Project Solutions

COMP9312_24T2

Q1 Index-based Shortest Distance

Reachability queries can be answered using the labels:

- ? $u \rightsquigarrow v$

  if $L_{out}(u) \bigcap L_{in}(v) \neq \emptyset$ then return true

  if $L_{out}(u) \bigcap L_{in}(v) = \emptyset$ then return false

Extend the idea to shortest distance~

2-Hop Cover for Shortest Distance

– each node u is assigned two label sets $L_{in}(u) \subseteq V$ and $L_{out}(u) \subseteq V$

– for item $(v, vd) \in L_{out}(u)$: the shortest distance from u to v is d.

– for item $(v', v'd) \in L_{in}(u)$: the shortest distance from v' to u is $v'd$.

Extend to undirected graphs:

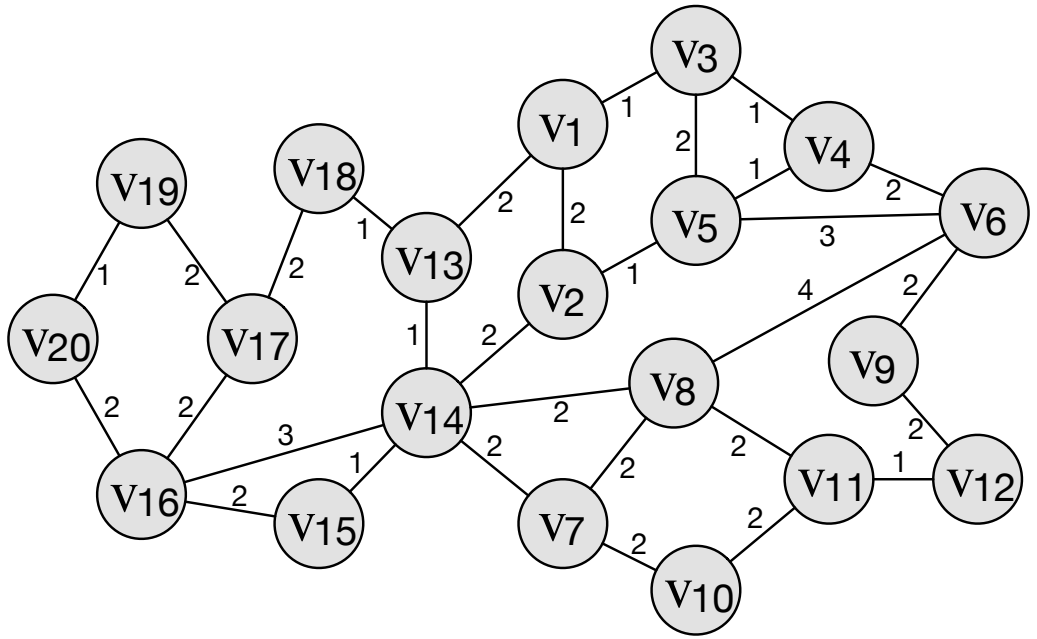
– each node u is assigned one label set $L(u) \subseteq V$

– for item $(v, vd) \in L(u)$: the shortest distance between u and v is d.

