Lec 6

Thursday, September 4, 2025 6:54 AM

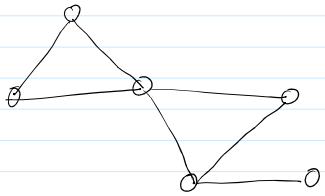
* HW 2 - due Thurs Sep 11
* Exam 1 on Sep 18 : 4 problems : 1st pb short answer ≈ 35%

Plan for today : Graph Algorithms, BFS, Shortest path, MST

Graph Algorithms

$G = (V, E)$ where $E \subseteq V \times V$

$|V| = n$, $|E| = m$ Usually $m \geq n$



Directed or undirected graph , Weighted or Unweighted edges

Degree (v): Number of edges incident to it in an undirected graph

(Otherwise, define in degree or out-degree)

Adjacency Matrix

$n \times n$ matrix

where $A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$

- Quick access by $O(n^2)$ space required
- Lookup edge (u, v) is $O(1)$

Adjacency list

* $n = |V|$ lists
* i-th entry of list v tells i-th neighbor

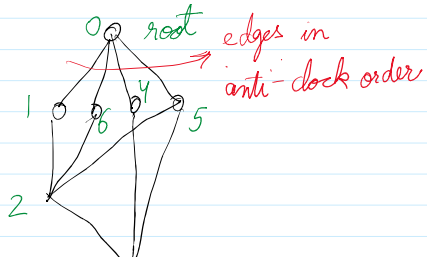
- Space is $O(n+m)$
- Lookup could take super-constant time

Questions like : Is G connected ? Find a (minimum) spanning tree, Shortest path, Max-weight Matchings , TSP, ...

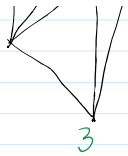
Breadth-First Search

Given a graph $G = (V, E)$ ← unweighted, undirected, adjacency list and a root $r \in V$.

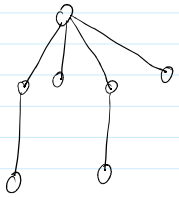
Find shortest paths from r to all vertices.



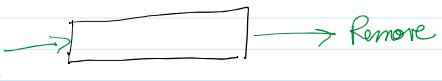
Find shortest paths from r to all vertices.



Thm: We can do this in $O(m+n)$ time.



Queue: First-in First-out



Both add/remove take $O(1)$ time

Queue $Q = \{root\}$ and array visited$[v]$ = false $\forall v \in V$

function BFS(Q)

  while $Q \neq \phi$

    v = Q.out

    visited$[v]$ = true

    for nbr u of v

      if visited(u) = false then Q.insert(v)

Time = $O(n+m)$

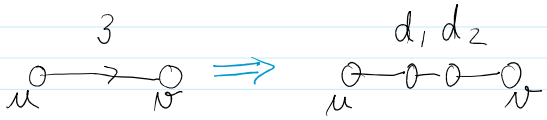
Exercise: Proof by induction that BFS-tree has shortest path (in # edges) to every vertex v

Shortest path with Edge Weights

$G = (V, E)$ and weights $w : E \to R_{\geq 0}$ ← assume W.L.O.G. integers

Given root $r \in V$, find shortest (weighted) path to all $v \in V$

Approach 1: Any edge (u, v)



Add $w_{uv} - 1$ dummy vertices s.t. unwtd length between u, v is w_{uv}.

$G \Longrightarrow G'$ run BFS(G', root).

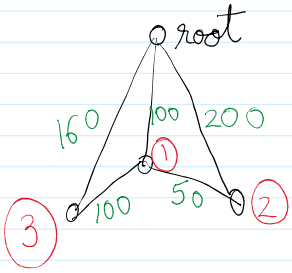
This is a pseudo-polytime algo.

This is a pseudo-polytime algo.

∴ time scales with poly (w_{max}) instead of poly$(\log w_{max}, n, \ldots)$

Approach 2: (Dijkstra's Algo)

Speedup calculations without explicitly writing G'.



Priority Queue: (key, value)
- key = vertex
- value = estimated hitting time

1) Insert / Update
2) Output min value
3) Remove a key

Assume each operation takes $O(\log(\#\text{keys}))$ time

Queue $Q = \{root\}$

function BFS(Q)

  while $Q = \phi$

    $v = Q.out$

    visited$[v]$ = true

    for nbr u of v

      if visited(u) = false then $Q.insert(v)$

$Q = \{(root, 0), (v_1, \infty), (v_2, \infty) \ldots\}$

while $Q \neq \phi$

  $v = Q.min$

  for nbr u of v:

    if $d(u) > d(v) + w(u,v)$

      $Q.update(u, d(v) + w(u,v))$

Time: m updates + n deletions

$\Rightarrow$ running time $O((m+n)\log n)$

Remarks: (1) Fibonacci heaps which allow updates in $O(1)$ time & deletions in $\log n$ time

$\Rightarrow$ time $= O(m + n\log n)$

(2) $O(m(\log n)^{2/3})$ time in STOC 2025

Breaking the Sorting Barrier for Directed Single-Source Shortest Paths

(2) $O(m(\log n)^{1/3})$ time in STOC 2025

Breaking the Sorting Barrier for Directed Single-Source Shortest Paths

From <https://arxiv.org/abs/2504.17033>

Minimum Spanning Tree

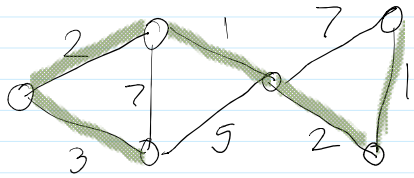
$G = (V, E)$, $w : E \to \mathbb{R}_{\geq 0}$

undirected, connected

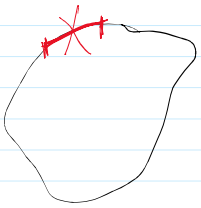
Find $T \subseteq E$ s.t.

(1) T is connected

(2) $\sum_{e \in T} w(e)$ is minimized



Obs 1: Optimal soln is a tree (there is no cycle)

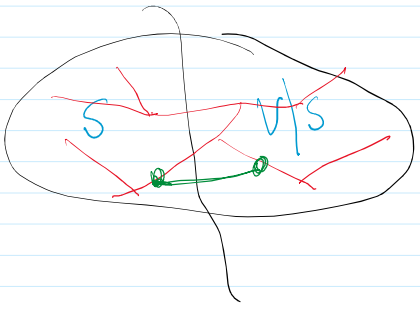


Thm: (Cut-property)

For any cut $(S, V \setminus S)$ where $S \subseteq V$ and $S \neq \emptyset$, $S \neq V$

Among edges $E(S, V \setminus S)$, the cheapest edge will be in the MST.

Pf:



Prove by contradiction:

$T \leftarrow$ optimal soln

$e \leftarrow$ cheapest cross edge

Note $T \cup e$ contains cycle

Note $T \cup e$ contains cycle

Drop the most expensive edge in this cycle.

$\Rightarrow$ Cost goes down while maintain connectivity