Dynamic Graph Algorithms

CSL-531

2026

Introduction

Dr. Shahbaz Khan
Department of Computer Science and Engineering,
Indian Institute of Technology Roorkee
shahbaz.khan@cs.iitr.ac.in

Time Complexity

$G \xrightarrow{U}$ Partial

Q: $O(1)$

Static

T

- **Preprocessing Time:** Initial time to preprocess the graph.
  - Shortest Path Tree from s
- **Update Time:** Time taken after each update to reconstruct data structure
  - ***Max:*** $O(T)$
  - ***Min:*** $O(1)$

  Q: $O(1)$
- **Query Time:** Time taken to answer query on the updated structure
  - ***Max:*** $O(T)$
  - ***Min:*** $O(1)$

$G \xrightarrow{U}$ Update $O(n)$, Query $O(1)$ — Eager

Upd $O(\log n)$, Query $O(\log n)$

Upd $O(1)$, Qu $O(n)$ — Lazy

Different tradeoffs for **Update** and **Query** time.

Worst-Case Bound vs **Amortized Bounds.**

Limitations and Bounds

For a given problem P having best static algo time Ts(n) $\Rightarrow O(n^3)$
Dynamic Algorithm Td(n):

$O(mn)$

Upper bound:

max(update,query) = Td(n) $= O(T_s(n))$

$O(m^2)$ X

Lower bound:

max(update,query) = Td(n) $\sqrt{n}$

Incremental, Decremental, Fully Dynamic?

$\rightarrow \Omega(T_s/m) \leftarrow \frac{mn}{m} = \Omega(n)$

WC

$\rightarrow$ Amortized

Dynamic Algorithm for *NP Complete Problem* in **P**?

Limitations and Bounds

For a given problem P having best static algo time Ts(n) $\Rightarrow O(n^3)$
Dynamic Algorithm Td(n):

$O(mn)$

Upper bound:

max(update,query) = Td(n) $= O(T_s(n))$

$O(m^2)$ X

Lower bound:

max(update,query) = Td(n) $\sqrt{n}$

Incremental, Decremental, Fully Dynamic?

$\Omega(T_s/m)$ ← $\frac{mn}{m}$ = $\Omega(n)$ → WC ✓ → Amortized ✓

Dynamic Algorithm for *NP Complete Problem* in **P**?

$G' = \phi$ $O(1)$

+e → $\sqrt{n}$

+e → $\sqrt{n}$

→ $\sqrt{n}$

m → $\sqrt{n}$

→ $\sqrt{n}$

G → $m\sqrt{n}$

Classical Problem

Incremental (Start from empty Graph)

