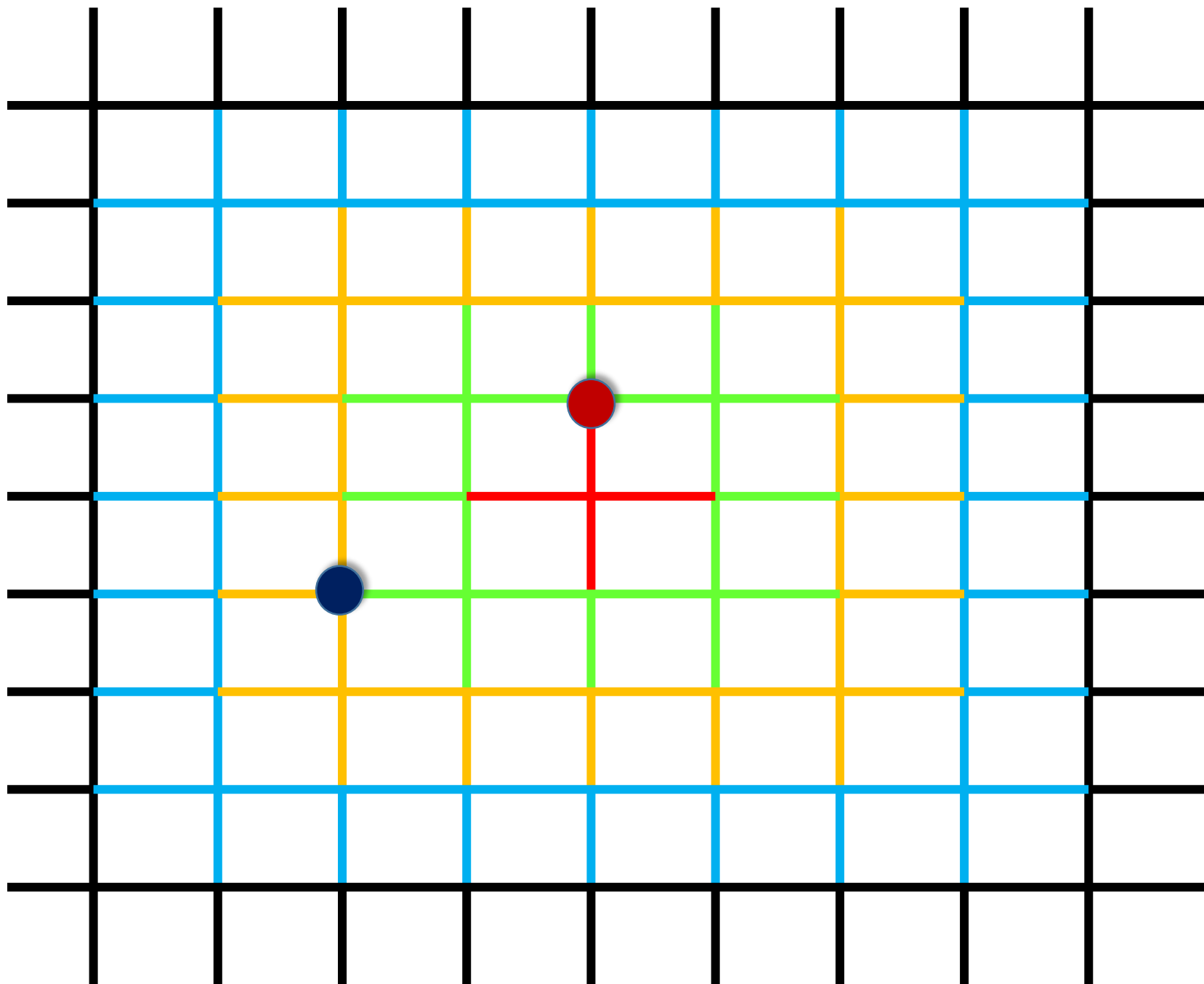


Search Algorithms

Uniform Cost Search

```
s = start
frontier = heap ({node})
explored = {}
while not empty(frontier):
    node = frontier.pop()
    if IS_GOAL(node): return SOLUTION(node)
    explored.add(node)
    for action in node.get_actions():
        child = APPLY(node, action)
        if child not in union(frontier, explored):
            frontier.add(child)
        else if child in frontier:
            frontier.decide_and_replace(child)
```



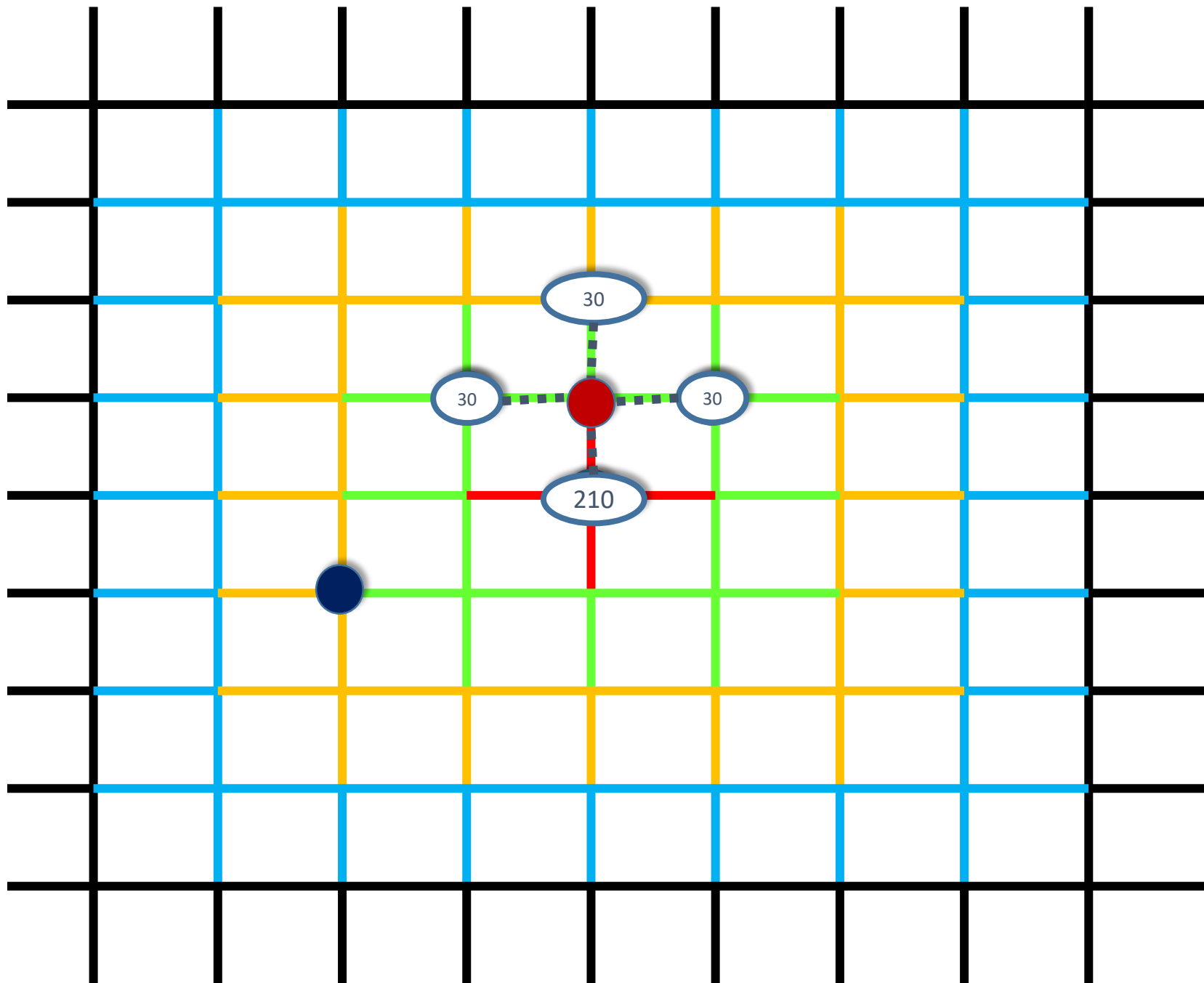
Path Cost :

Path Cost :

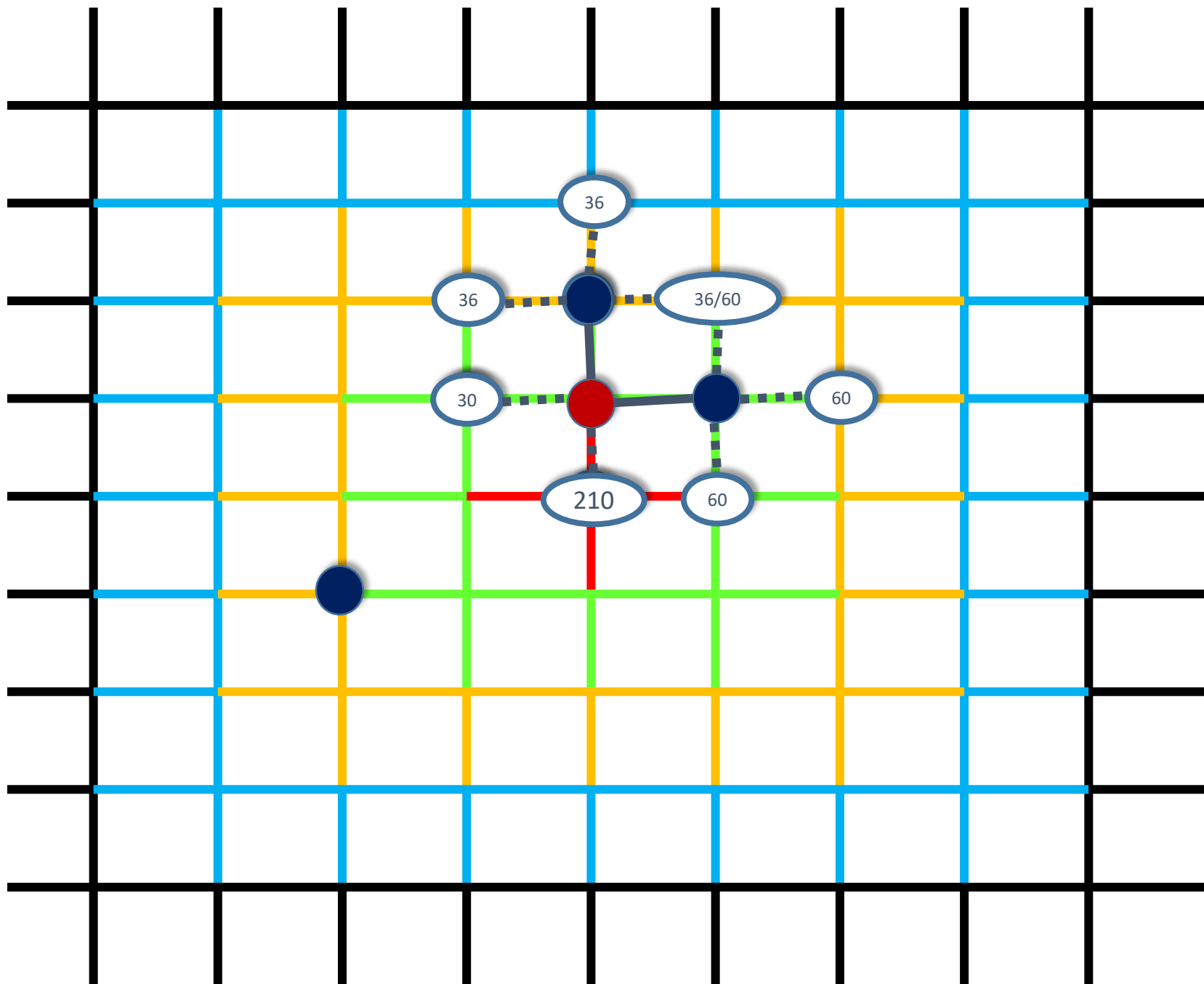
Path Cost :

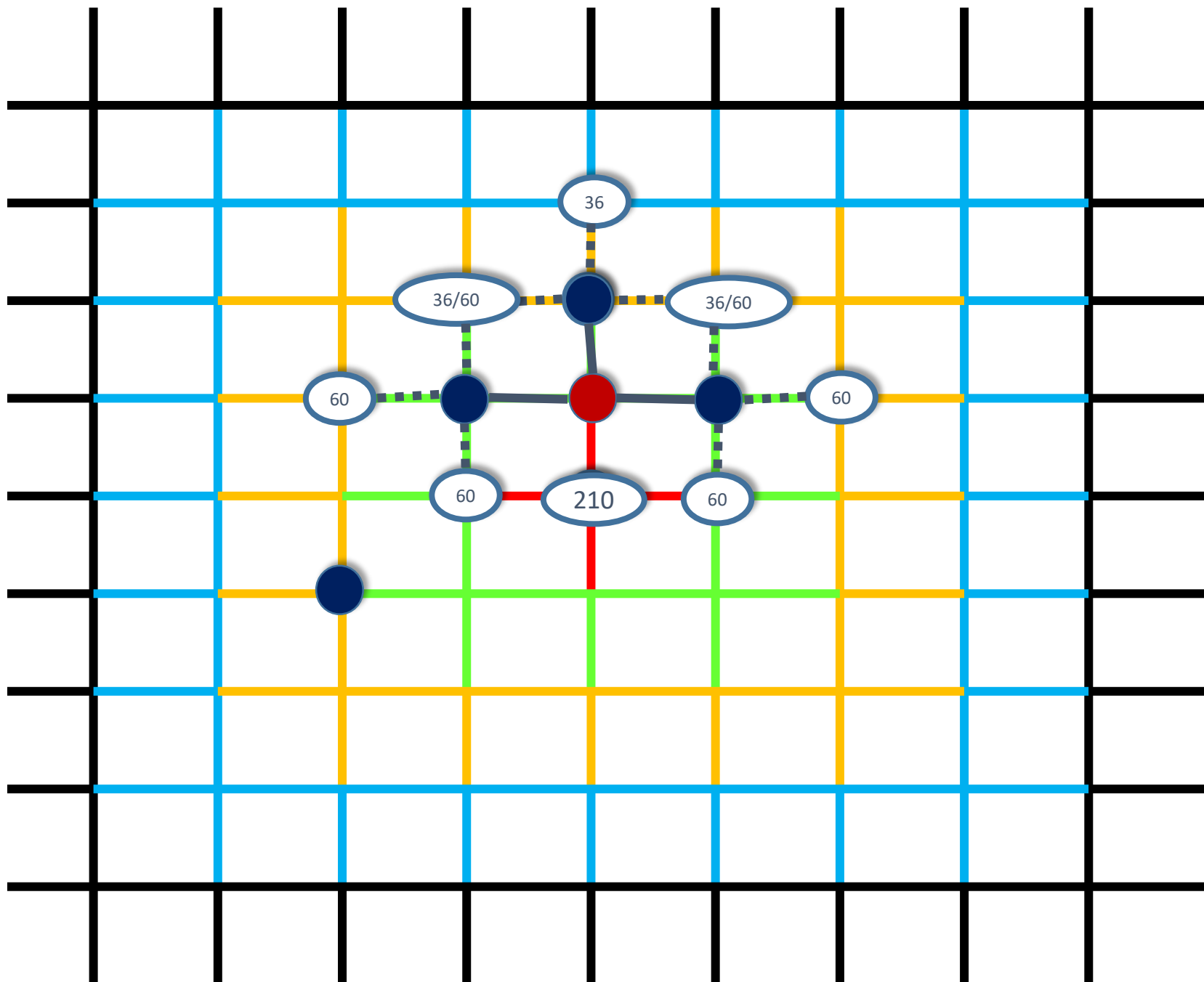
Path Cost :

