9/15/2018 # Dijkstra's Shortest Path

Input: Directed (or non-directed) graph $G = (V, E)$ — V: verteces, E: Edges.

- Each edge has nonnegative len l_e.
- Source vertex s.

Output: For each $v \in V$, compute

$L(v) :=$ length of a shortest s-v path in G.

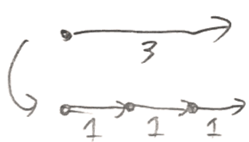
Assumptions:

1. [for convenience] $\forall v \in V$, $\exists$ an $s \rightsquigarrow v$ path. (Can use DFS, BFS for preprocessing to get rid of unreachable vertex)
2. [Important] $l_e \geq 0$ $\forall e \in E$

Is BFS enough?

$\Rightarrow$ Yes, IF $l_e = 1$ for every edge e.

Can we replace the edges by unit length edges and do BFS?



$\Rightarrow$ Blows up the graph too much.

Pseudo Code

Initialize: (Source vertex = s)

- $X = [s]$ [Explored verteces]
- $A[s] = 0$ [A: Func. to compute shorest path to dist. from s]
- $B[s] = $ None [Computed shorest path. Not needed in actual implementation]

Main loop

While $X \neq V$ $\quad v \longrightarrow w$

- From edges $(v, w) \in E$ with $v \in X$, $w \notin X$, pick the one that minimalizes

$$A[v] + l_{vw} \quad \left(\text{Dijkstra's greedy criterian}\right)$$

- add w^* to X (mean the best choice)
- Set $A[w^*] := A[v^*] + l_{v^*w^*}$
- Set $B[w^*] := B[v^*] \cup (v^*, w^*)$

⇒ In naive implementation $O(m \cdot n)$.

while loop: $(n-1)$ iteration

$\theta(m)$ for each loop

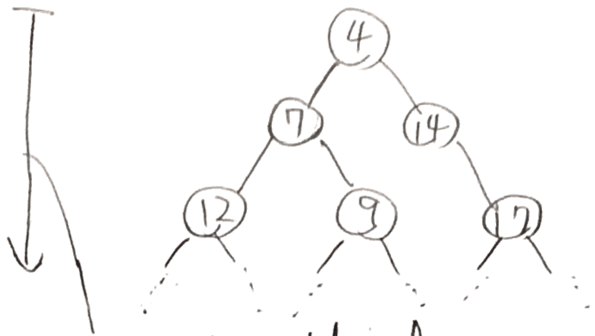
⇒ How the data structure help us?

⇒ One of the raison d'etre of the data structure, heap, is to how to do the thing minimal.

What is "heap"?

⇒ Conceptually, a perfect balanced binary tree that allows to perform Insert, Extract-min in $O(\log n)$ time.

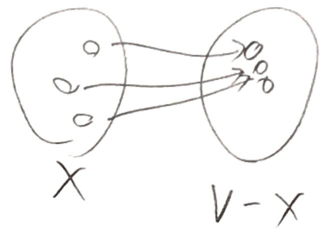
heap



Ideally, height of the tree is $\log_2 n$

- At every node, key ≤ Children's keys
- Root node is the min in tree.
- Extract-min by extract-root followed by heapifying.
- Insert via bubbling up.
- Need ability to delete from middle.

Heap invariants



#1: Elements in heap = vertices of V-X.

#2: Key[v] have to be smallest score of an possible edge to visit that node.

⇒ tricky part is to update the key in heap.

⇒ Decrease key

when w extracted from heap, and added to X.

- for each edge $(w,v) \in E$:
  - if $v \in V-X$ (i.e. in heap)
    - delete v from heap.
    - recompute $key[v] = \min(Key[v], \underbrace{A[w] + l_{wv}}_{\text{greedy score to } (w,v)})$
    - re-insert v into heap.

Speed of Dijkstra

⇒ $O(m \log n)$ like sorting

fast!