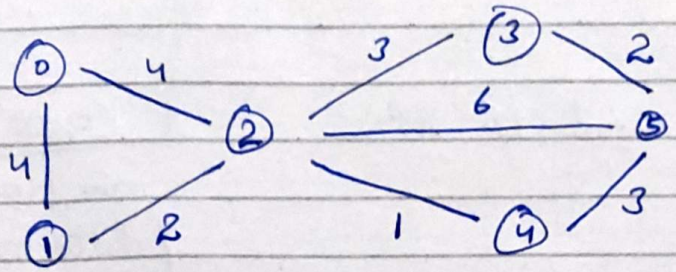
# PATH ALGORITHMS GRAPH

## Dijkstras Algorithm:

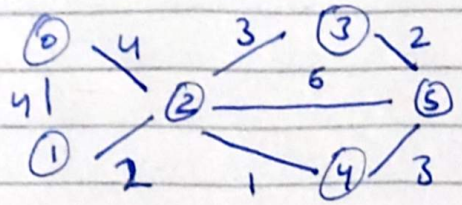
Src = 0




adj list:

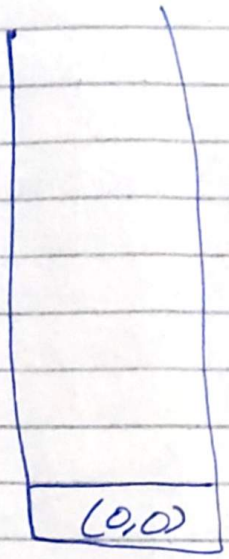
e w

0 -> {2,4} , {1,4}
1 -> {0,4} , {2,2}
2 -> {0,4} , {1,2} , {3,3} , {4,1} , {5,6}
3 -> {2,3} , {5,2}
4 -> {2,1} , {5,3}
5 -> {3,2} , {2,6} , {4,3}




Step 1

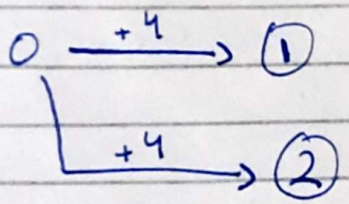
dist {0:0}




min heap
{dist, node}

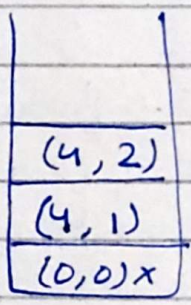
Steps are now similar to BFS

node popped = 0 , 0 → dist




node = 1
d = 4

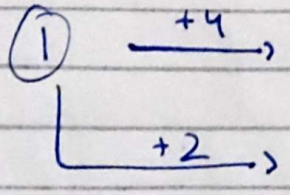
node 2
d = 4




dist = [0, 4, 4]
0, 1, 2

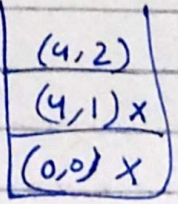
Second iteration:

node = 1 , dist = 4




node = 0
d = 8 x dist > old dist

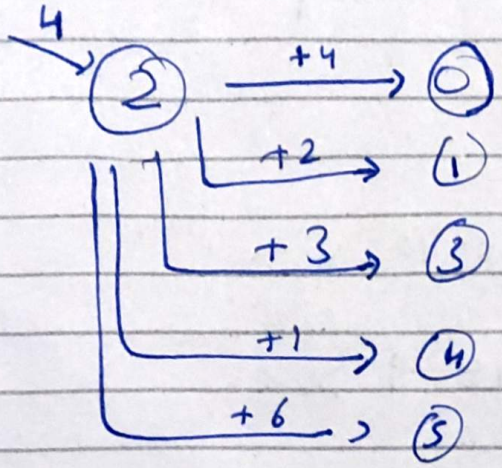
node = 2
d = 6 x dist > old dist



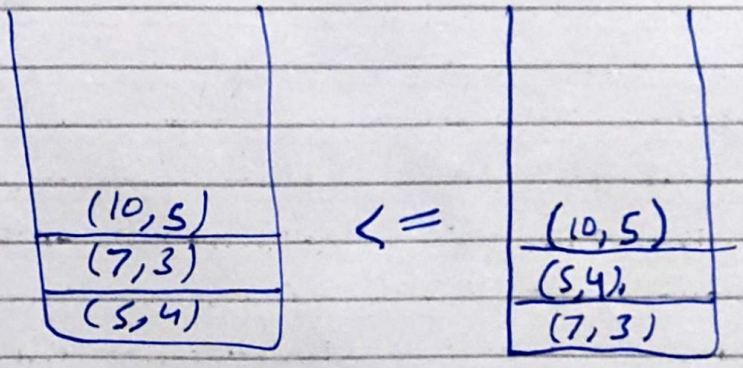

dist = [0, 4, 4]
0, 1, 2

third iteration:

node = 2 dist = 4 dist = [0, 4, 4
0 1 2

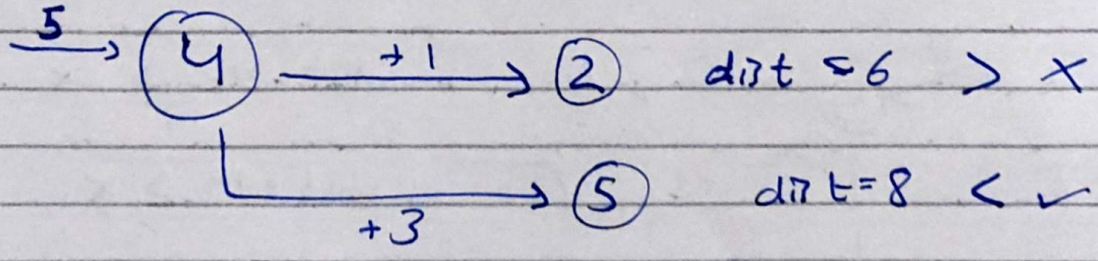



dist = [0, 4, 4, 7, 5, 10]
0 1 2 3 4 5



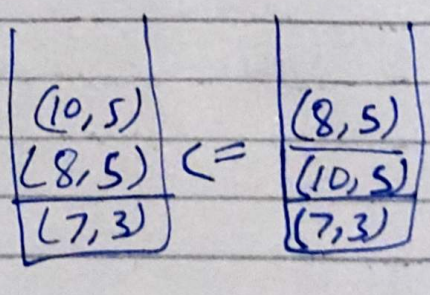

after sort min head

forth iteration:



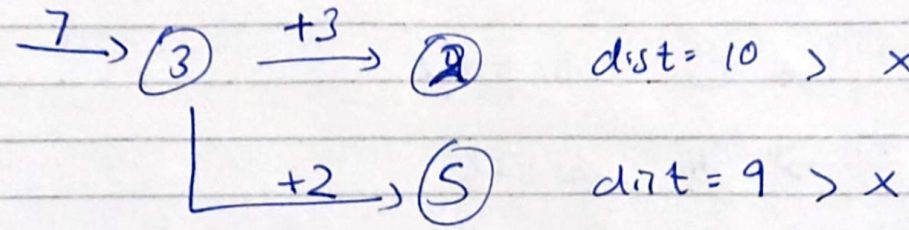

dist = [0, 4, 4, 7, 5, 8]
0 1 2 3 4 5

~~fifth iteration.~~

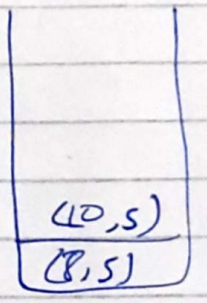


fifth iteration:

node = 3, dist = 7

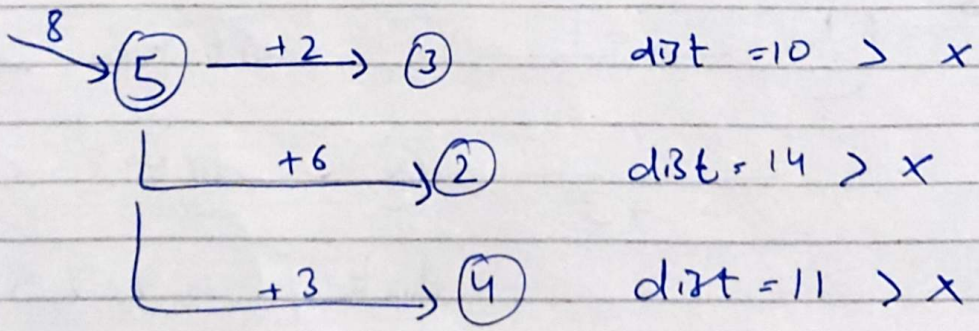



dist = [0, 4, 4, 7, 5, 8]
0 1 2 3 4 5

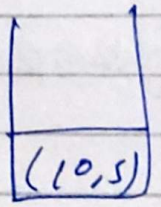



Sixth iteration:

node = 5, dist = 8




Now only heap is




which is still same

distances[0:0, 1:4, 2:4,
3:7, 4:5, 5:8]

dijkstra's is not applicable to negative weights.

for cases like
(10, 5)
or
other

we put condition

if dist > dist[node]
continue.

