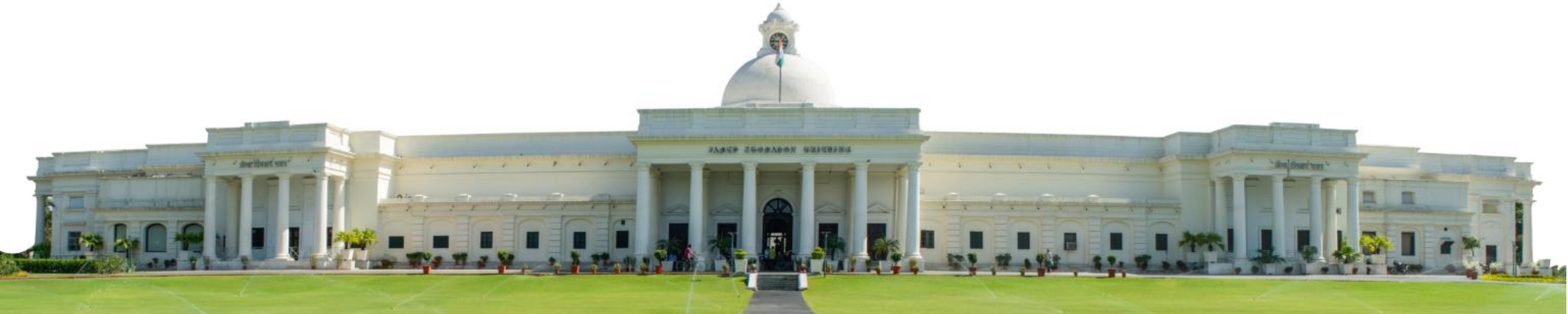
Dynamic Graph Algorithms

CSC-531

Incremental All Pairs Connectivity

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



Disjoint Set Union

Operations:

- ***Find(x)*** = Report the set containing x
- ***Union(x,y)*** = Merge the sets containing x and y

Implementation:

Tree:

Store all elements in Set Si, in a Tree T[i]

Store pointer to parent in Tree

Find(x) Report Root(x)

Union(x,y) Find(x) and Find(y). Make Root of ***smaller*** tree a child of the root of ***larger*** tree

1- *Rank(single element) = 0*
2- Union T(r1) and T(r2)
If (Rank(r1)>= Rank(r2))
Make r1 as root
Rank(r1)= max(Rank(r1),Rank(r2)+1)

WIth Smaller Rank Heuristic
=> Height
O(log n)

Height

Disjoint Set Union

n_{par} = # vertices whose par are changed

Path Compression:

After Find(x), Make Root(x) the parent of each element on the path.

$\log^* n$

$$T(n) = O(1) + O(|P|)$$

$$O(1) + O(x^1_{par}) + O(y^2_{par})$$

$$= O(1) + O(1) + O(n_{par}) = \text{Local}\ (O(\log^* n) + \text{global}\ O(x_{par}))$$

Algorithm:

Find(x) — Report Root(x). ***Path Compression on path from x to Root(x)***

Union(x,y) — **a** = Find(x) and **b** = Find(y).

Make Root of ***smaller*** tree a child of the root of ***larger*** tree

If(Rank[a] >= Rank[b])

Make *b* the child of *a*

Rank[a] = max(Rank[a], Rank[b]+1)

else

Make *a* the child of *b*

Rank[b] = max(Rank[b], Rank[a]+1)

$O(1)$

x par = # vertices whose new par in same group

y par = # new — — — diff group

Disjoint Set Union

WIth Smaller Rank Heuristic

Property 1: If Rank(k) = r and root k
=> k has $\geq 2^r$ descendants
Property 2: Rank(k) < Rank (par(k))
Property 3: Number of vertices
with Rank r $\leq n/2^r$

1- *Rank(single element) = 0*
2- Union T(r1) and T(r2)
If (Rank(r1)>= Rank(r2))
Make r1 as root
Rank(r1)= max(Rank(r1),Rank(r2)+1)

group (k) = log*(rank (k))

log*n = { 0 if n<=1
1+ log*(log n) if n>1

Group	Rank
0	[0,1]
1	(1,2]
2	(2,2^2]
3	(2^2,2^2^2]
g	(2^^g-1, 2^^g]

Property 1: Number of Groups => log* n
Property 2: Number of Vertices in Group g =>
$\leq n/2\text{^^}(g-1)$

Ackerman's Function

$\boxed{K(m,n)}$ Tarjan
$\alpha(n)$ CLRS

$A_{k+1}(x.rank) = A_k^{x.rank+1}(x.rank)$

$A_k(j)$ for $k \geq 0$ and $j \geq 1$ is defined as

$$A_k(j) = \begin{cases} j+1 & \text{if } k == 0 \\ A_{k-1}^{j+1}(j) & \text{if } k \geq 1 \end{cases}$$

$\overbrace{A_k \circ A_k \circ A_k \circ \cdots A_k(j)}^{i}$

$A_k^0(j) = j,$ $\quad A_k^1(j) = A_k(j)$ $\quad A_k^i(j) = A_k(A_k^{i-1}(j))$, for $i \geq 1$

Properties:

1- $A_1(j) = 2j + 1$

2- $A_2(j) = 2^{j+1}(j+1) - 1$

3- $A_3(1) = 2047,$

4- $A_4(1) > 2^{2048} > 10^{80}$ # atoms in universe

Inverse Ackerman's Function

$\alpha(n) = \min\{k: A_k(1) \geq n\}$

$0 = \alpha(n)$ for $n \in [0,2]$, $\quad 1 = \alpha(n)$ for $n \in [3]$, $\quad 2 = \alpha(n)$ for $n \in [4,7]$,

$3 = \alpha(n)$ for $n \in [8,2047]$,

$4 = \alpha(n)$ for $n \in [2048, 2^{2048}]$

$A_4(1)$

Potential Function

$\phi = \sum \phi(v)$ for $v \in V$

$$\phi(v) = \begin{cases} \alpha(n) \times x.rank & x \text{ is root or } x.rank = 0 \\ (\alpha(n) - level(x)) \times x.rank - iter(x) & else \end{cases}$$

$level(x) = \max\{k : par(x).rank \geq A_k(x.rank)\}$

$iter(x) = \max\{i : par(x).rank \geq A^{i}_{level(x)}(x.rank)\}$

const

par(x).rank < A_{k+1}(x.rank) 2

$A_k^{x.rank+1}( )$, 1

Iter

level

0

x.rank

Properties

1- $level(x) \in [0, \alpha(n)]$

2- $iter(x) \in [1, x.rank]$

3- $\phi(x) \in [0, \alpha(n) \times x.rank]$

x.rank

A_k $A_k \circ A_k \circ A_k$(x.rank)

A_{k+1}(x.rank)

$\Delta\phi(v) \leq -1$ If iter or level changes] Not root &

Complexity

- Cost of UNION

HAW

Complexity

- Cost of FIND

HW