Path Cost :



Path Cost :
Path Cost :
Path Cost :
Path Cost :
Path Cost :





Path Cost :

Path Cost :

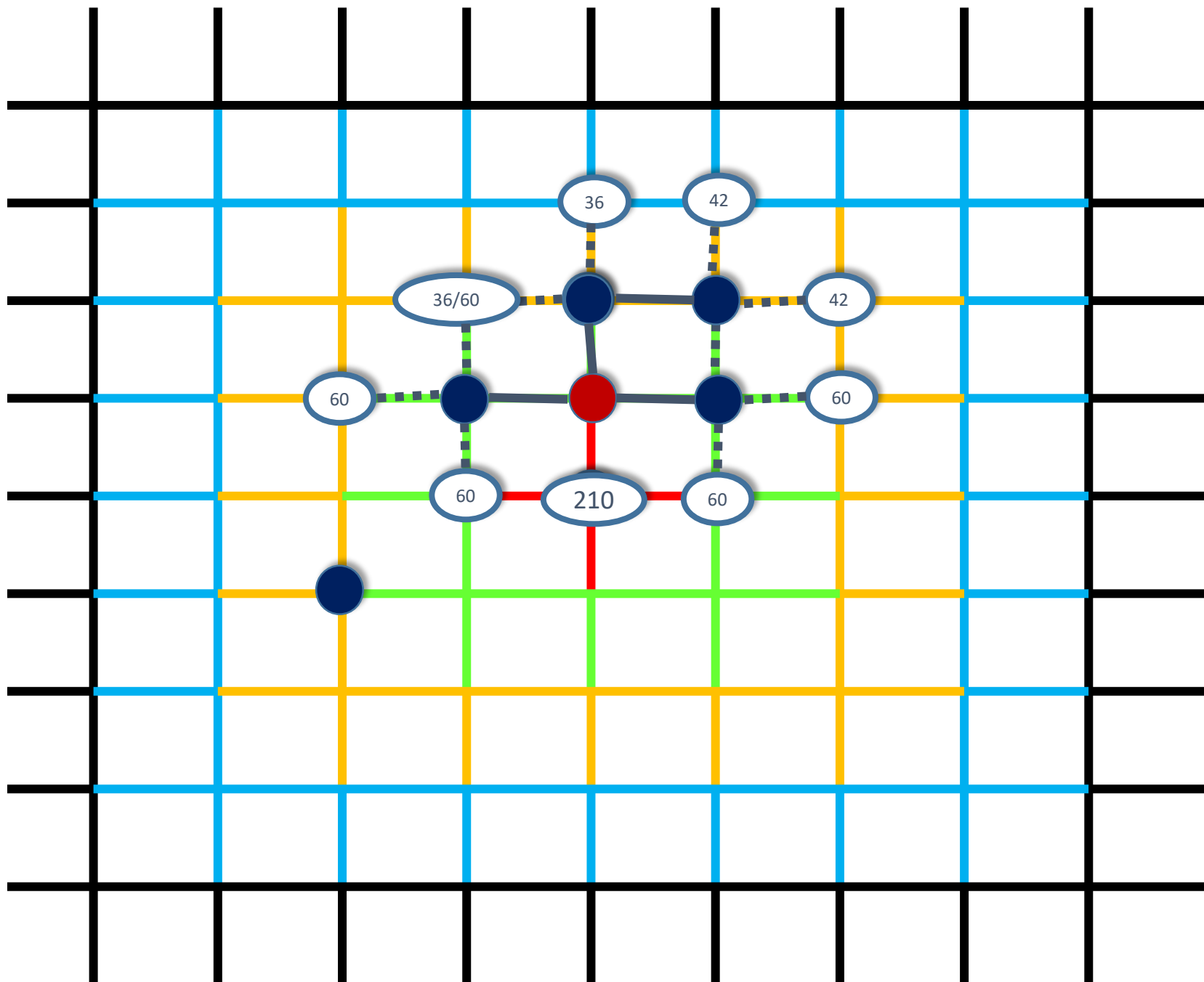
Path Cost :

Path Cost :

Path Cost :

