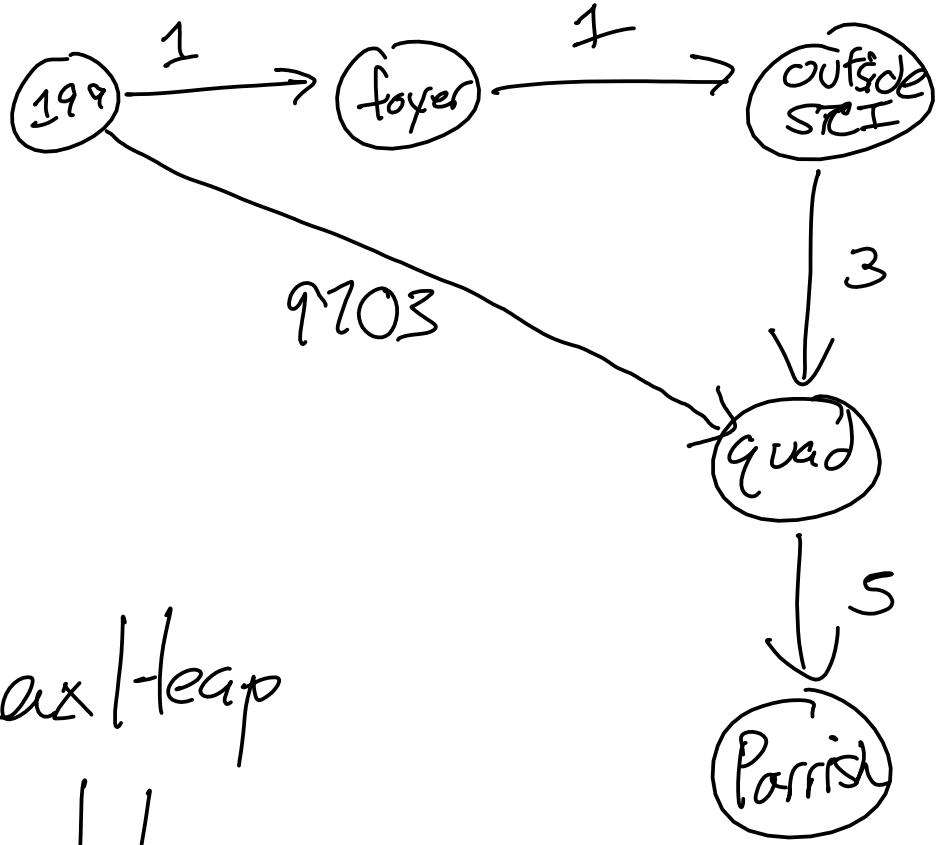
Graph Algorithms

Graph: Set of vertices
Set of edges

edge: source vertex
target vertex
weight
label

alg	in	out	ds
DFS	Graph start V end V	bool	Stack
BFS	Graph start V end V	path	Queue
BFS All	Graph start V	Dictionary end V ↦ # steps	Queue

```
Function BFSAll(Graph g, V start):
    Dictionary<V, int> dist ← new HT
    dist.insert(start, 0)
    Queue<V> q ← new LQ
    q.enqueue(start)
    While q is not empty:                          O(|V|)
        V current ← q.dequeue()                    } O(1)
        int curCost ← dist.get(current)            }
        For each neighbor n of current:            O(|V|)   } O(|E|)
            If n is not a key in dist:             }
                q.enqueue(n)
                dist.insert(n, curCost+1)
    Return dist
```



outside SCI -2

quad -9703

MaxHeap

negate all priorities

MinHeap

Function Dijkstras(Graph g, V start):

Dictionary<V, W> cost ← new HT

cost.insert(start, 0)

PQ<W, V> pq ← new MinHeap

pq.enqueue(0, start)

While pq is not empty:

O(|V|)

V current ← pq.dequeue()

W curCost ← cost.get(current)

O(|E|)

For each outgoing edge e of current:

O(|V|)

V n ← e.target

W newCost ← curCost + e.weight

If n is not a key in cost:

If not in cost

cost.insert(n, newCost)

pq.enqueue(newCost, n) ← min heap

→ Else If

newCost < cost.get(n):

cost.update(n, newCost)

pq.~~enqueue~~(newCost, n)

update

Return cost

$O(|V|^2 \log(|V|))$

Other versions

$O(|V|^2)$