18-344 Recitation 8

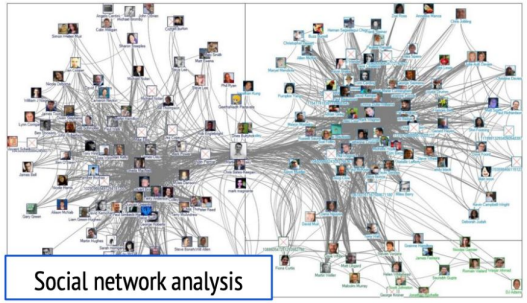
Lab4 - Graph Processing Optimization

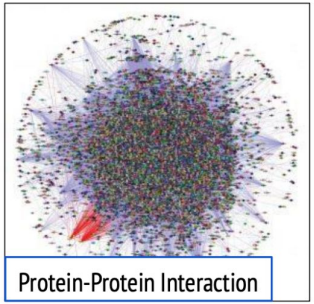
About Sparse Problems

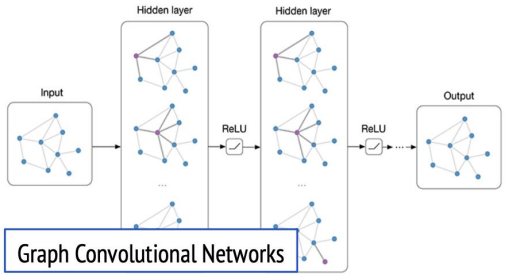
Sparse Problems

- What is a sparse problem? Why are they called “sparse”?
  - Graph Processing Problems are Sparse Problems
  - Machine Learning Problems are Sparse Problems
- What makes sparse problems hard?

Path Planning

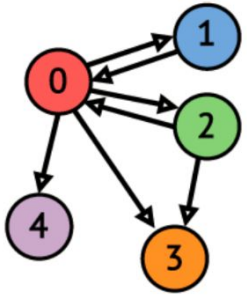

Social network analysis


Protein-Protein Interaction



Graph Convolutional Networks

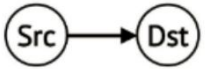
What does a graph processing program look like?



0	1
2	0
1	0
0	2
2	3
0	4
0	3

COO
(EdgeList)

```
for e in EL:
  dstData[e.dst] =
     f(srcData[e.src],dstData[e.dst])
```



Edge

dstData

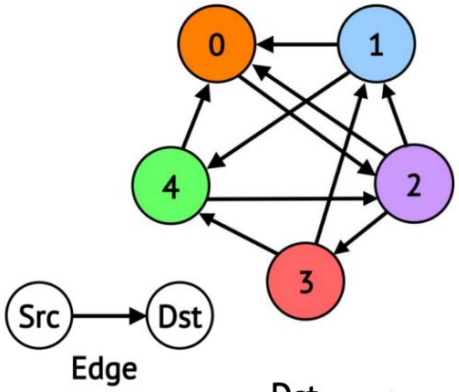
srcData

stores vertex property information

if srcData == dstData, updating in-place;

often "swap" srcData & dstData from 1 iteration to the next iteration

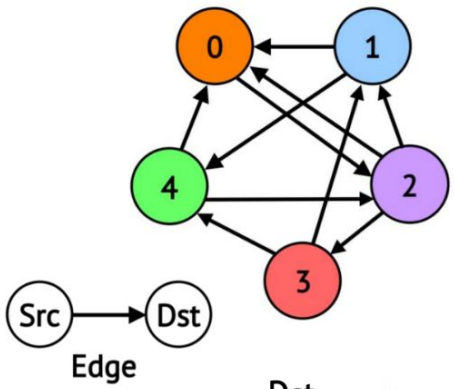
Nobody EVER uses the adjacency matrix!

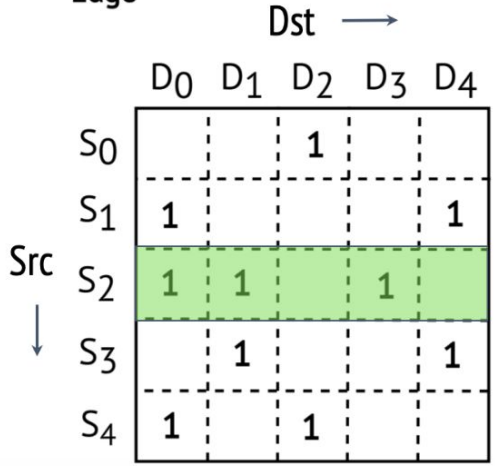


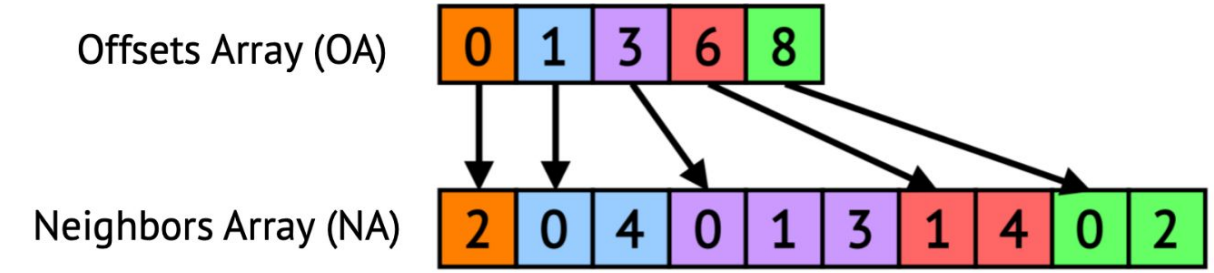
Src ↓ / Dst →	D_0	D_1	D_2	D_3	D_4
S_0			1		
S_1	1				1
S_2	1	1		1	
S_3		1			1
S_4	1		1		

Why would the Adjacency Matrix not be used?