Conventio

Clockwise



Path Cost 1

Path Cost 2

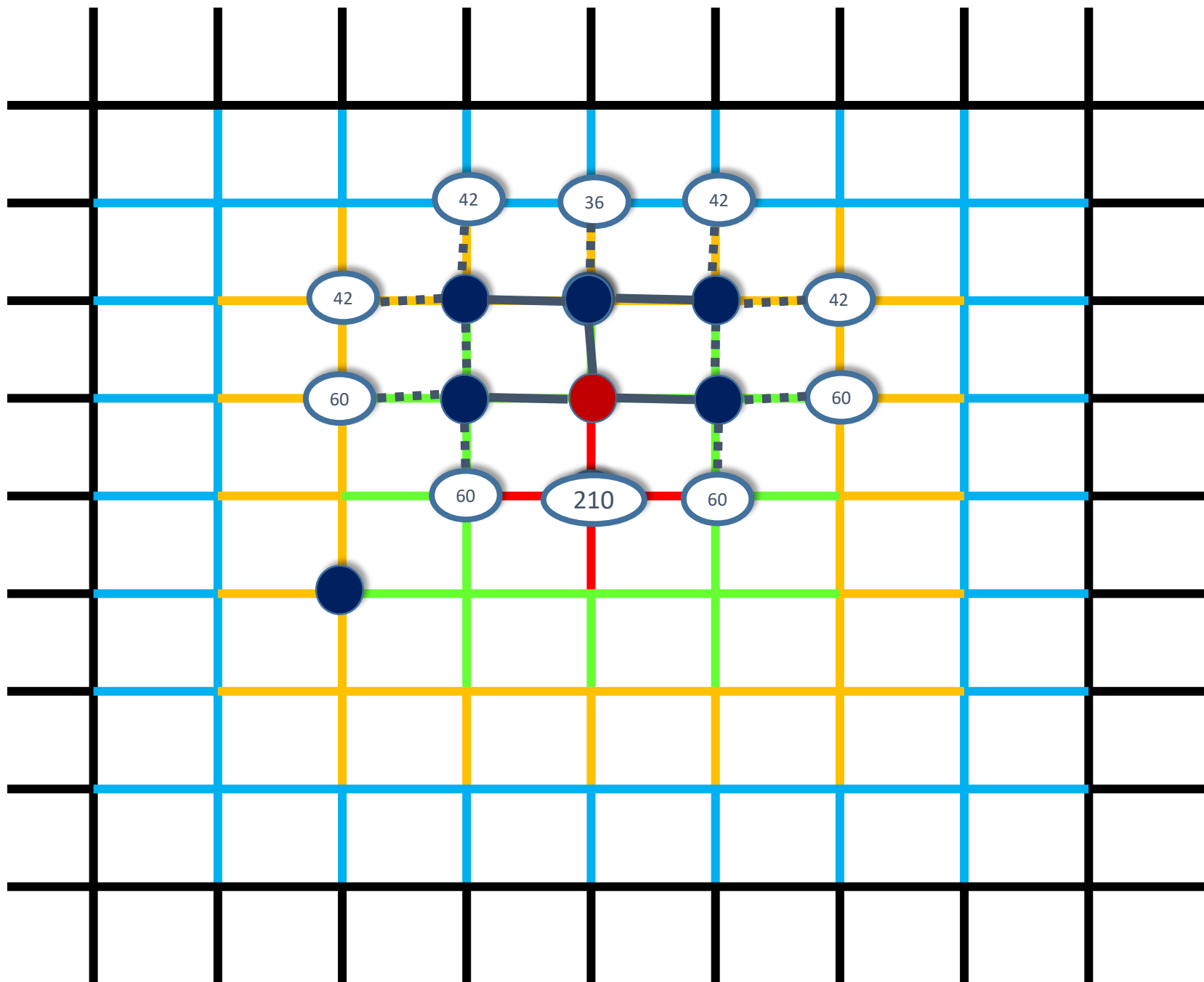
Path Cost 3

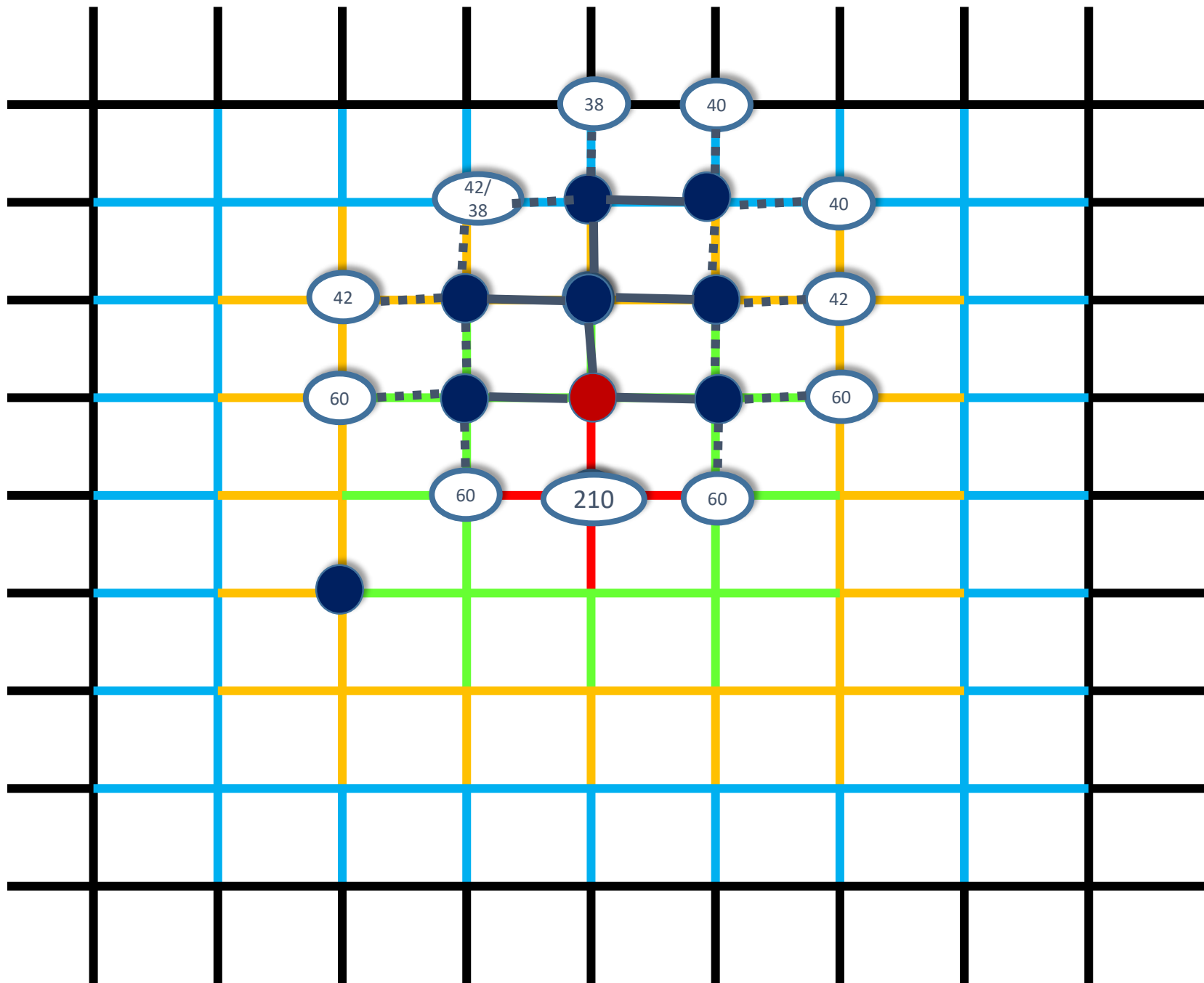
Path Cost 4

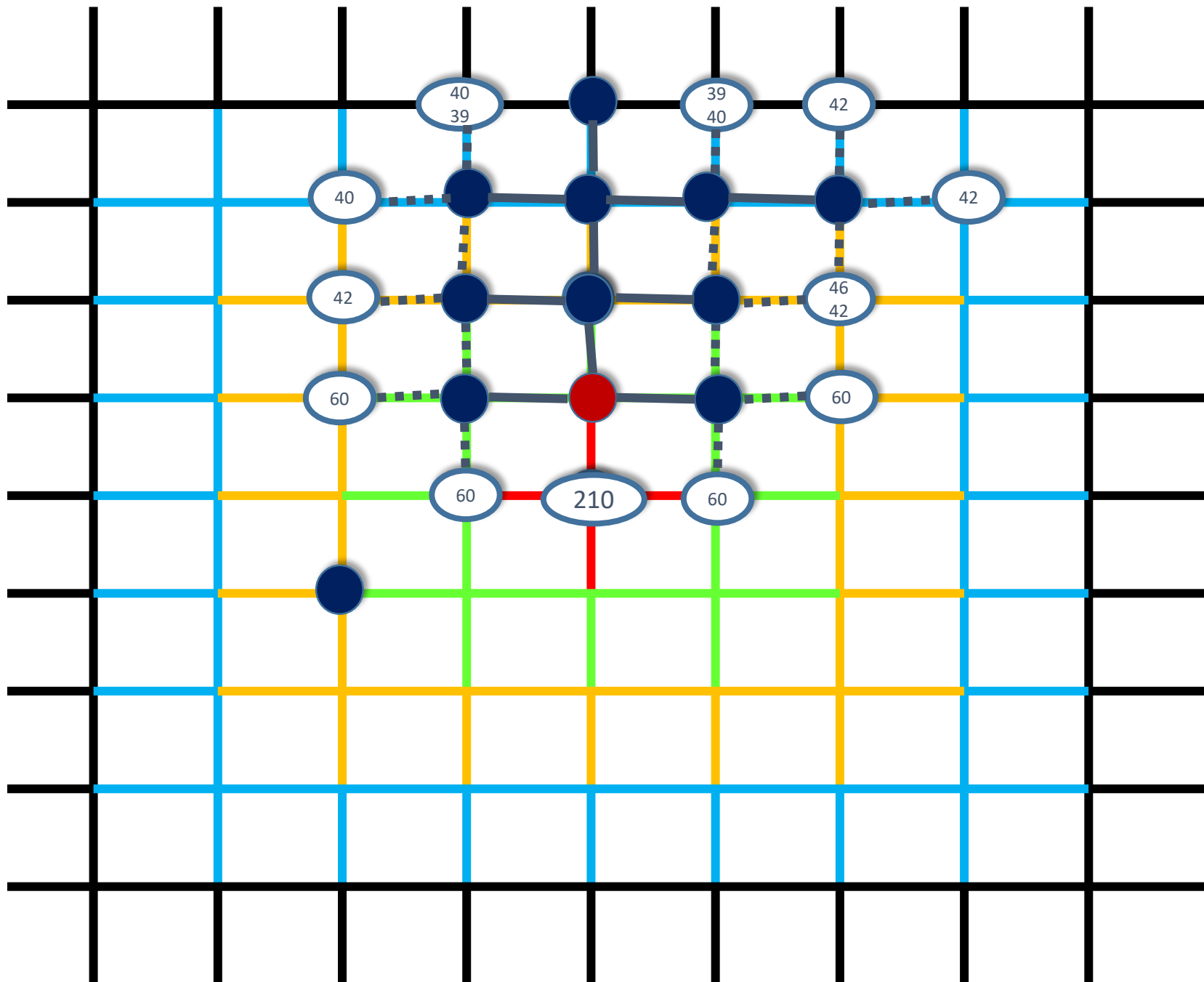
Path Cost 5

Conventio

Clockwise







Path Cost :

Path Cost :

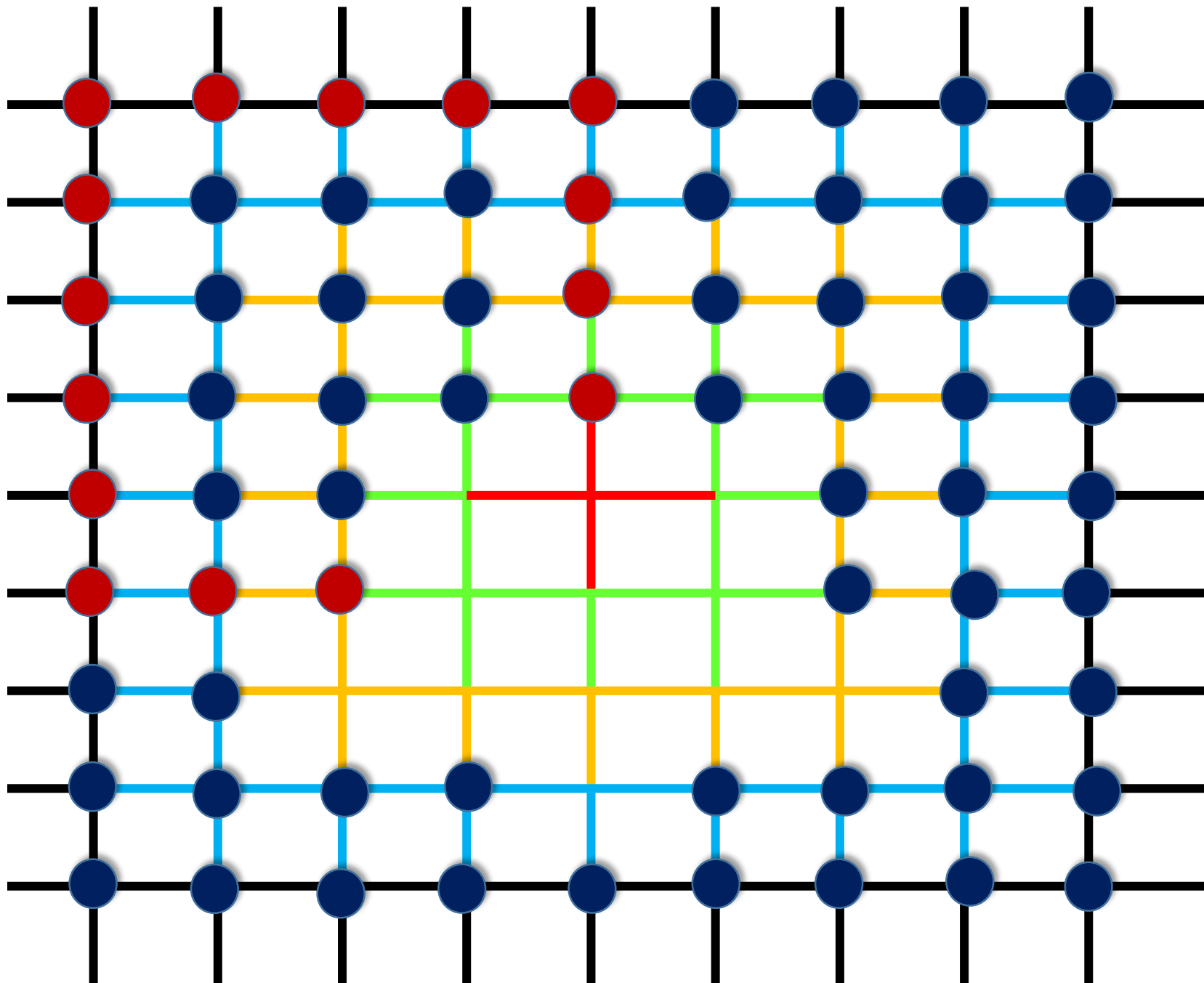
Path Cost :

Path Cost :

Path Cost :

Conventio

Clockwise



Path Cost 1

Path Cost 2

Path Cost 3

Path Cost 4

Path Cost 5

Conventional

Clockwise

Number of

Shortest C

directional Uniform Cost Search

any manner of expanding frontiers is OK

- Alternating both frontiers – good for parallel computing
- Taking the min – good in weighted graphs where hubs have high cost

stopping criterion:

- $\min(\text{forward}) + \min(\text{reverse}) > \text{shortest_path_in_graph}$
- Note: intersection of explored sets, means you check for your stopping criterion when you POP from the queue.

• Search

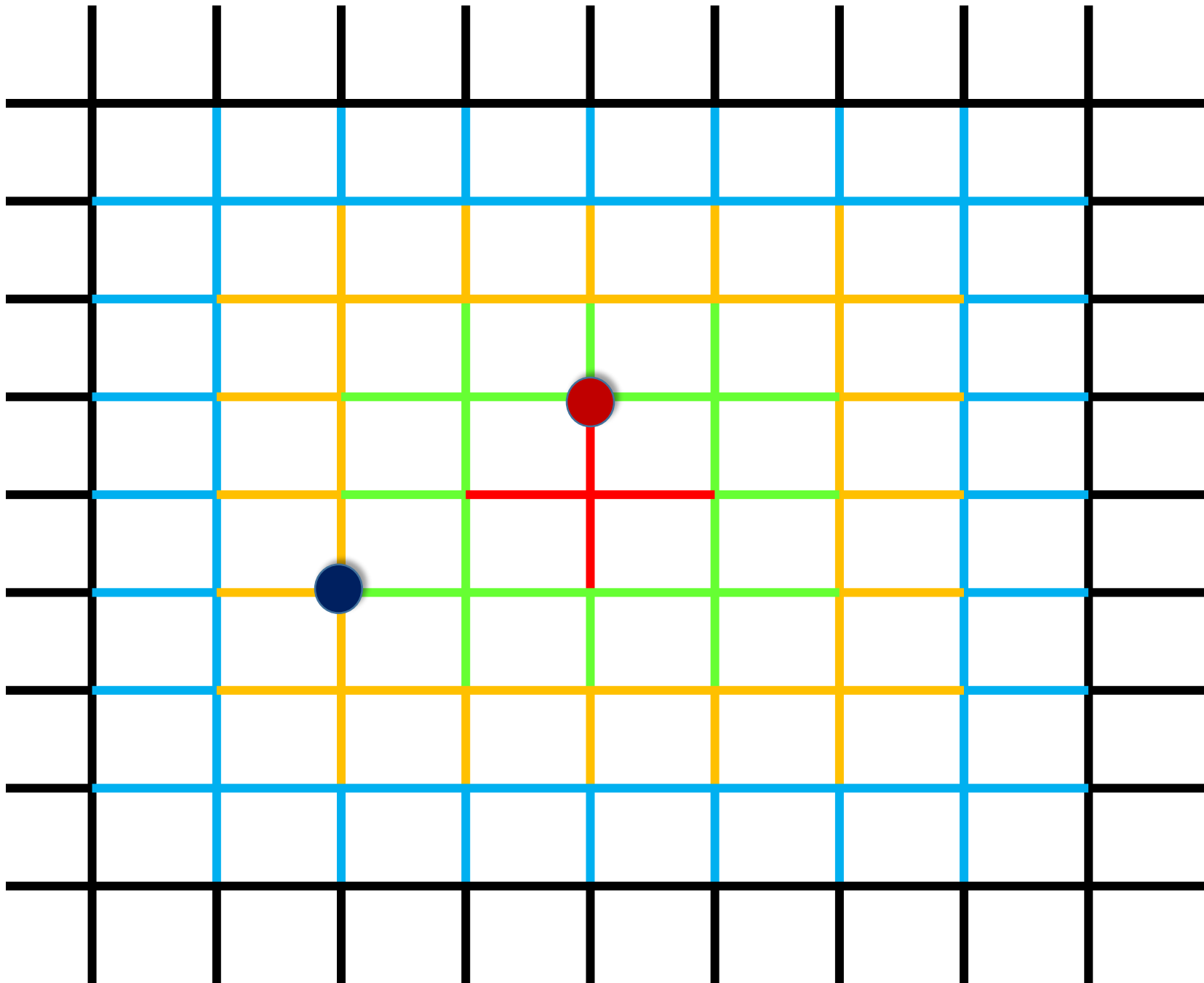
range the heap sort to include a heuristic function

$$f(\text{state}) = h(\text{state}) + g(\text{state})$$

choice of a good heuristic:

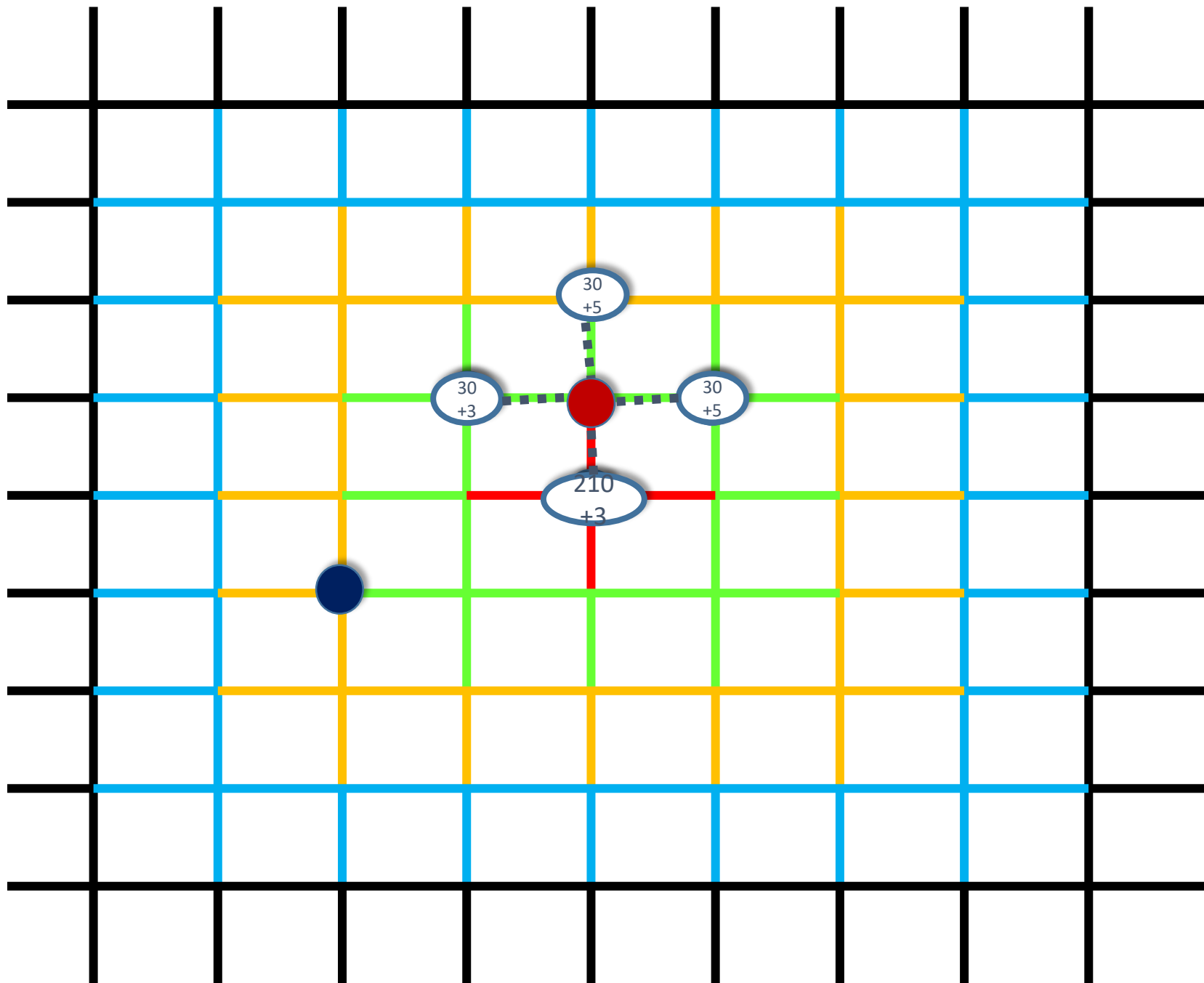
- Admissible: underestimates
- Consistent (strict): monotonic

