# Time complexity Of Matrix + Array

T(accessing element in Matrix) → O(1)
T(accessing element in array) → O(1)

Time complexity of the methods

① T(createPQ) = O(V) [best, worst] T(V)

V x O(1) → insert each element

② T(insert) = T(fix)

③ T(fix) = O(index) [worst] , O(1) [best]
(index → PQsize)

∵ there is only 1 element out of place (at index) when this method is called.
and it always has to be moved up (towards 0) so a max of index -1

④ T(delete) = O(PQsize) + T(fix) = max(O(n), O(n)) = O(n)

finding the index of the vertex to delete is O(PQsize)[worst] due to the array of indexes used. (O(1) → best)
index + T(index)

⑤ T(emptyPQ) = O(1)

⑥ T(initialiseAlgo) = O(4V) → O(V)

⑦ T(fillGraph) = $O(V^2)$

⑦ Dijkstra ⟶ $O(V^2) + O(E \cdot V)$

in worst case : $E = V(V-1)$

$\therefore O(V^3)$

① initialiseAlgo

② createPQ → $O(V)$

③ fix ⟶ $O(V)$

$O(V^2) + O(EV)$

while (!emptyPQ) ⟶ runs V times (∵PQ has V elements, 1 is removed per iteration)

① delete ⟶ $O(V)$

for (every neighbour not in S) {

3 comp ← constant

① delete

② insert

} $O(V)$

} $O(\frac{E}{V} \cdot V) = O(E)$

}

worst case :

$q \cdot E = (q \cdot V)(q \cdot V - 1)$ (connected graph)

⑦ Dijkstra ⟶ $O(V^2) + O(EV)$ best case : $E = V-1$

$\therefore O(V^2) + O(V \cdot V-1)$

$= O(V^2)$

① initialiseAlgo

② createPQ → $O(V)$

③ fix ⟶ $O(V)$

while (!emptyPQ) ⟶ runs V times (∵PQ has V elements, 1 is removed per iteration)

① delete ⟶ $O(V)$

$O(V^2) + O(E \cdot V)$

for (every neighbour not in S) {

3 comps ← constant

① delete

② insert

} $O(V)$

$O(\frac{E}{V} \cdot V)$

$= O(E)$

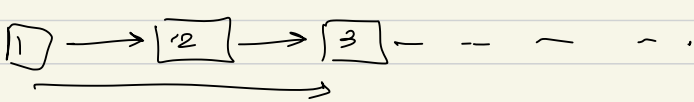
}

best case:

✓ $g \cdot E = g \cdot V - 1$ (a list)

↳ only one path → by default, shortest path

each vertex has 1 adjacent vertex



# Time complexity Of List + Heap

T(accessing element in list) → O(1) — we are only accessing the next one

T(accessing element in heap) → O(1) — due to indexes

Time complexity of the methods

① T(createPQ) = O(V) [best, worst] T(V)

V x O(1) → insert each element

③ T(rearrange) = $O(\log_2 index)$ [worst] , O(1) [best] — PQ size → V

only one element in the heap is out of place. It has to be moved up. Max → it's depth → $(\log_2 index)$
min → 1

worst case index = PQ size

worst case PQ size : V

④ T(deleteMin) = T(fix_Heap)

T(fix_Heap) → $O(2\log_2 V) = O(\log_2 V)$

⑤ T(emptyPQ) = O(1)

⑥ T(initialiseAlgo) = O(4V) → O(V)

⑦ T(fillGraph) = O(V + E)

⑦ Dijkstra $\longrightarrow O(V \log V) + O(E \log V)$

① initialiseAlgo $\rightarrow O(V)$
② createPQ $\rightarrow O(V)$
③ rearrange $\longrightarrow O(\log_2 V)$

$O(V \log V) + O(E \log V)$

while (!emptyPQ) $\longrightarrow$ runs V times (∵PQ has V elements, 1 is removed per iteration)

① deleteMin $\longrightarrow O(\log_2 V)$

for (every neighbour not in S) { $\longrightarrow O(\frac{E}{V} \log_2 V)$

3 comp ← constant
① rearrange $\longrightarrow O(\log_2 V)$

} }

worst case : $E = V \cdot (V-1)$

$O(V^2 \log V)$

best case : $E = V - 1$

$O(V \log V)$