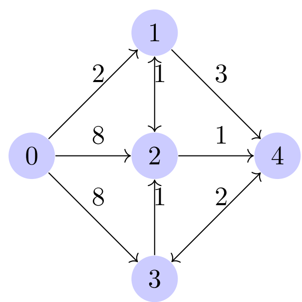
1 Dijkstra's Algorithm

Dijkstra's Algorithm is another example of greedy algorithms.



Input : Graph $G = (V, E)$ weighted w/non-negative weights. $s \in V$ (s is a vector in G).

Output : $\forall v \in V$ length of the shortest path from s to v ($\forall v$ in V)

1.1 Algorithm

Maintain $S \subseteq V$: vertices for which we already have found shortest paths. *Convention:* If $(u, v) \notin E$ i.e its not an edge then $w(u, v) = \infty$

Algorithm 1 Dijkstra's Algorithm

procedure DIJKSTRA'S ALGORITHM(G, s) ▷ $G(V, E)$ weighted graph w/non-negative weights, $s \in V$, s is a vertex in V
  $S \leftarrow \{s\}$ ▷ a set containing only s
  $d(S) \leftarrow 0$
  repeat
    Pick $v \in V - S$ that minimizes $min\, d(u) + w(u, v)$ ▷ $d \rightarrow$ distance, $w \rightarrow$ weight
    $S \leftarrow S \bigcup \{v\}$ ▷ add v to S
    $d(v) \leftarrow min\, d(u) + w(u, v)$
  until There are no more vertices that can be considered.
  return $\forall v \in V$ length of the shortest path from s to v ($\forall v$ in V).
end procedure

1.2 Correctness

Lemma 1.1. *Every time a vertex $v \in V$ is added to S. $d(v)$ is the length of the shortest path from s to v.*

Proof. By Induction
Base Case: $S = \{s\} \quad d(s) = 0$, indeed 0 is the length of the shortest path $s \rightsquigarrow s$
Induction Hypotheses: Lemma is *true* for certain s.
Induction Step: We'll prove that the lemma is still *true* after we add v to S.
Let $u \in S$ such that $d(u) + w(u, v)$ is minimum.
There exists a path $s \rightsquigarrow v$ of length $d(u) + w(u, v)$: the shortest path from s to u, then the edge (u, v)
Assume as a way of contradiction that there is a path $s \rightsquigarrow v$ of length smaller the $d(u) + w(u, v)$.

$s \in S \rightsquigarrow v \notin S$
On path from s to v there much be an edge $(u', v') \in E$ where $u' \in S, v' \in S \Rightarrow d(u') + w(u', v') < d(u) + w(u, v)$
$d(u') + w(u', v')$ is part of path that is shorter than $s \rightsquigarrow u \rightarrow v$
So the algorithm would have picked $d(u') + w(u', v')$ or better and not $d(u) + w(u, v)$.
This is a contradiction. $\square$

Note 1.1. If there are negative weights, $d(u') + w(u', v') < d(u) + w(u, v)$ does not hold.

2 Priority Queue

Maintain a set of elements, each has a value "key".

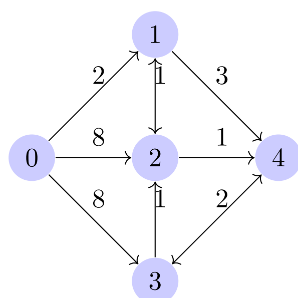
- **Insert(**Q , X**):** Insert X with value key(X)
- **Min(**Q**):** Report if $Q \neq 0$ or $Q = 0$
- **Extract-Min(**Q**):** Returns min and deletes it from Q
- **Decrease-Key(**Q , X , k**):** Changes Key(X) to k if k <key(X)

3 Dijkstra's Algorithm Implementation

Q will maintain vertices, specifically $V - S$ (set minus)
key(v) = min $d(u) + w(u, v)$ where $u \in S$

Algorithm 2 Dijkstra's Algorithm Implementation using Priority Queue

procedure DIJKSTRA'S ALGORITHM IMPLEMENTATION(G, s) $\triangleright$ $G = V, E$ size: $m = |E|$ $n = |V|$
    $\forall v \in V\,, d(v) \leftarrow \infty$
    $d(s) \leftarrow 0$
    Insert all vertices to Q. $\triangleright$ $O(n \log n)$
    while $Q \neq 0$ **do**
        $u \leftarrow$ Extract-min(Q) $\triangleright$ $O(n \log n)$
        for all neighbor v of u , $, v \in Q$ **do**
            Decrease-Key $(Q\,, v\,, d(u) + w(u, v))$
        end for $\triangleright$ $O(m \log n)$
    end while
end procedure



$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & \infty & \infty & \infty & \infty \\ - & 2 & 8 & 8 & \infty \\ - & - & 3 & - & 5 \\ - & - & - & - & 4 \\ - & - & - & 6 & - \end{bmatrix}$$

Path: 0, 1, 2, 4, 3

Note 3.1. Always remember to add $d + w$

An implementation of priority queue is binary heap. You can use an array to store all the values in a binary head, children "$\geq$" parent

- **Insert$(Q\,, X)$:** $O(\log n)$
- **Min(Q):** $O(1)$
- **Extract-Min(Q):** $O(\log n)$
- **Decrease-Key$(Q\,, X\,, k)$:** $O(\log n)$

Note 3.2. Can sort with priority queues n inserts + n extract-mins has to take $\Omega(n \log n)$