the better the heuristic, the quicker the search

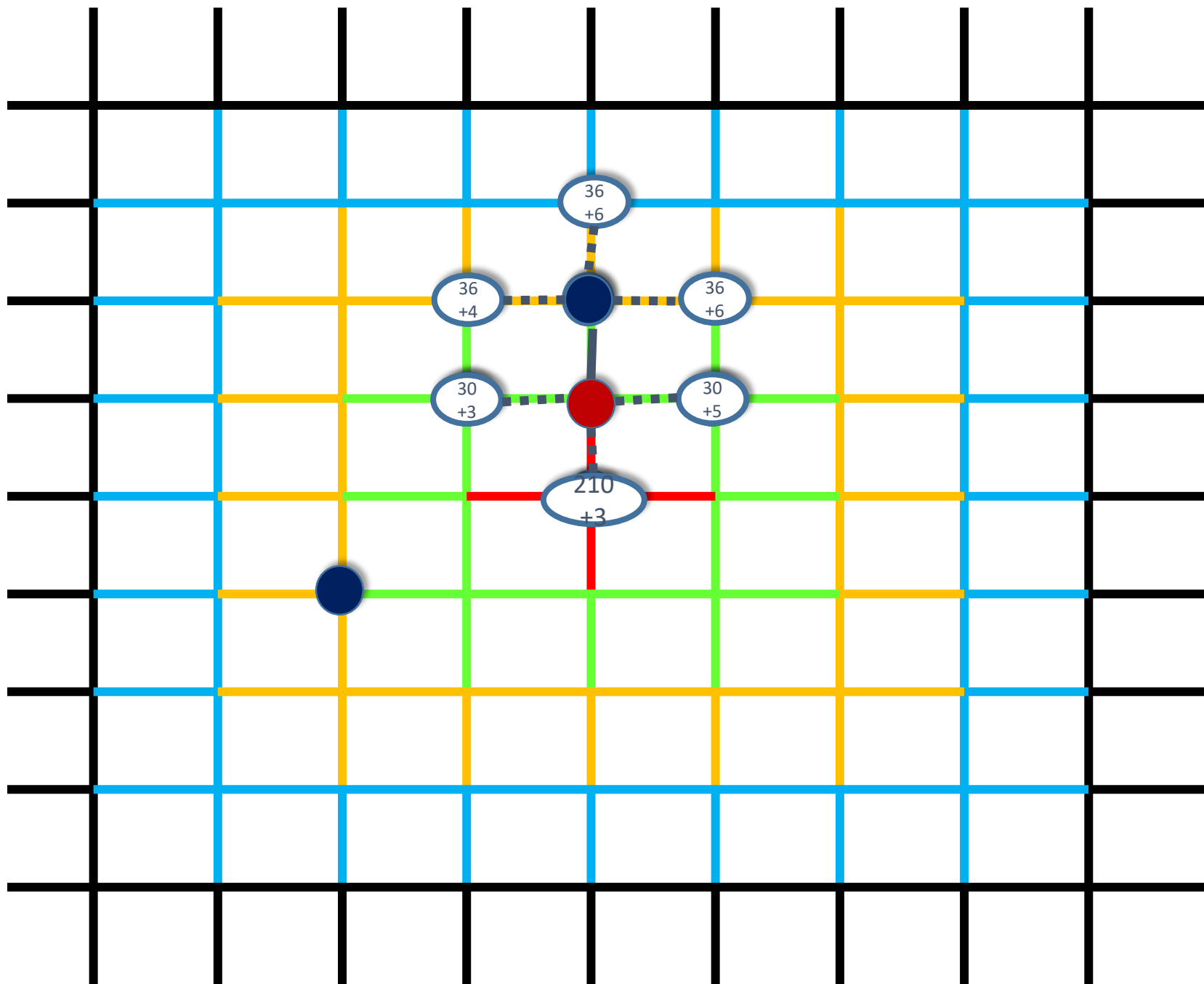


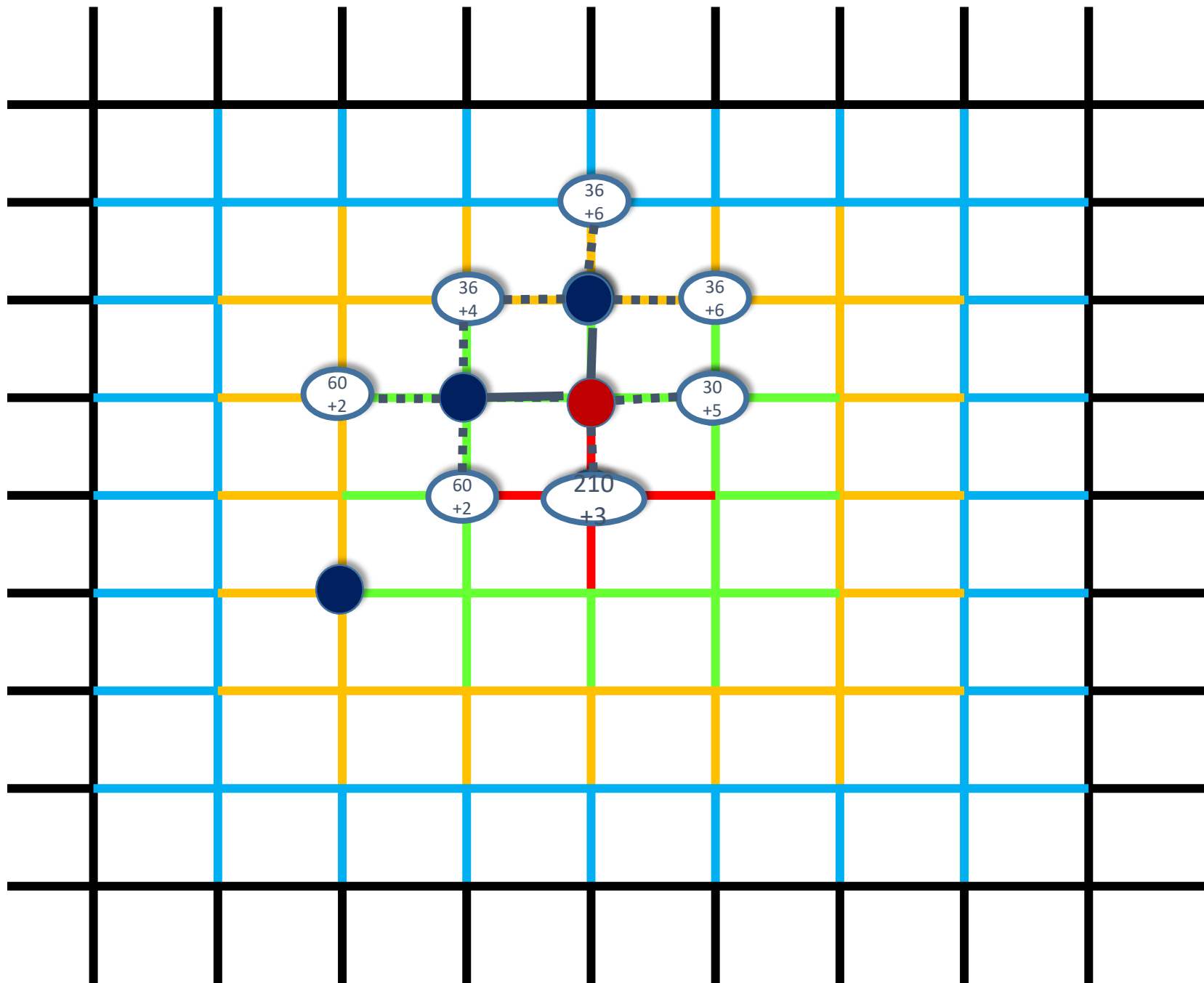
Path Cost :
Path Cost :
Path Cost :
Path Cost :
Path Cost :

Notes:
- Manha
Distanc
Heurist



Path Cost :
Path Cost :
Path Cost :
Path Cost :
Path Cost :





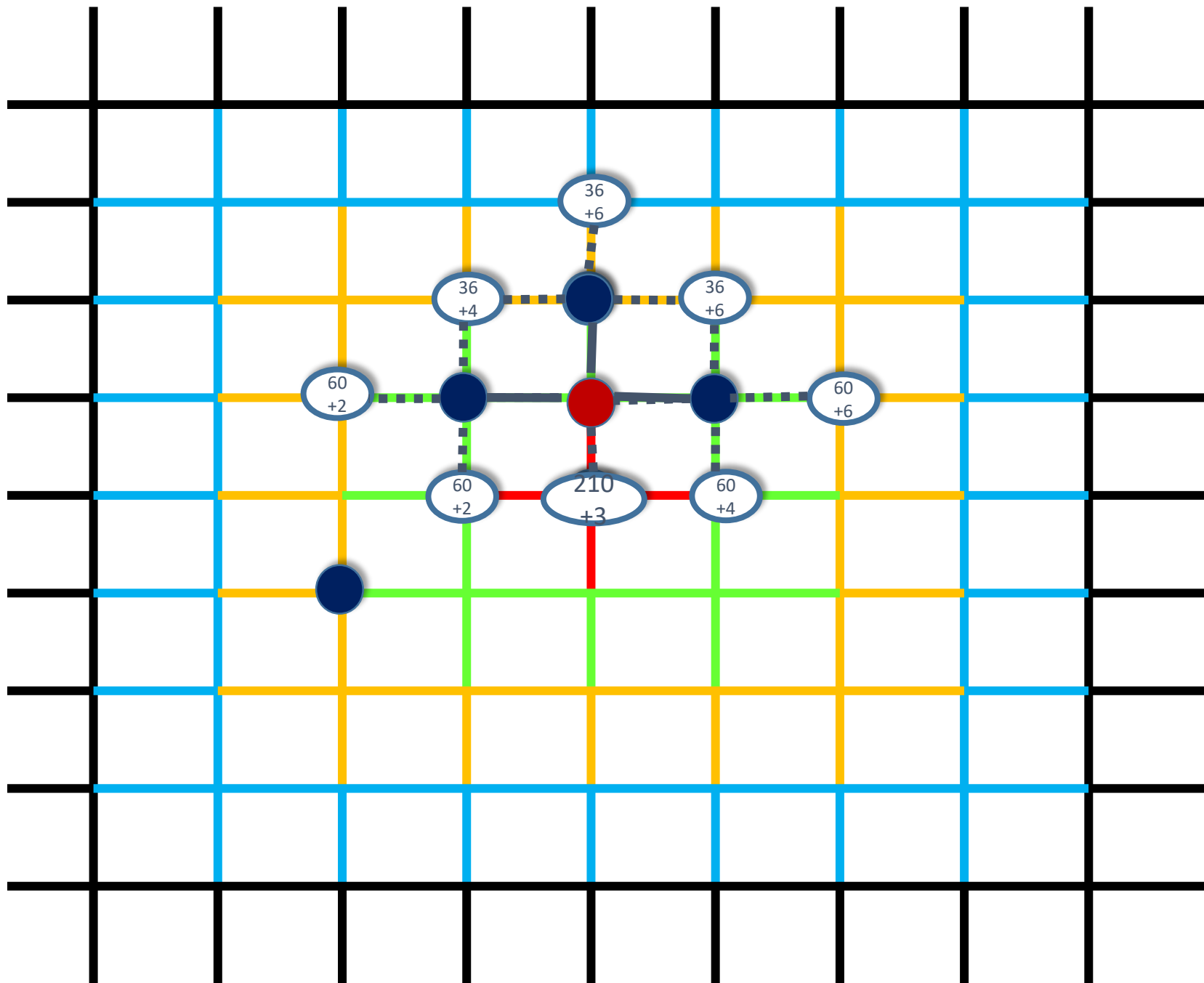
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