$$dist(s,t) = \min_{u \in L(s) \cap L(t)} dist(s,u) + dist(u,t)$$

2-Hop Cover for Shortest Distance

V	2-Hop Label
v_1	$\{(v_1,0),(v_2,2),(v_{13},2),(v_{14},3)\}$
v_2	$\{(v_2,0),(v_{14},2)\}$
v_3	$\{(v_1,1),(v_2,3),(v_3,0),(v_4,1),(v_5,2),(v_6,3),(v_{13},3),(v_{14},4)\}$
v_4	$\{(v_1,2),(v_2,2),(v_4,0),(v_5,1),(v_6,2),(v_{13},4),(v_{14},4)\}$
v_5	$\{(v_1,3),(v_2,1),(v_5,0),(v_6,3),(v_{14},3)\}$
v_6	$\{(v_1,4),(v_2,4),(v_6,0),(v_{13},6),(v_{14},6)\}$
v_7	$\{(v_6,6),(v_7,0),(v_8,2),(v_{14},2)\}$
v_8	$\{(v_6,4),(v_8,0),(v_{14},2)\}$
v_9	$\{(v_1,6),(v_2,6),(v_6,2),(v_7,7),(v_8,5),(v_9,0),(v_{11},3),(v_{13},8),(v_{14},7)\}$
v_{10}	$\{(v_6,7),(v_7,2),(v_8,4),(v_{10},0),(v_{11},2),(v_{14},4)\}$
v_{11}	$\{(v_6,5),(v_7,4),(v_8,2),(v_{11},0),(v_{14},4)\}$
v_{12}	$\{(v_1,8),(v_6,4),(v_7,5),(v_8,3),(v_9,2),(v_{11},1),(v_{12},0),(v_{14},5)\}$
v_{13}	$\{(v_{13},0),(v_{14},1)\}$
v_{14}	$\{(v_{14},0)\}$
v_{15}	$\{(v_{14},1),(v_{15},0),(v_{16},2)\}$
v_{16}	$\{(v_{14},3),(v_{16},0),(v_{18},4)\}$
v_{17}	$\{(v_{13},3),(v_{14},4),(v_{16},2),(v_{17},0),(v_{18},2)\}$
v_{18}	$\{(v_{13},1),(v_{14},2),(v_{18},0)\}$
v_{19}	$\{(v_{13},5),(v_{14},6),(v_{16},3),(v_{17},2),(v_{18},4),(v_{19},0),(v_{20},1)\}$
v_{20}	$\{(v_{13},6),(v_{14},5),(v_{16},2),(v_{17},3),(v_{18},5),(v_{20},0)\}$

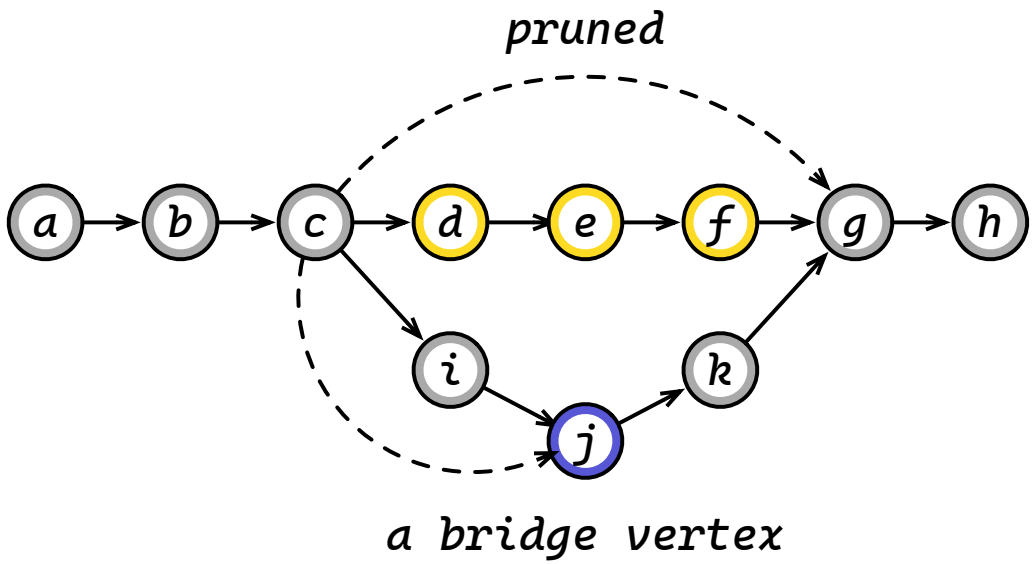


An example of 2-hop label for the above graph

```
Dist(v1,v20) = 8
```

How to compute a minimal 2-hop label

The idea of reachability:

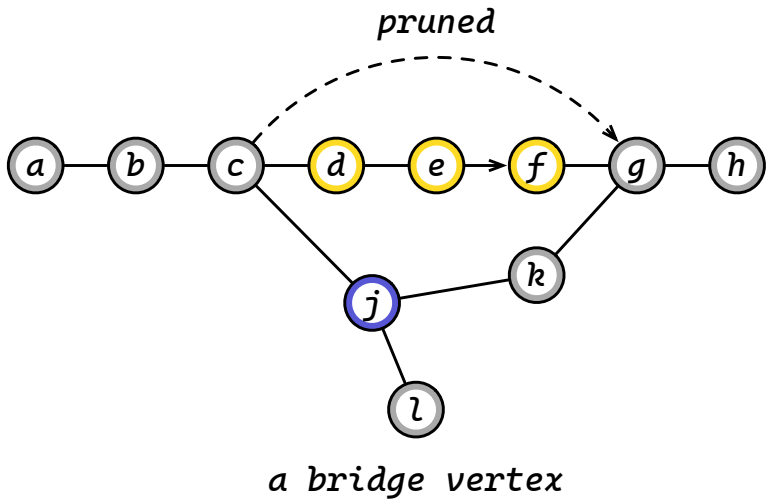


How to compute a minimal 2-hop label

Extend to shortest distance:

