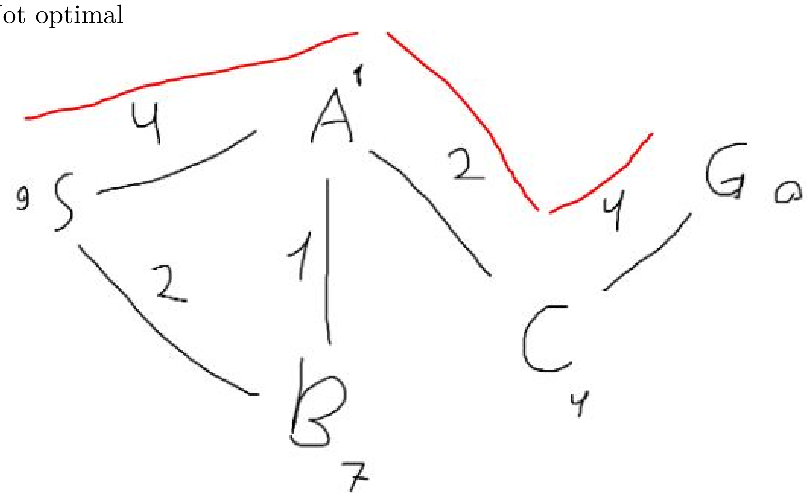
- M[241 = 0 + 241]
- L[314 = 70 + 244], D[317 = 75 + 242]
- T[510 = 181 + 329], D[317 = 75 + 242]
- T[510 = 181 + 329], C[355 = 120 + 75 + 160]
- T[510 = 181 + 329], R[534 = 146 + 195 + 193], P[433 = 138 + 195 + 100]
- T[510 = 181 + 329], R[534 = 146 + 195 + 193], B[404 = 333 + 101 + 0], R[623 = 97 + 333 + 193]
- B (404 path cost)

b. It wouldn't change the outcome but the frontier would be different. More efficient in terms of memory, same time because we only keep one of each letter state.

Problem 4



a. Admissible but not consistent
b. Not optimal



c. Not optimal

- S[0 + 9 = 9]
- A[4 + 1 = 5], B[2 + 7 = 9]
- C[6 + 4 = 10], B[2 + 7 = 9], new B is more costly thus don't replace
- C[6 + 4 = 10], S and A are explored
- G[10 + 0 = 10], A is explored
- G tested and found solution