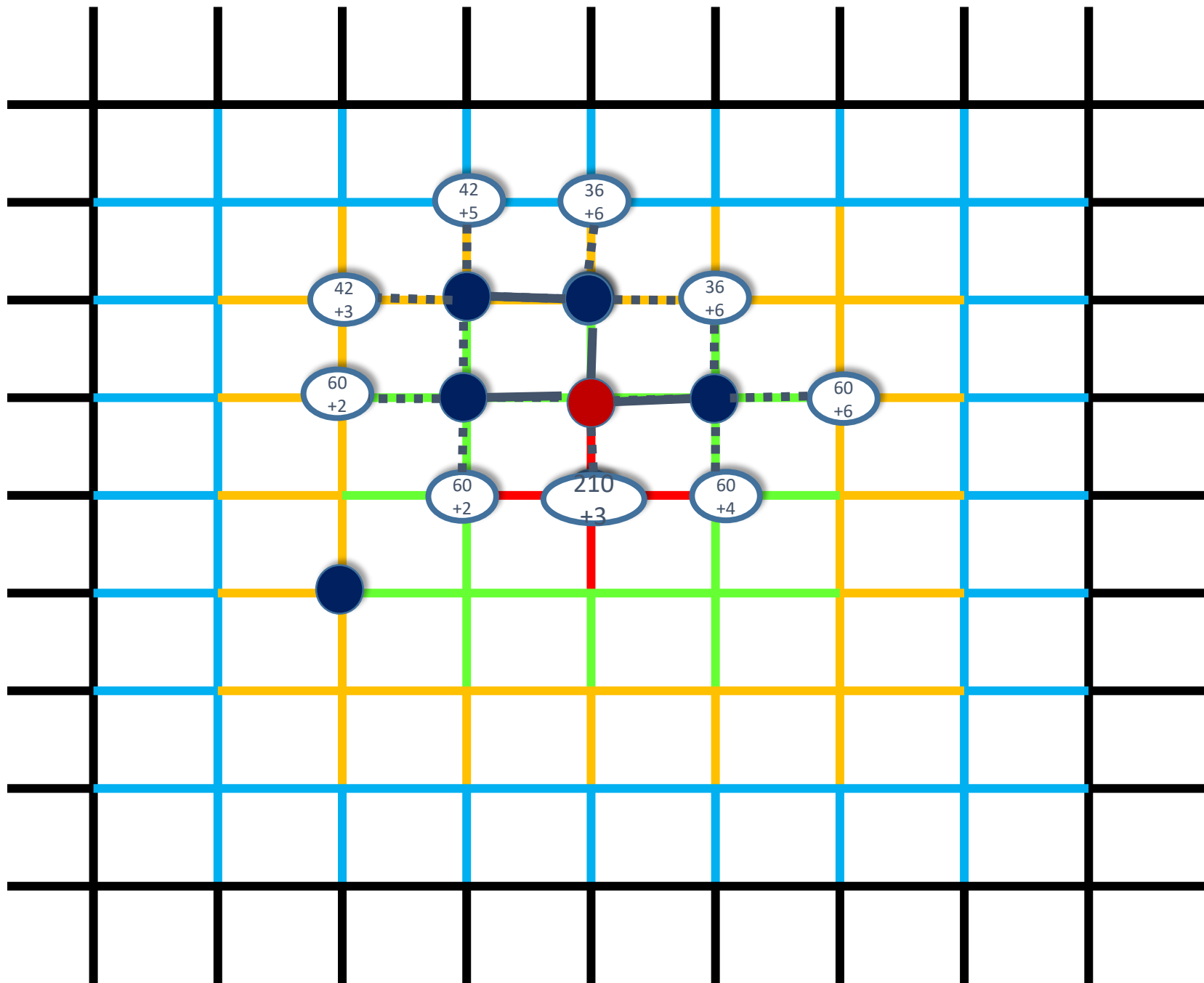
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



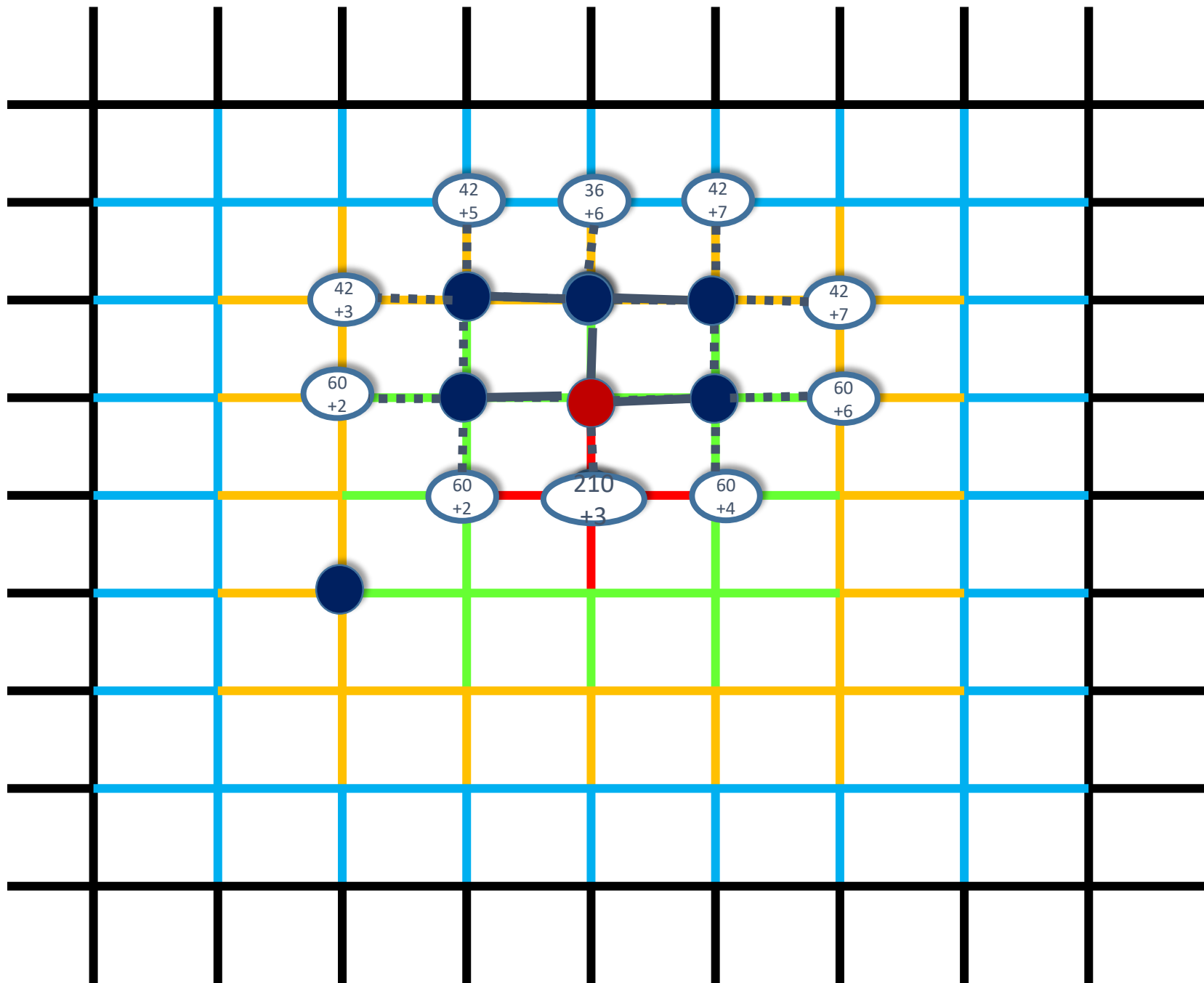
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



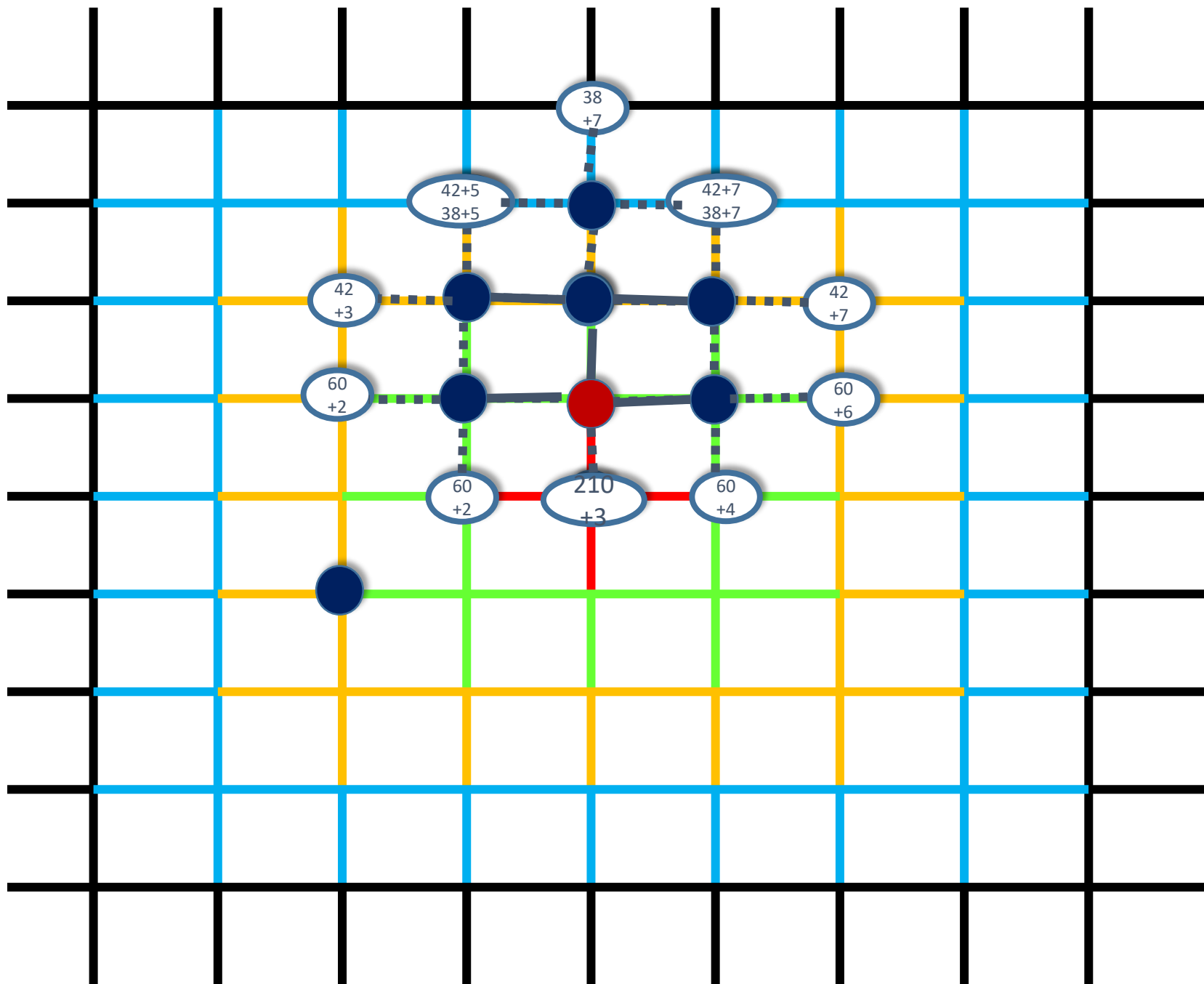
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



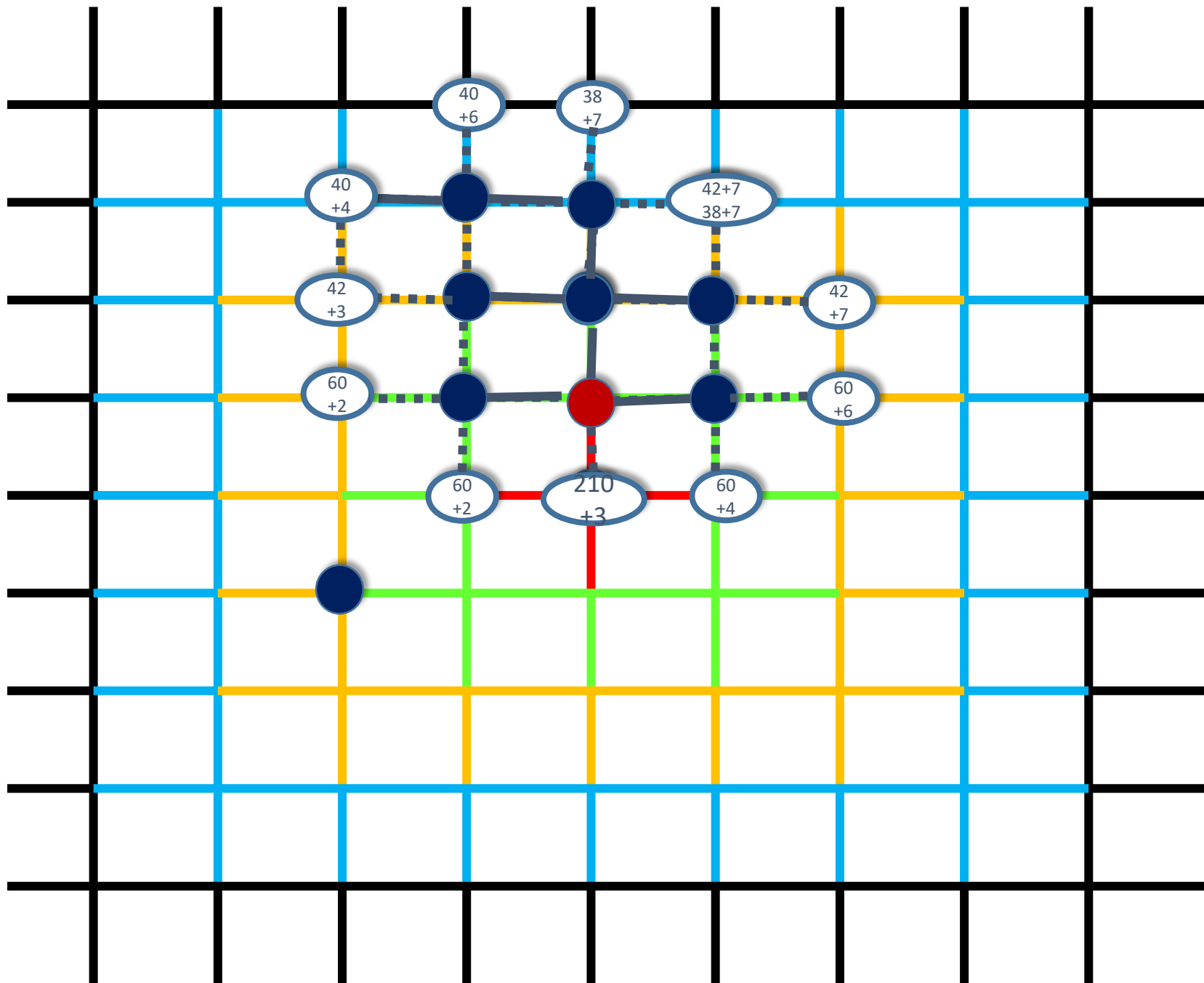
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