$(j, 1) \in L(\mathrm{c}), (j, 2) \in L(\mathrm{g})$

$dist(c, g) = dist(c, j) + dist(j, g)$



To Compute a Minimal 2-Hop Cover

compute a minimal 2-hop cover for reachability in a directed graph

For each node u in the graph from high-degree to low-degree:

- *add u into both $L_{in}(u)$ and $L_{out}(u)$;*
- *mark u as processed;*
- *conduct BFS from u and for each reached node w:*
  - *if (u,w) has been covered: stop exploring out-neighbors of w;*
  - *else: add u into $L_{in}(w)$;*
- *conduct reverse BFS from u and for each reached node w':*
  - *if (w',u) has been covered: stop exploring in-neighbors of w';*
  - *else: add u into $L_{out}(w')$;*

To Compute a Minimal 2-Hop Cover

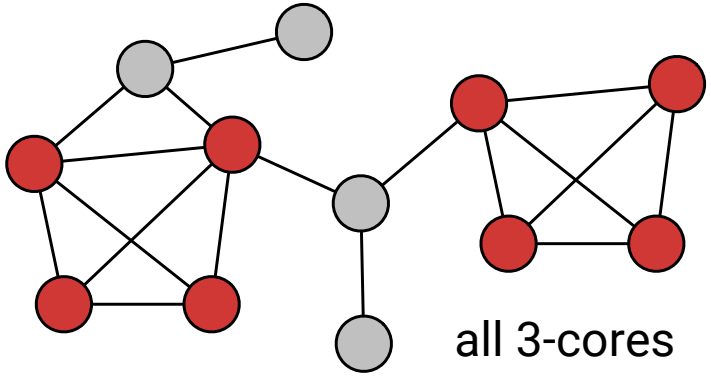
Extend to shortest distance in an undirected unweighted graph

For each node u in the graph from high-degree to low-degree:

- *add (u,0) into both L(u) and $L_{out}(u)$;*
- *mark u as processed;*
- *conduct BFS from u and for each reached node w and dist(u,w):*
  - *if (u,w) has been covered: stop exploring out-neighbors of w;*
  - *else: add u into $L_{in}(w)$;*

(u,w) has been covered: dist(u,w) computed by the existing index = dist(u,w)

Q2 K-Core Vertex Set



A non-trivial baseline:

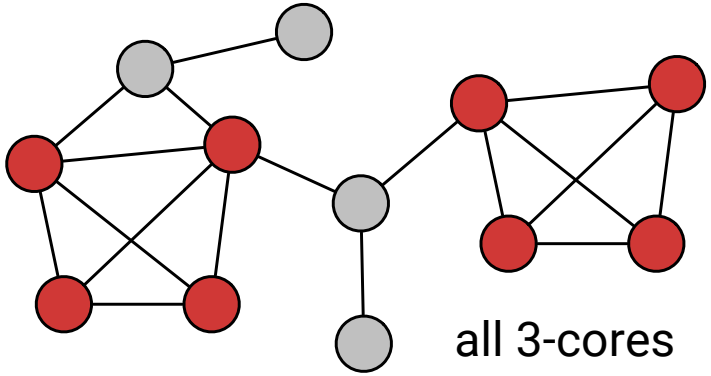
Index: core numbers of all vertices

Query: Compute connected components by edges between resulting vertices

Index Space: O(n)+graph, indexing time: O(m), query time: O(m)

A correct implementation of this method should have 10—13 points

Q2 K-Core Vertex Set



Move forward a little bit:

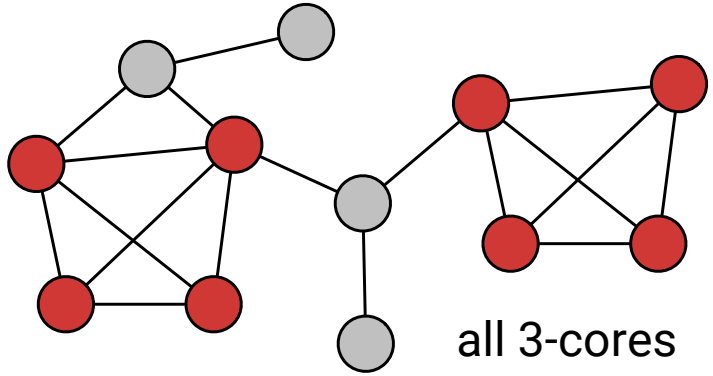
Index: core numbers of all vertices, sorted edges

Query: Compute connected components by edges between resulting vertices

~~Index Space: O(n)+graph, indexing time: O(m), query time: O(m)~~

Index Space: O(m), indexing time: O(m), query time: O(#edges in k-core)

Q2 K-Core Vertex Set



Opportunities:

Index Space: O(m), indexing time: O(m), O(#edges in k-core)

What you can do to achieve better query time, e.g., O(n), optimal?

Pathway to design an index-based solution:

Query Efficiency -> Index Space -> Indexing time

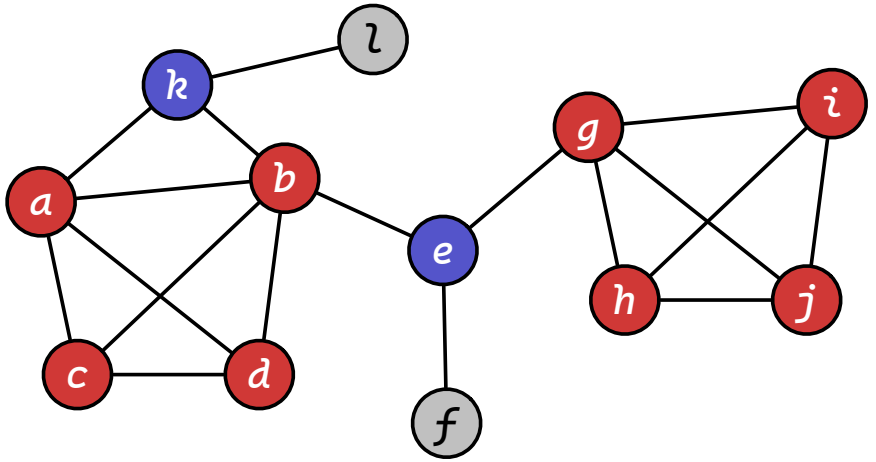
Start by ignoring index space and index time for now…

Q2 K-Core Vertex Set

What index can help you achieve the optimal query time complexity?

Optimal means the time is linear to the result size

```
//3-core
[[a,b,c,d],[g,h,i,j]]
//2-core
[[a,b,c,d,g,h,i,j,k,e]]
//1-core
[[a,b,c,d,g,h,i,j,k,e,l,f]]
```

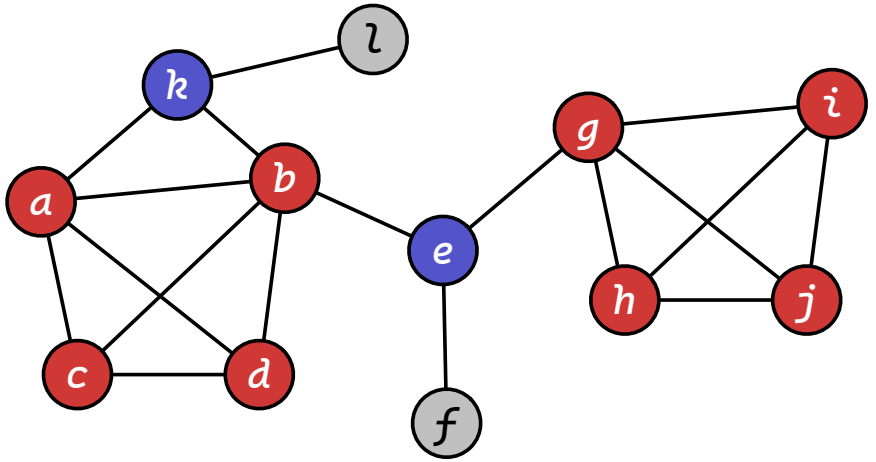


Index size?