Edge Updates ⇒ Preprocessing $O(1)$

Upper Bound = $O(T_s)$

Lower Bound = $\Omega(T_s/m)$] Else static algo improved

Assuming edge updates

Because Preprocessing takes $O(1)$ time

$G(V, \phi)$

updates $= m$

m edge additions

$G(V, E)$

Vertex : [V along with all edges incident on existing vertices

→ Singleton vertex : $\Omega(1)$ Add vertex w/o edges

→ Vertex with edges = $\Omega(T_s/n)$

updates $= O(n)$

For NP Hard

Update in Polynomial

↳ $\overset{n}{m} \times P$ = Poly

Works for Both WC & Amortized

Decremental

BFS tree

$G(V, V \times V)$

$n^2 - m = O(n^2)$

$G(V, E)$

Upper Bound $= O(T_s)$

Lower Bound $=$

Preprocessing $+ \; n^2 \times T_d \le T_s$

$\boxed{O(n^2 - m)} + n^2 \times T_d \le T_s$

$T_s = O(n^2)$

$T_s = \Omega(1)$ Trivial

$T_s = \omega(n^2)$

$T_d = \Omega(T_s/n^2)$

Works for Both WC & Amortized

NP Hard

TSP

Complete Weighted Graph

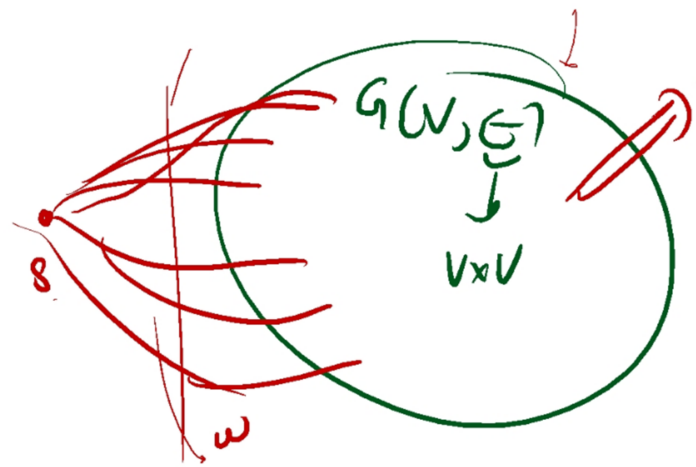
→ Find min cost cycle covering all vertices

Exception

Weighted Graph

Minimum Spanning Tree

Minimum Spanning Tree



min weight edge is w

Vertex Deletion

Upper Bound $T_d = O(T_s)$

Preprocessing + $1 \times T_d \leq T_s$

$O(n^{*})$ deletions of s

$T_d = \Omega(T_s)$

$T_d = \Theta(T_s)$

Edge Deletion

$T_d = O(T_s)$

Preprocess + #Upd $\times T_d \leq T_s$

$O(n)$ $(n-1)$

$T_d = \Omega(T_s/n)$

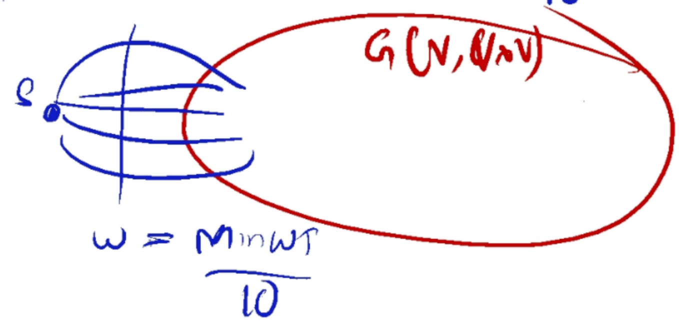
Travelling Salesman

Complete Weighted graph

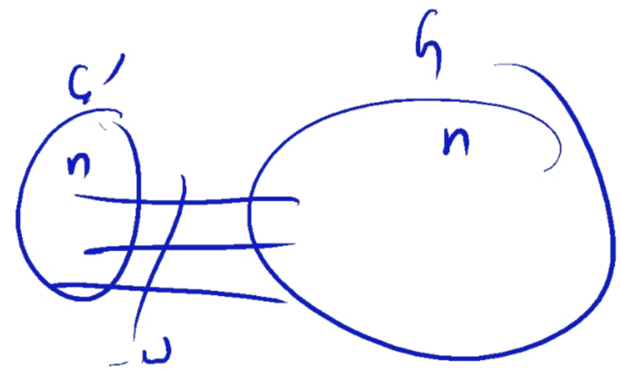
Find a cycle of minimum weight that covers all vertices

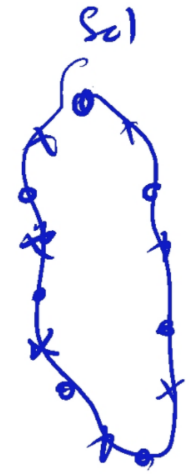
⇒ Simple (No vertex is repeated)

(W/o simple constraint | Similar to MST)



With simple cycle constraint





Vertex deletions

Preprocessing + $n \times T_d \leq T_s$

$O(n^2)$

$T_d = \Omega(T_s/n)$

Edge deletions

(Relax Complete Graph Constraint)

$O(n^2) + n^2 T_d \leq T_s$

$T_d = \Omega(T_s/n^2)$

Fully Dynamic

Algo

Td $O(T_s/n^{?})$

Increment Td $O(T_s/n^3)$

Decremental Td : $O(T_s/n)$

Lower Bound

λ

Fully dynamic Algo implies Incremental & Decremental Algo

Inc/Dec Lower bound implies equivalent bound on fully dynamic

Classical Problem

Incremental Reachability

Given a graph under edge insertions, maintain all vertices reachable from s

What is optimal update and query time?

Static

$O(m+n)$

Dynamic

$O(1)$ time

↳ Ω

Amortized?

WC. time?