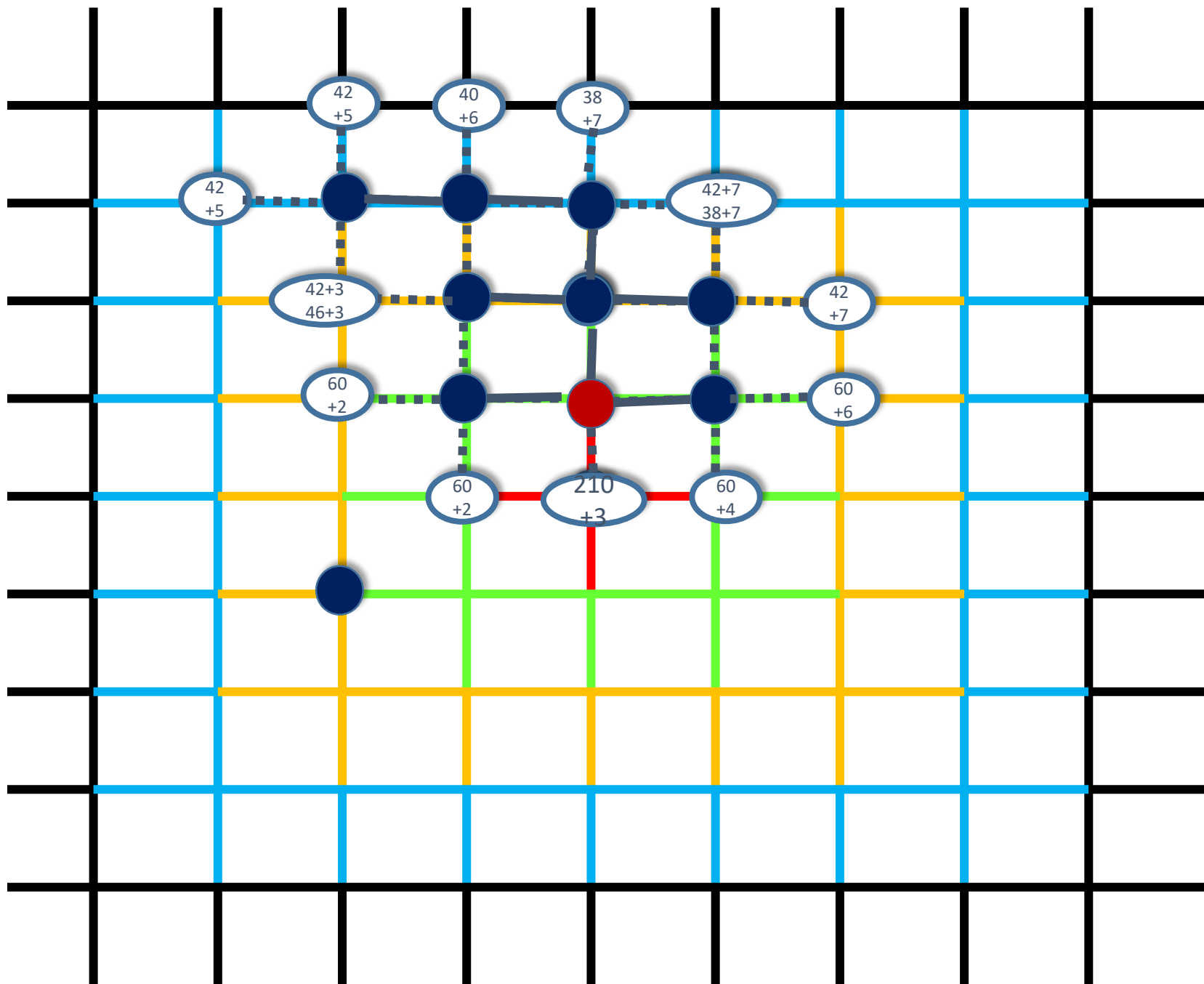
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



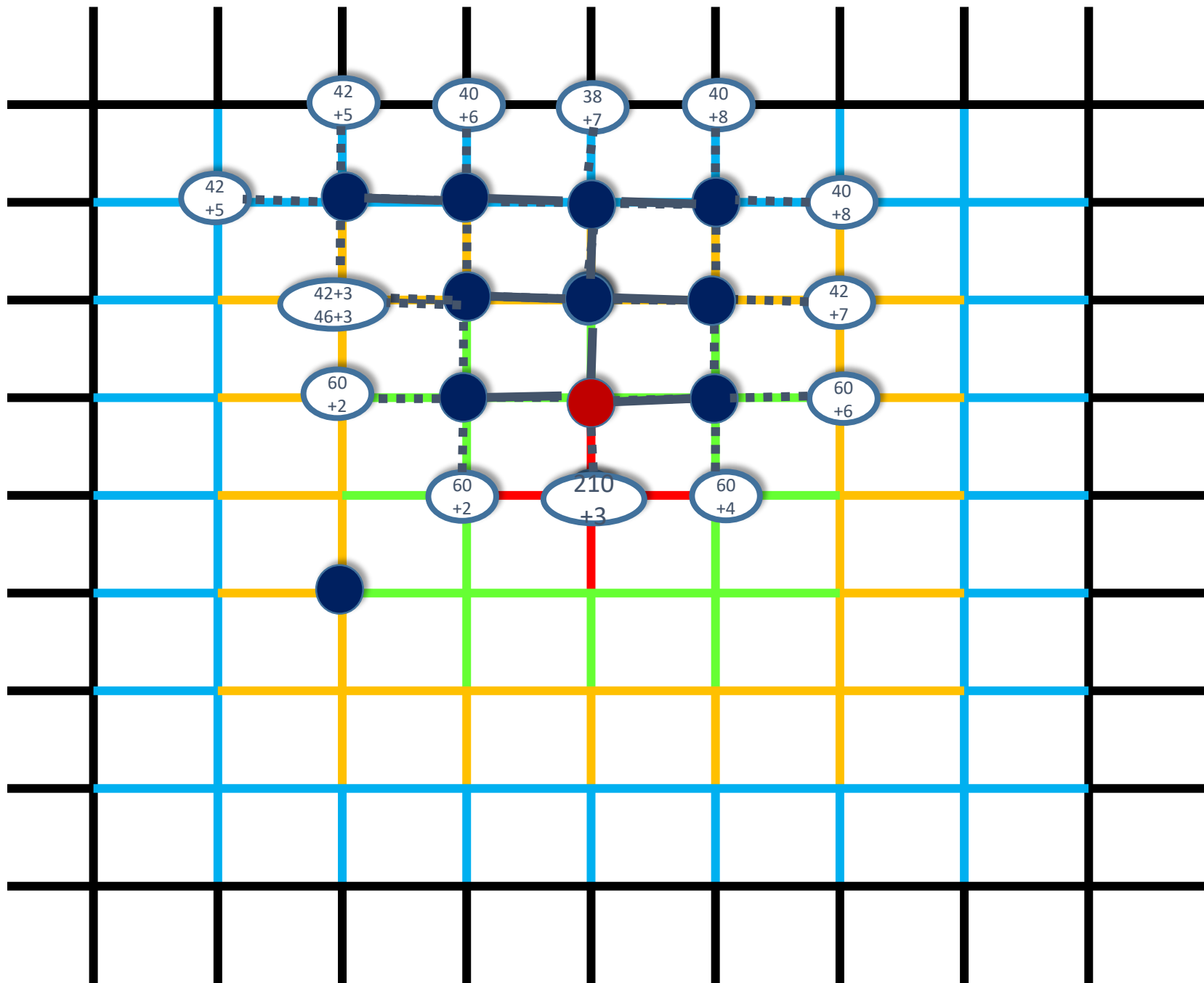
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



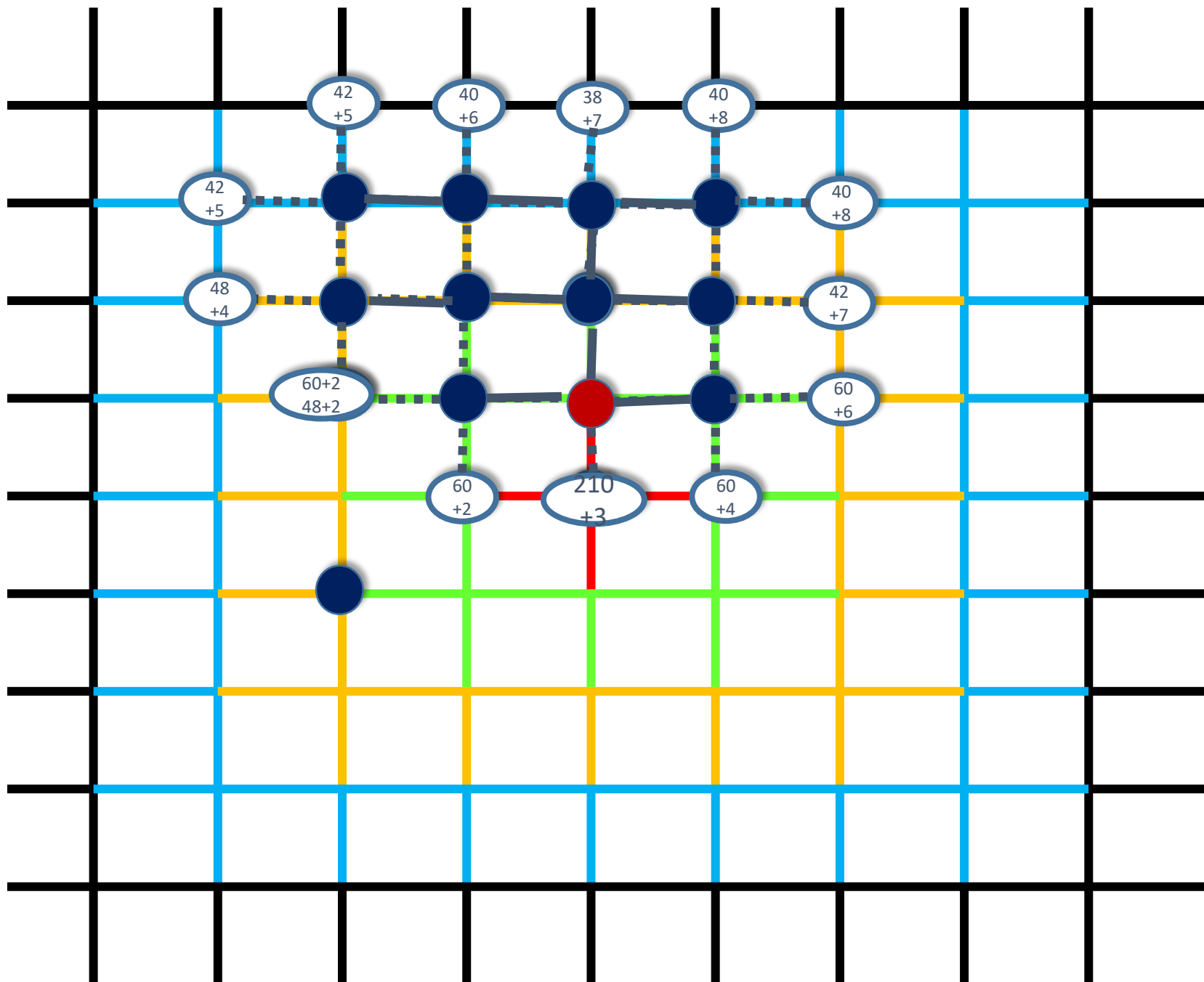
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



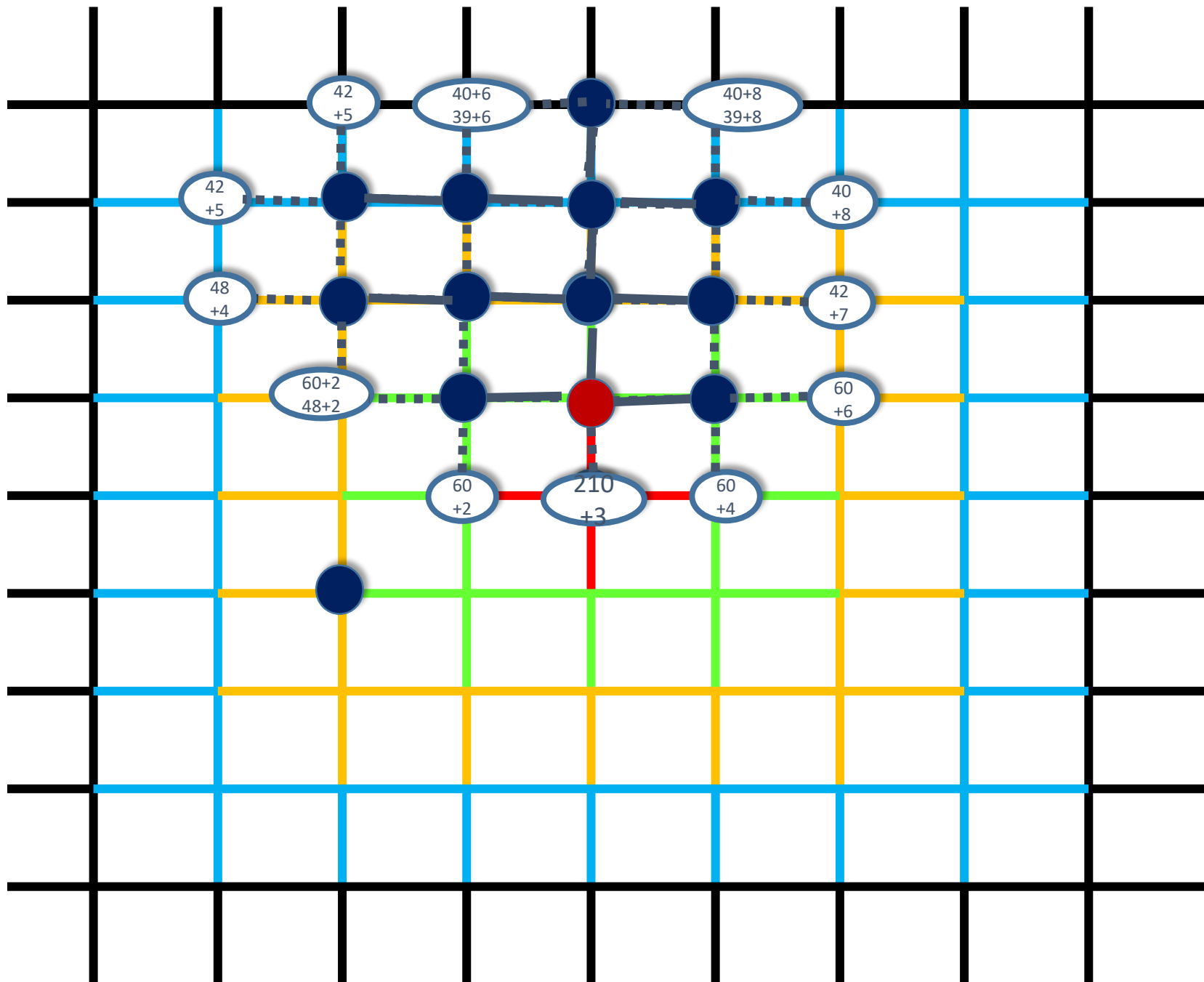
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