Q2 K-Core Vertex Set

A great amount of overlaps/redundancy in the current index…

```
//3-core
[[a,b,c,d],[g,h,i,j]]
//2-core
[[a,b,c,d,g,h,i,j,k,e]]
//1-core
[[a,b,c,d,g,h,i,j,k,e,l,f]]
```

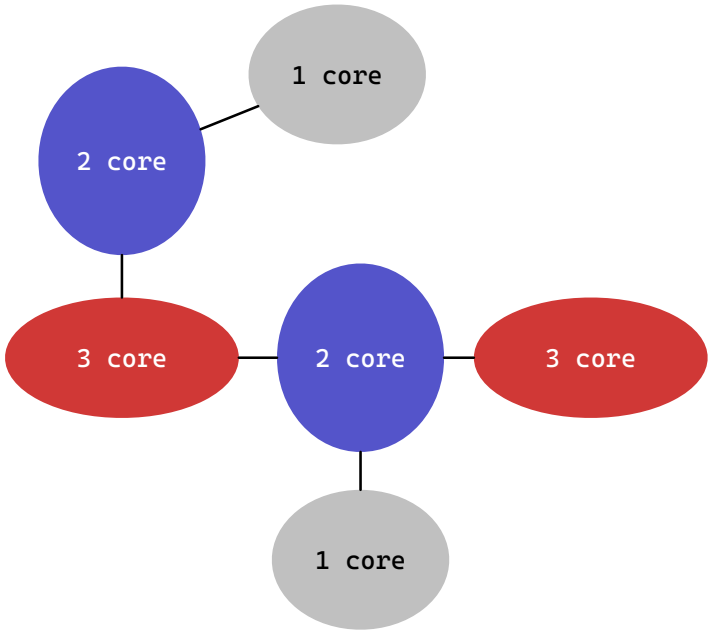


How to avoid the redundancy?

Q2 K-Core Vertex Set

How to avoid the redundancy?

```
//3-core, all good for the moment, no redundancy
[[a,b,c,d],[g,h,i,j]]
```

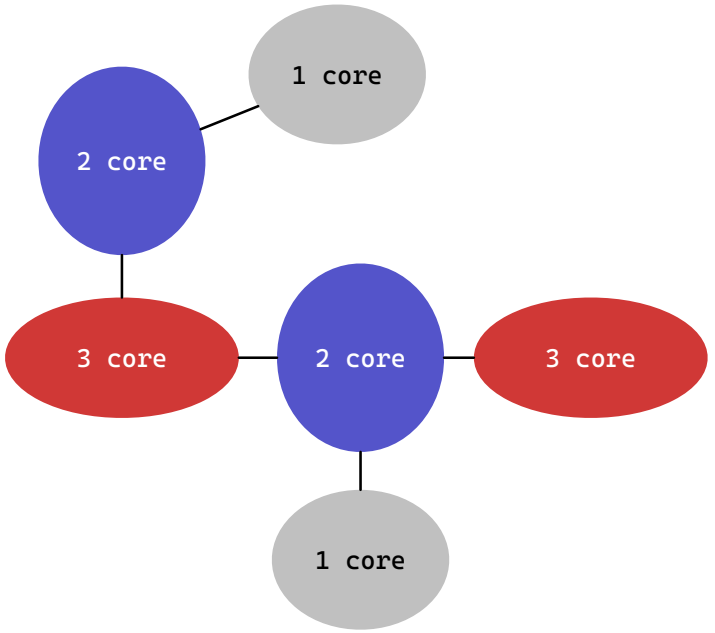


The term k-core in the figure is not rigorous because 2-core should contain 3-core. The correct term for this case is k-shell.