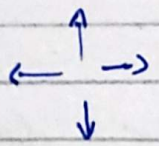
Problem 1:

Shortest distance in a binary maze

| | | | |
|---|---|---|---|
| 1 | (1) src | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | (1) dst | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |

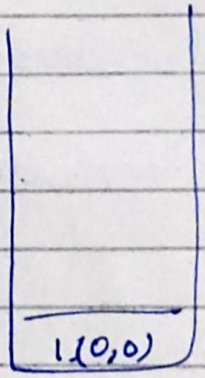
src = {0, 1}
dest = {2, 2}

we can go in 4 direction

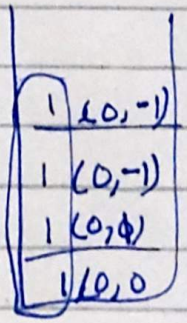


it can be used by queue. BFS

But we will do it with dijkstra.
we will not using pq
why?




suppose if we go in 4 directions




So pq is invalid same

we dont have to do log(N) sort on every adding of distance so we will just do queue.

i = 0

src

| | | | |
|---|---|---|---|
| 1 | 0 | 1 | ∞ |
| ∞ | 1 | ∞ | ∞ |
| : | ∞ | ∞ | .. |
| : | : | : | : |
| : | : | : | : |

| |
|---|
| (1,(1,0)) |
| (1,(0,2)) |
| (1,(0,0)) |
| (0,(0,1)) |

i = h ...

| | | | |
|---|---|---|---|
| 1 | 0 | 1 | 2 |
| 2 | 1 | 2 | |
| 3 | 2 | (3) | |
| ∞ | 3 | ∞ | |

dst.

Same approach is used in the path minimum effort problem.

on in leet code problem where we can go in 8 directions.

# Path with minimum effort problem.

src
↓
1 2 2
3 8 2
5 3 5 → dst

efforts

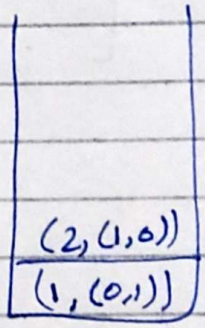
0 ∞ ∞
∞ ∞ ∞
∞ ∞ ∞


for the path man of all distance is answer.

Second iteration

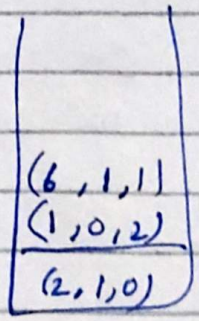
(0,0)

0 1 ∞
2 ∞ ∞
∞ ∞ ∞




~~third~~ Second iteration

0 1 1 → cuz we want to put max
2 6 ∞
∞ ∞ ∞



third iteration.

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 2 | 6 | ∞ |
| 2 | ∞ | ∞ |

| |
|---|
| (2,2,0) |
| (6,1,0) |
| (2,1,0) |

diff = 1 , (0, 2)

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 2 | 6 | 1 |
| ∞ | ∞ | ∞ |

| |
|---|
| (1,1,2) |
| (6,1,1) |
| (2,1,0) |
| (1,0,2) X |

forth iteration

diff = 1 (r, c = 1, 2)

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 2 | 6 ← x | 1 |
| ∞ | ∞ | 3 ↓ |

| |
|---|
| (3,2,2) |
| (6,1,1) |
| (2,1,0) |
| (1,1,2) X |

fifth :

diff = 2, r, c = 1, 0

| | | |
|---|---|---|
| 0 x | 1 | 1 |
| 2 ↑ x → 6 | | 1 |
| ↓ 2 | | |

| |
|---|
| (2,2,1) |
| (2,2,0) X |
| (6,1,1) |
| 3,2,2 |

Sixth

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 2 | 6 | 1 |
| 2 | 2 | 3 |

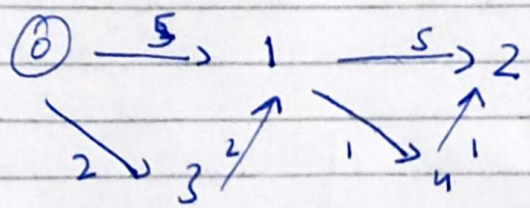
Seventh

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 2 | 6 | 1 |
| 2 | 2 | 2 |

# CHEAPEST FLIGHTS

## within K stops.

Simple dijkstra

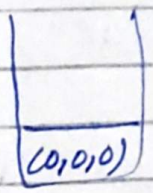



K=2

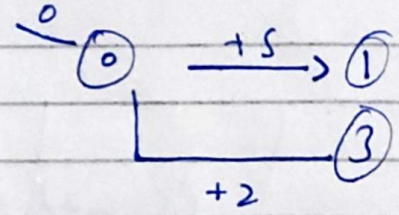
dst=2

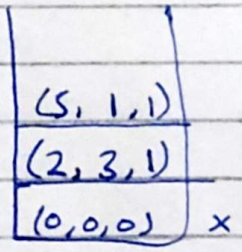
dist=[0 ∞ ∞ ∞ ∞]

0 1 2 3 4




dist, node, stops.