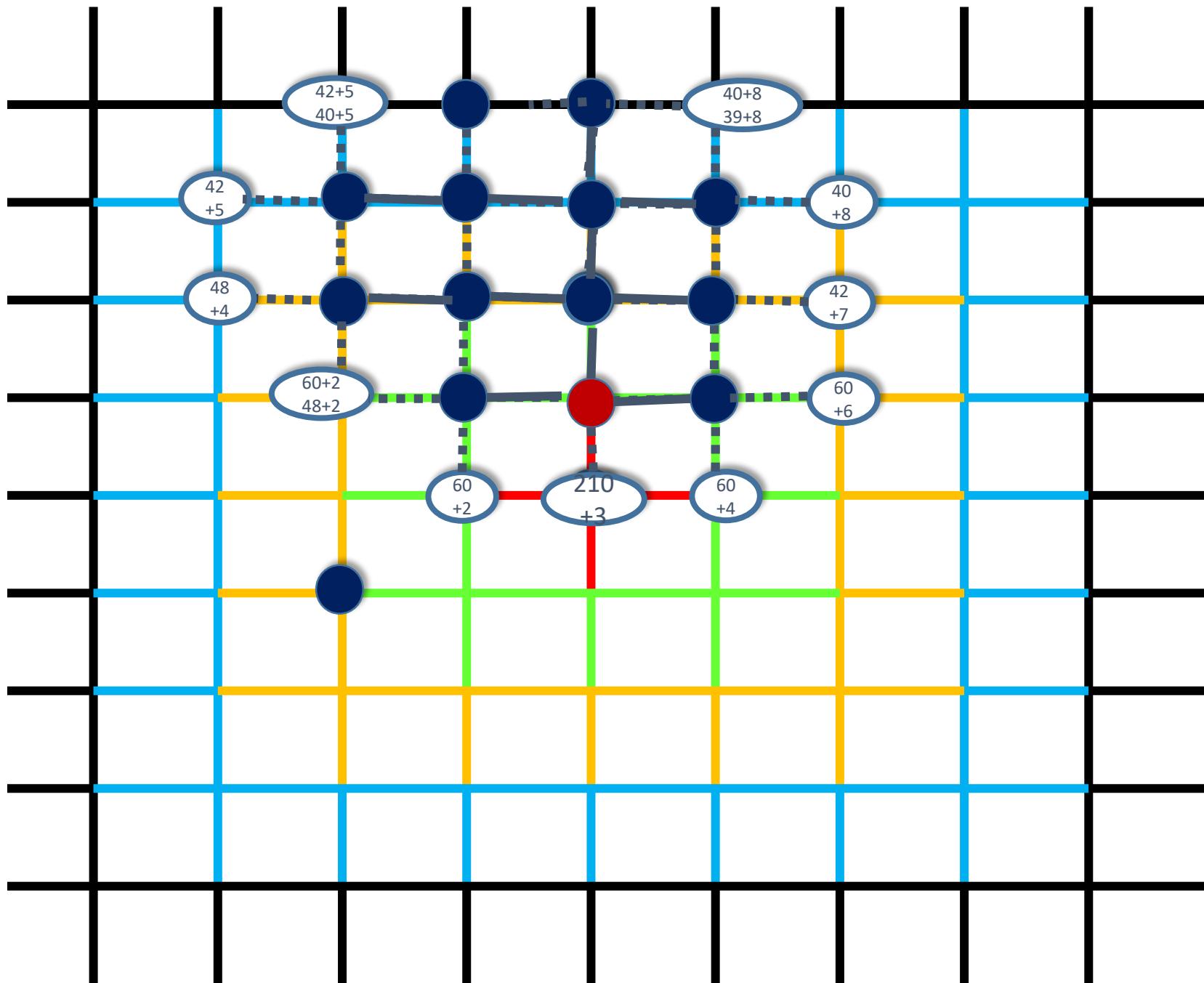
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



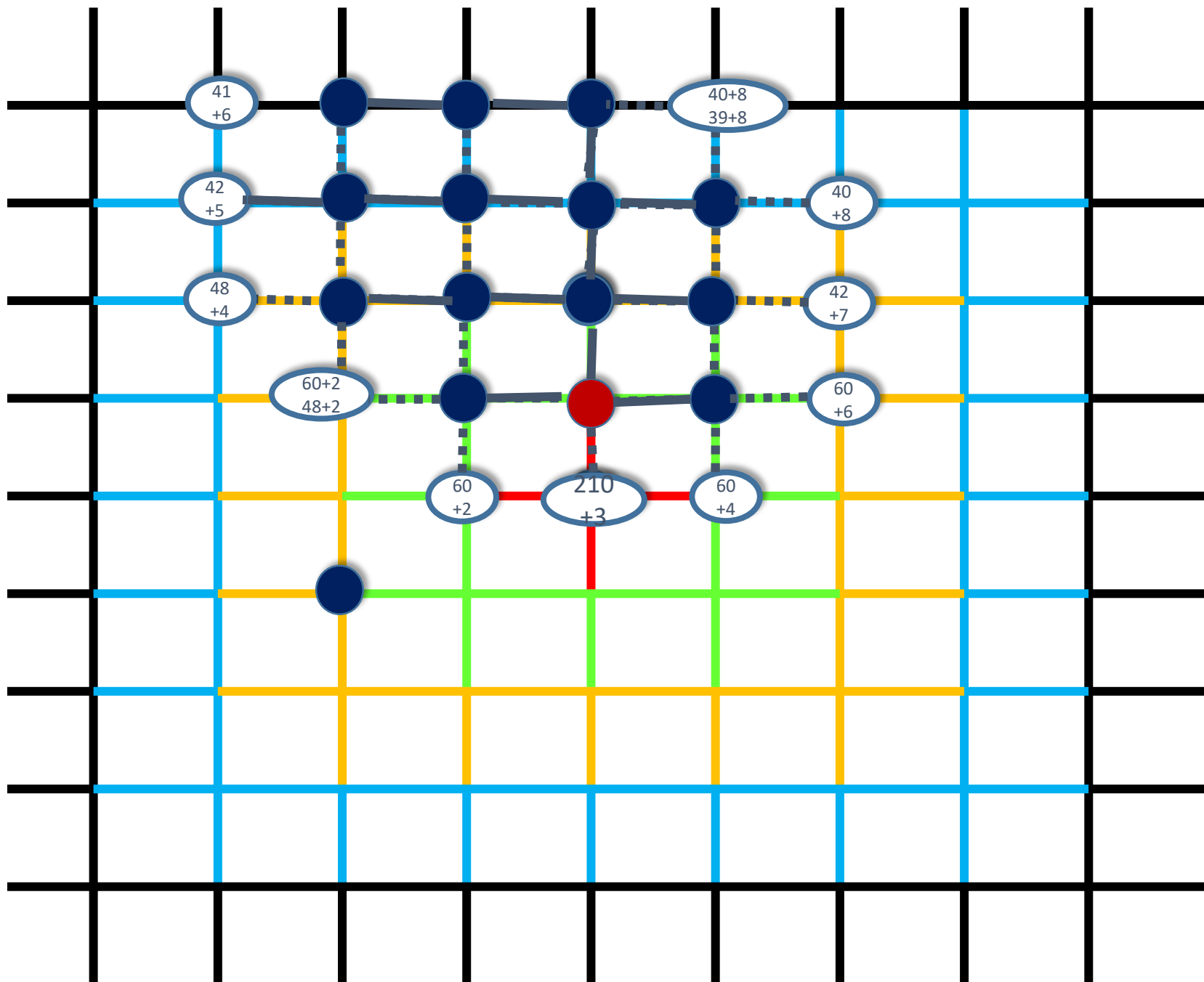
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



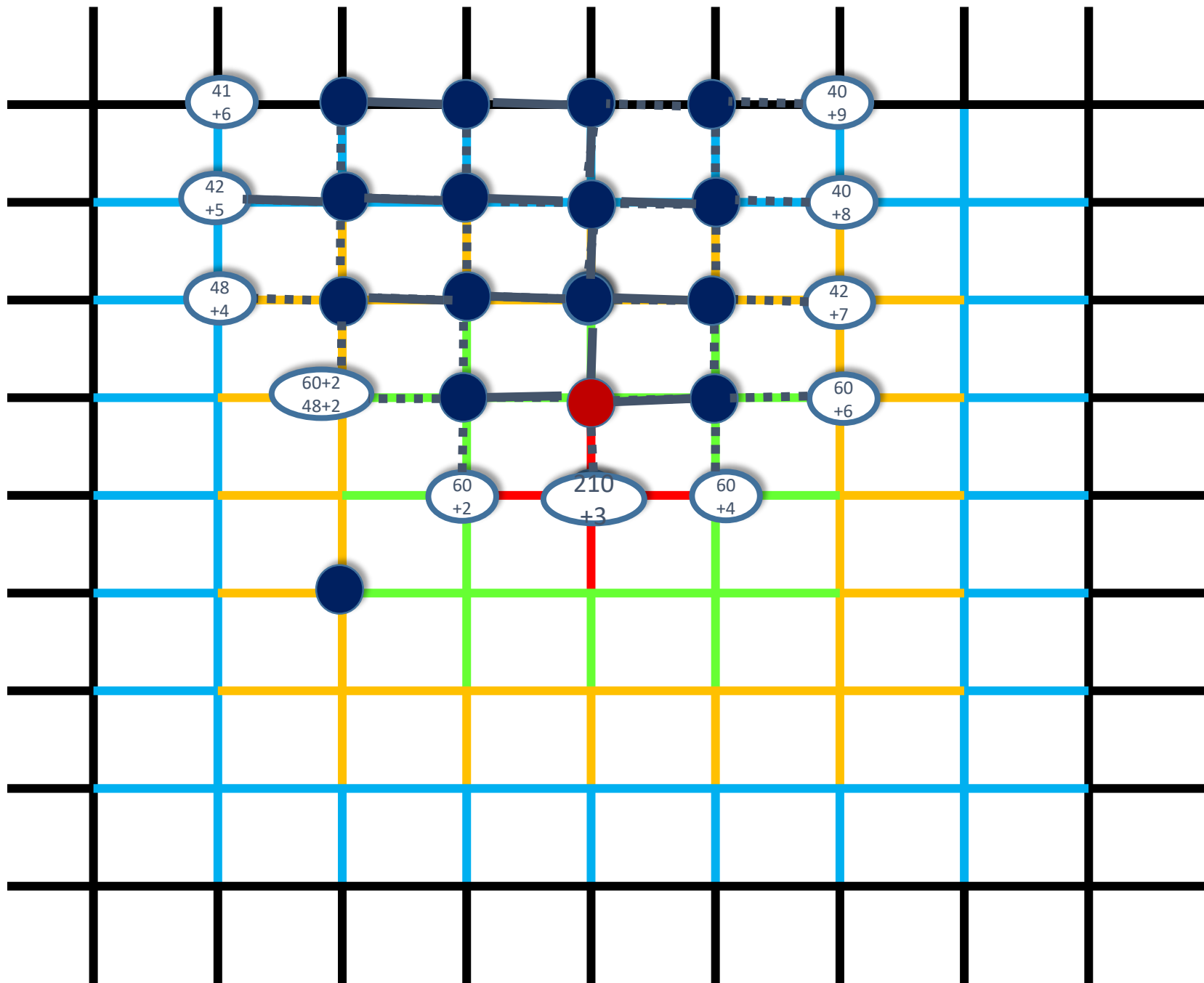
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