Q2 K-Core Vertex Set

How to avoid the redundancy?

```
//3-core
[[a,b,c,d],[g,h,i,j]]
//2-core?
[[a,b,c,d,g,h,i,j,k,e]]
```



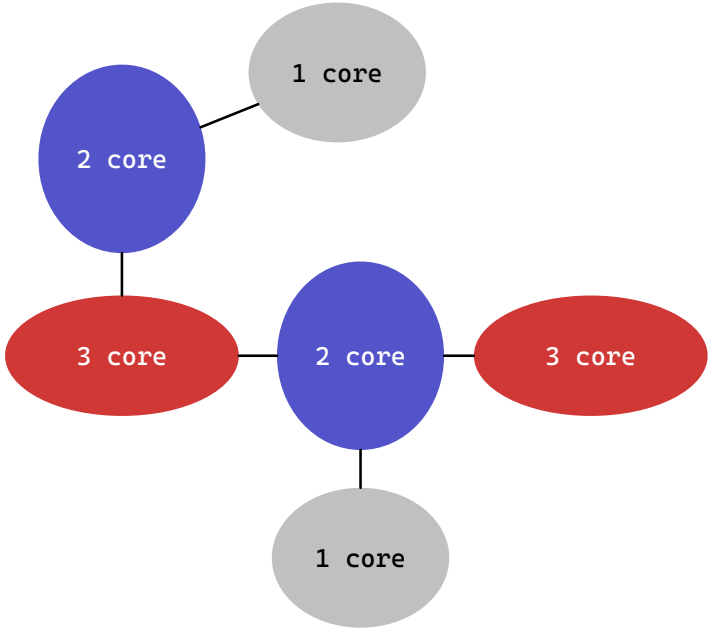
Vertices in 3-core belong to the 2-core of k and e if they are connected.

So we need a structure to connect them…

Q2 K-Core Vertex Set

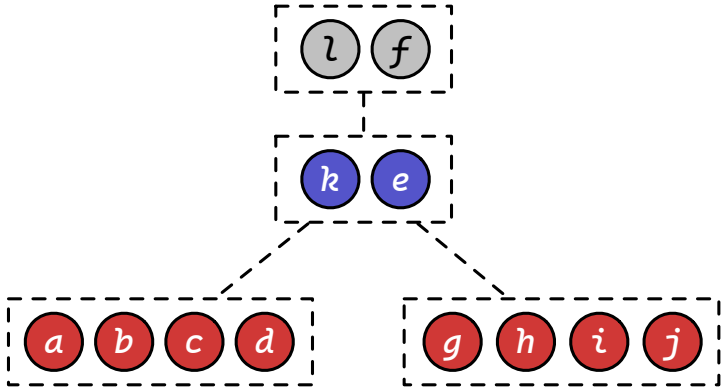
How to avoid the redundancy?

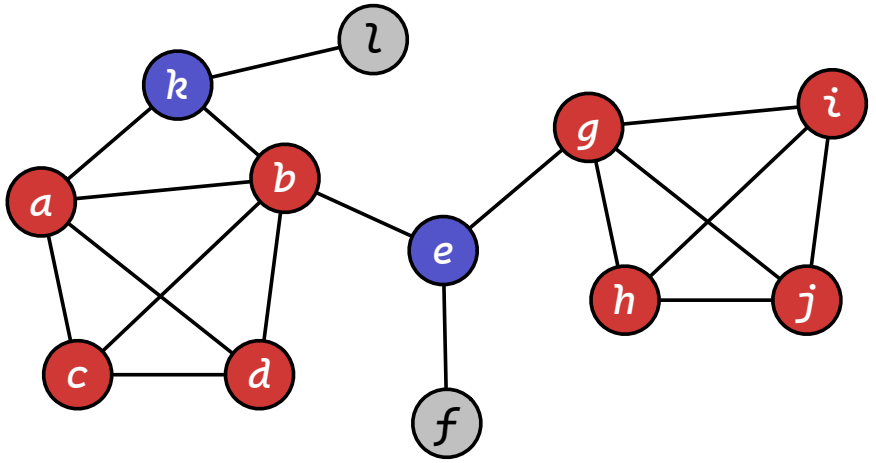
```
//3-core
[[a,b,c,d],[g,h,i,j]]
//2-core
[[a,b,c,d,g,h,i,j,k,e]]
//1-core?
[[a,b,c,d,g,h,i,j,k,e,l,f]]
```



Q2 K-Core Vertex Set

The index…





Index Space: O(n)