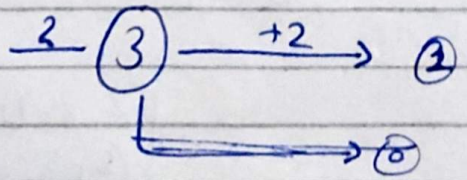


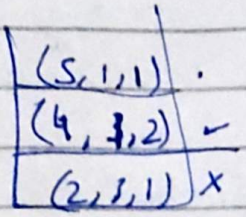

dist=[0 5 ∞ 2 ∞]

0 1 2 3 4

iteration 2:




dist=4 K ✓




dist=[0, 4, ∞, 2, ∞]

0 1 2 3 4

~~same for (4, 3, 2) cuz no neighbor~~

dist = 4 , node = 1 , 2

4 → (1) —+5→ 2 dist = 9 stops 3

└—+1→ 4 dist=5 stops=3

dist = [ 0, 4, 9, 2, 5 ]
0 1 2 3 4

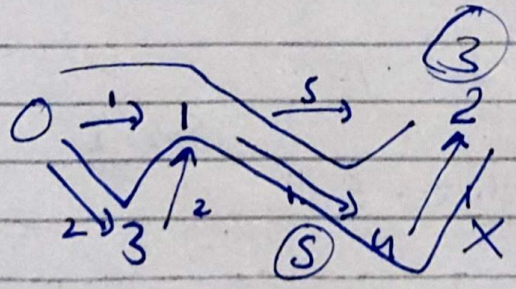
(5, 4, 3)
(9, 2, 3)
(5, 1, 1)

dist = 5, node = 1 , stop = 1

5 → (1) —+5→ 2 = 10 stop = 2

└—+1→ 4 = 6 stop = 2

dist = [0, 4, 9, 2, 5]

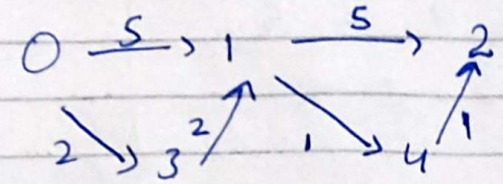



cannot go here
bcz stops >

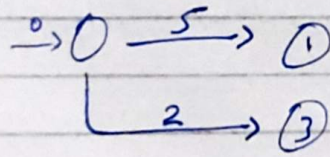
we dont even need a priority queue.

here is how.

we will store w.r.t stops

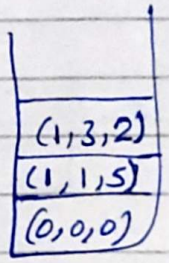



dist = 0 , node = 0 , stop = 0



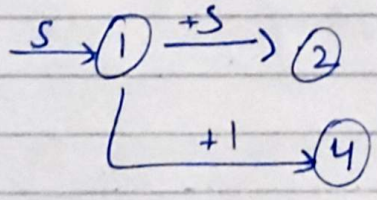

dist = 5

dist = 2




stops, node, dist.

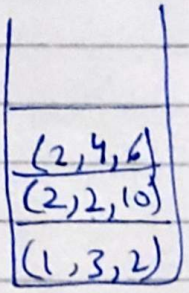
second iteration




dist = 10

dist = 6

stops = 2

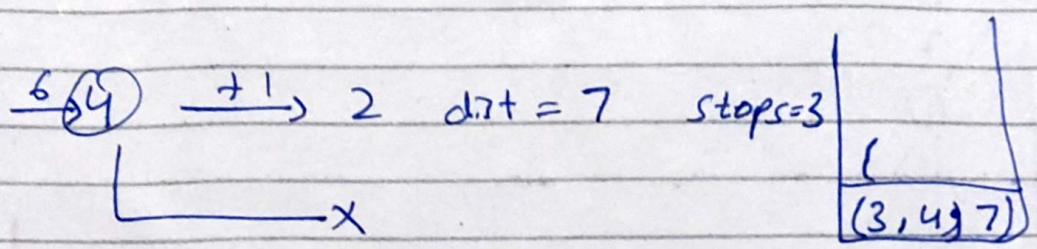



dist = [0 5 10 2 6]

0 1 2 3 4

dist=[0, ∞, ∞, ∞, ∞]  stops=2, node=2

(2) cant go from here to anywhere




ended

dist = [0, 5, 7, 2, 6]

So we ~~dist~~ didn't even need priority queue cuz of constants increment in queue and the problem is solved.