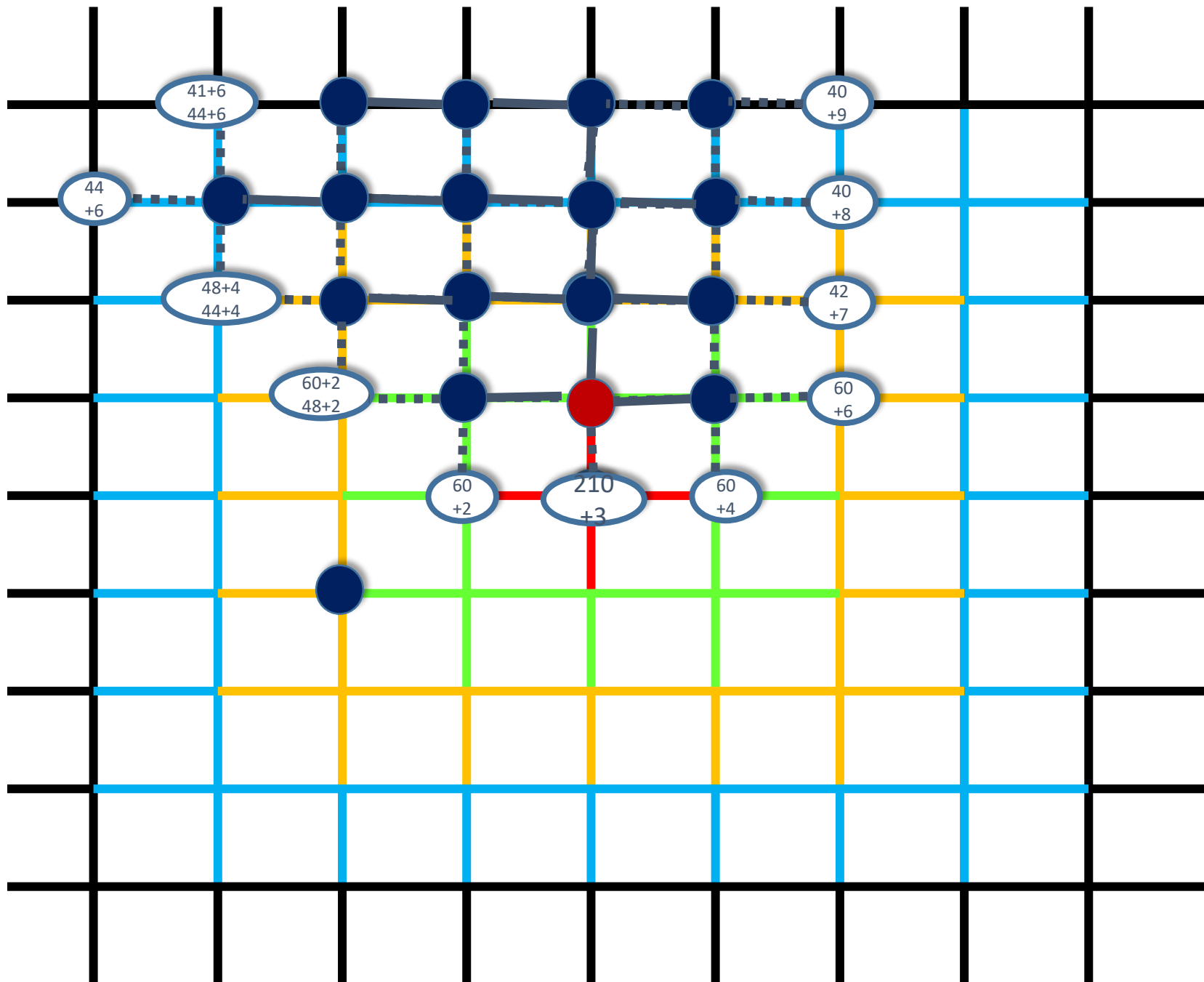
Path Cost 1

Path Cost 2

Path Cost 3

Path Cost 4

Path Cost 5



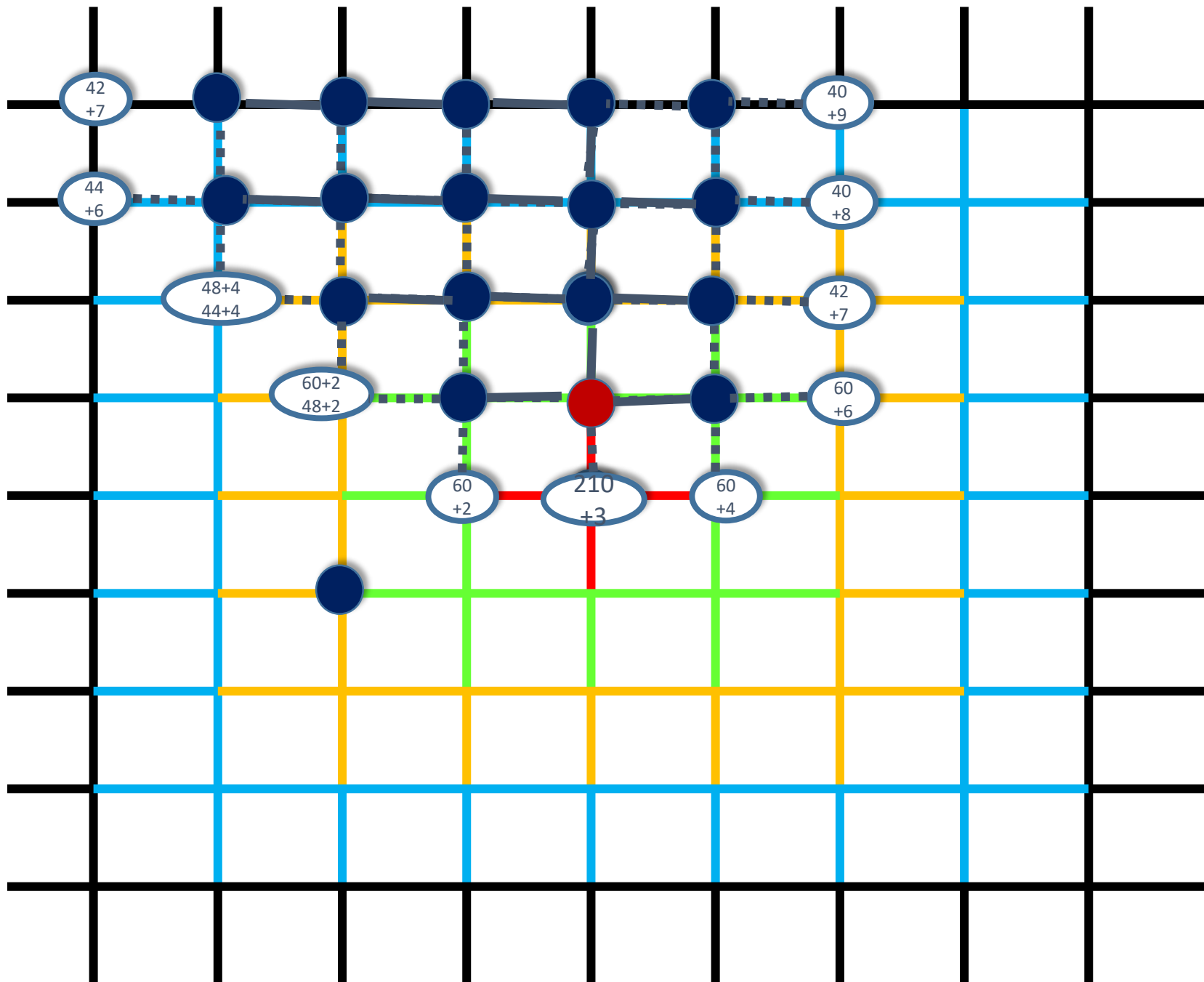
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



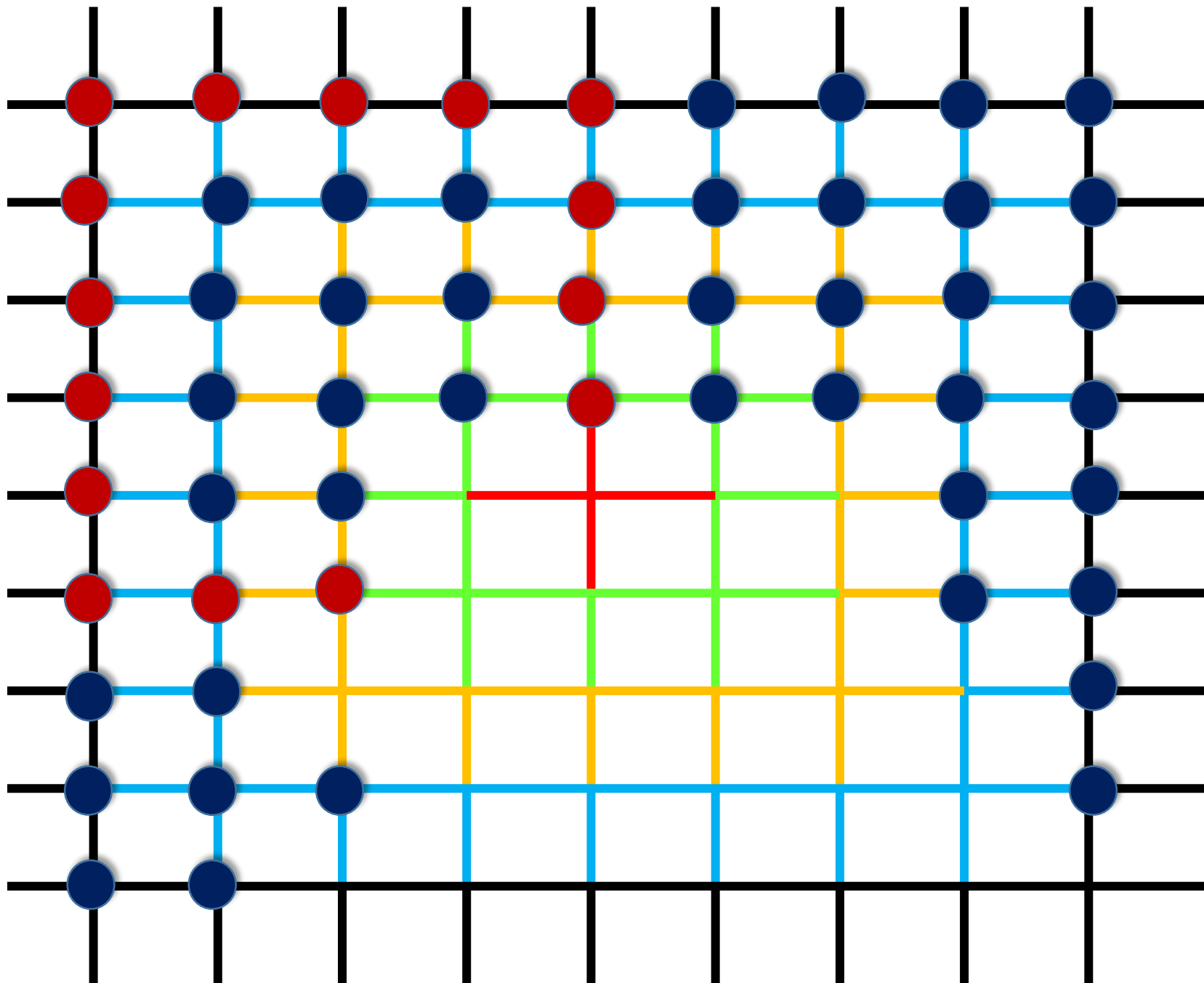
Path Cost :

Path Cost :

Path Cost :

Path Cost :

Path Cost :



Path Cost 1
Path Cost 2
Path Cost 3
Path Cost 4

Number Edges
Shortest Path

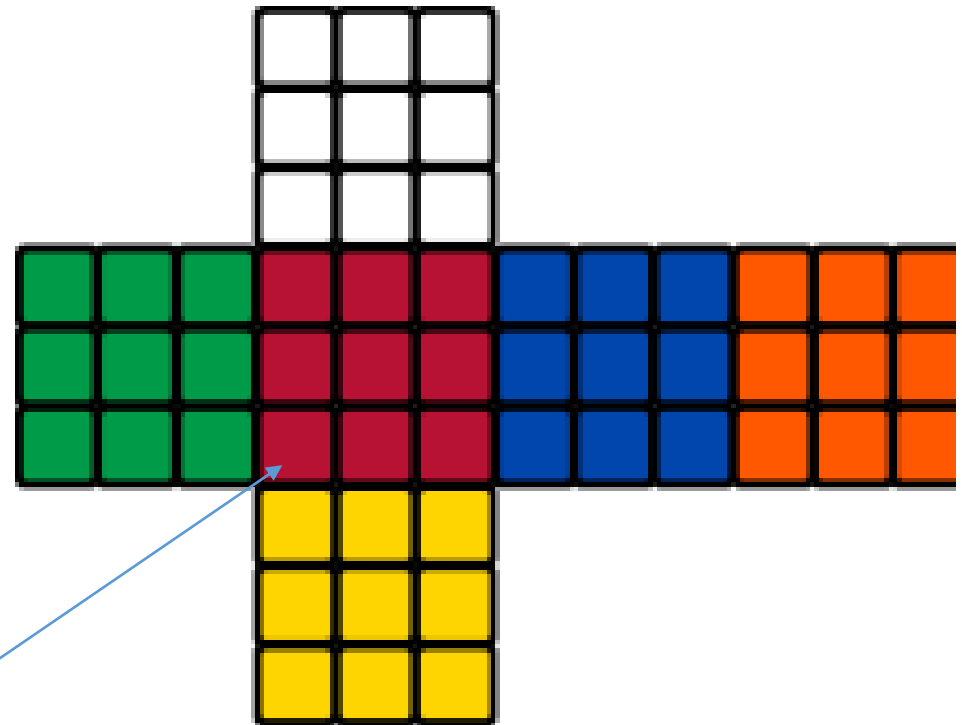
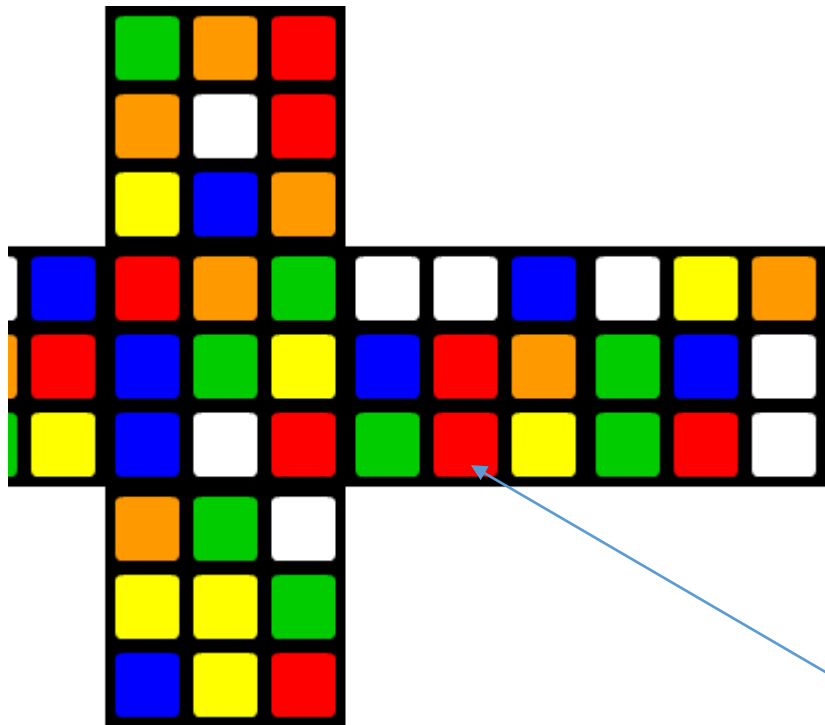
Choosing Good Heuristics

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State



Red