Q2 K-Core Vertex Set

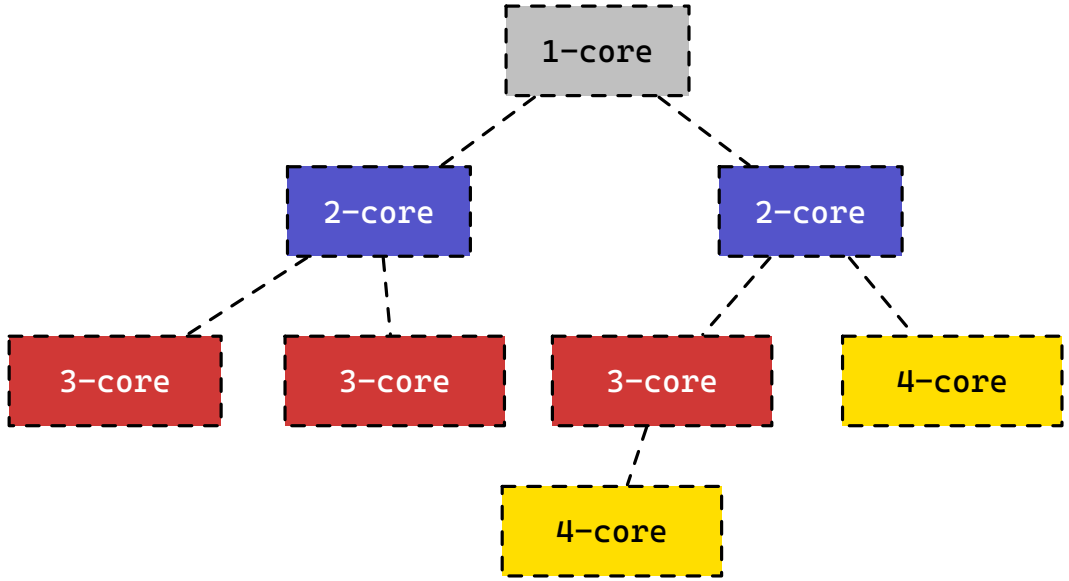
Given a query integer k, how to process queries?

Option 1:

1. Start from the largest core number
2. Search upward

Option 2:

1. Start from the tree node of k
2. Search downward



A generalized example

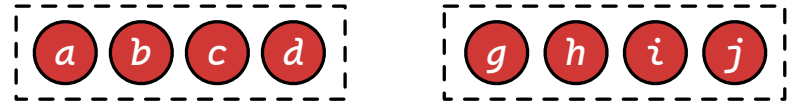
Now we have O(n) index space and optimal query time.

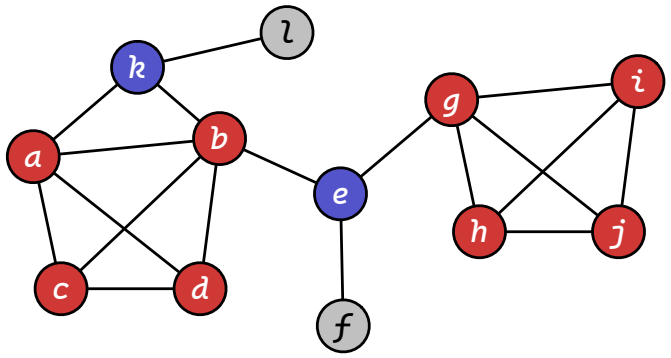
Q2 K-Core Vertex Set

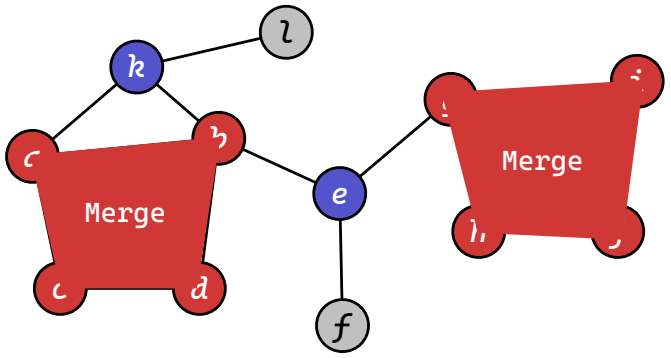
Index Construction

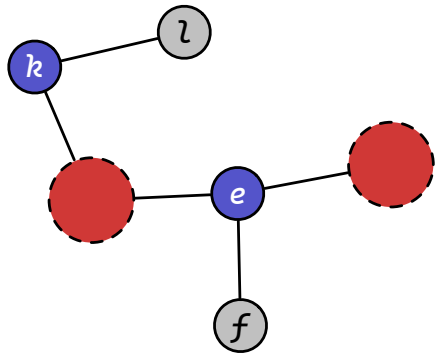
Connectivity -> disjoint set

Merge from the largest core number, i.e., 3





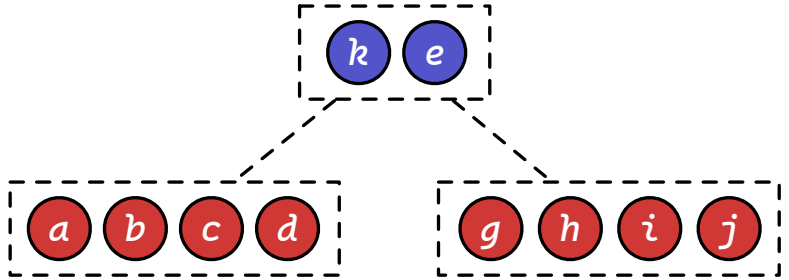


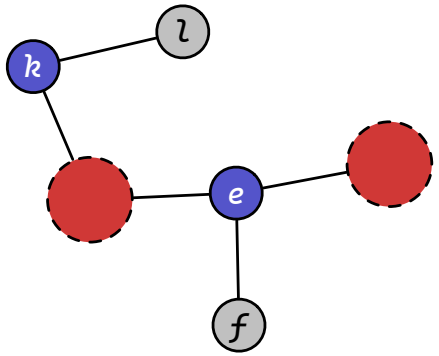


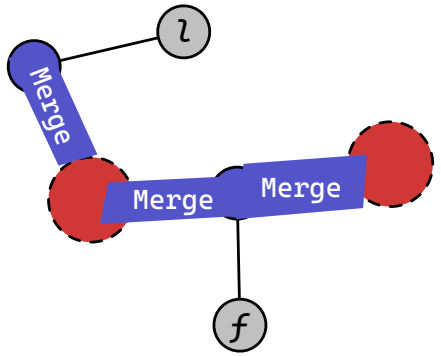
Q2 K-Core Vertex Set

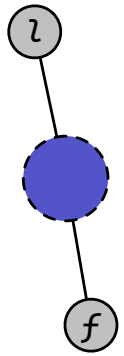
Index Construction

Merge for 2-core





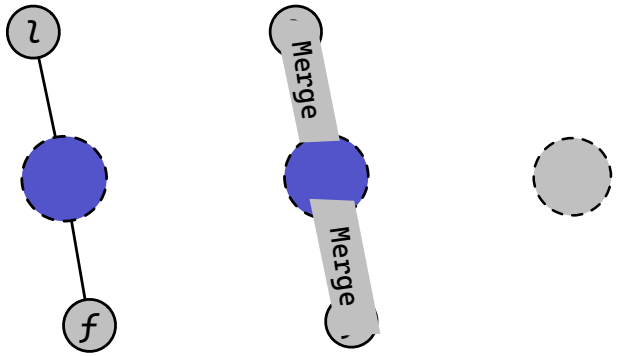


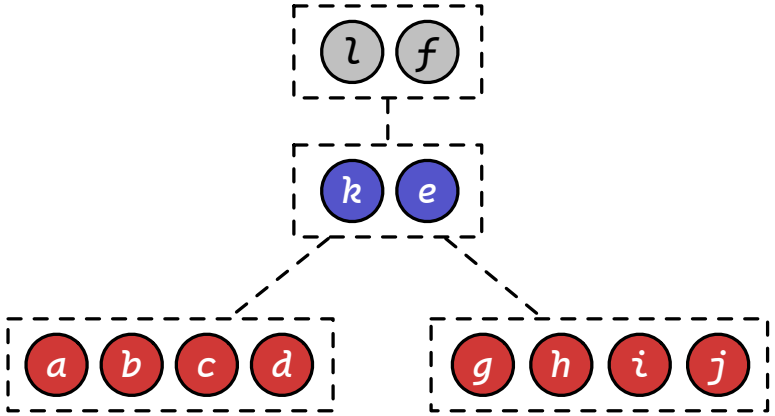


Q2 K-Core Vertex Set

Index Construction

Merge for 1-core





Indexing time: O(m), index space: O(n), query time: optimal

Sample code

Request the project sample solution for the project by email