Compressed Sparse Data Structures for Feasible Memory Size





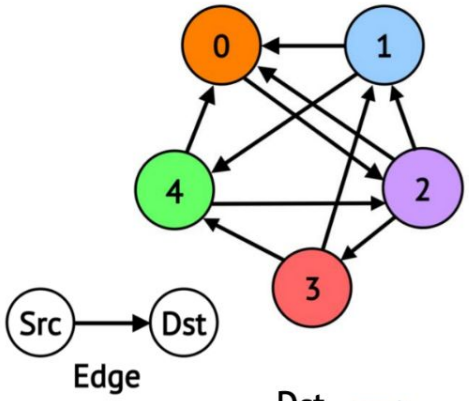


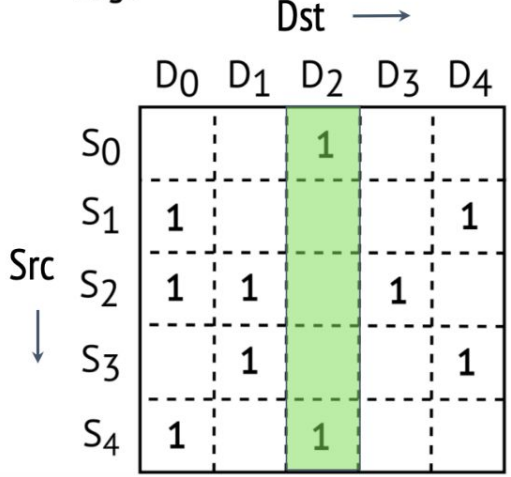
Compressed Sparse Row (CSR)
Outgoing Neighbors

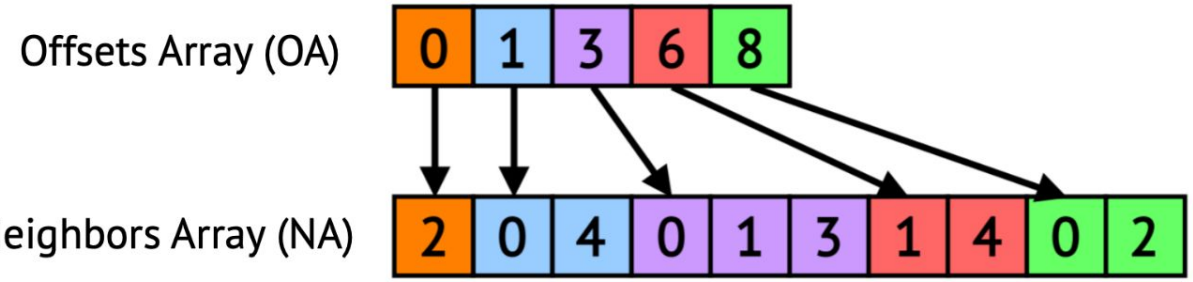
Vertex Property Array
i.e., srcData / dstData

Often we will leave the vertex property array implicitly defined when we talk about sparse structures, but it is always there

Compressed Sparse Data Structures for Feasible Memory Size



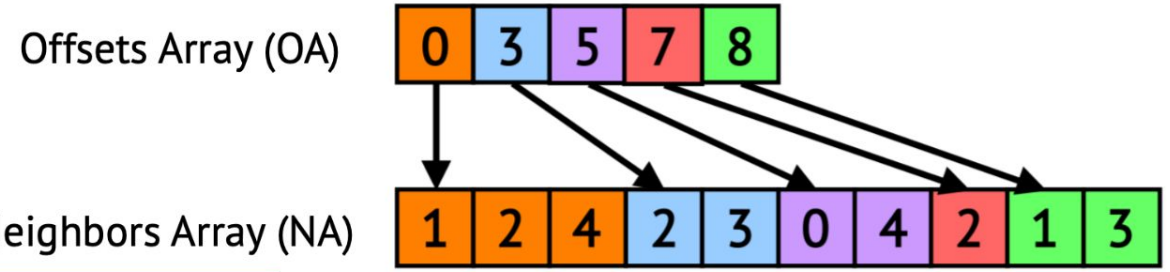




EdgeList sorted by SrcIDs

Compressed Sparse Row (CSR)
Outgoing Neighbors

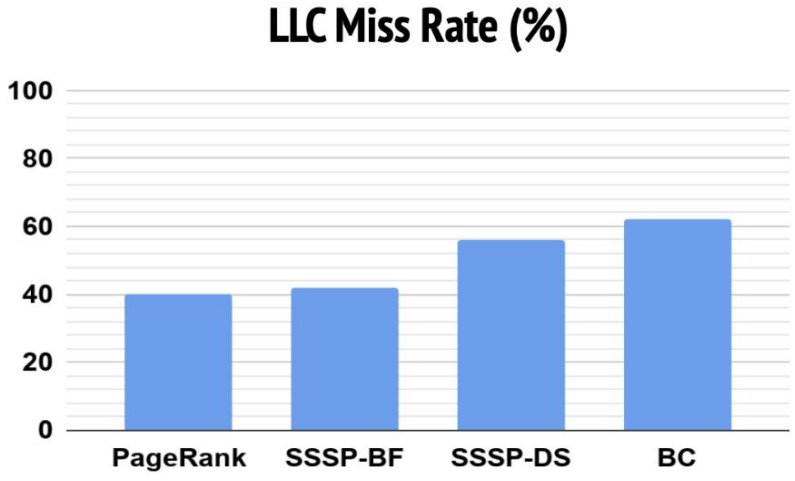
The CSC is the *transpose* of the CSR

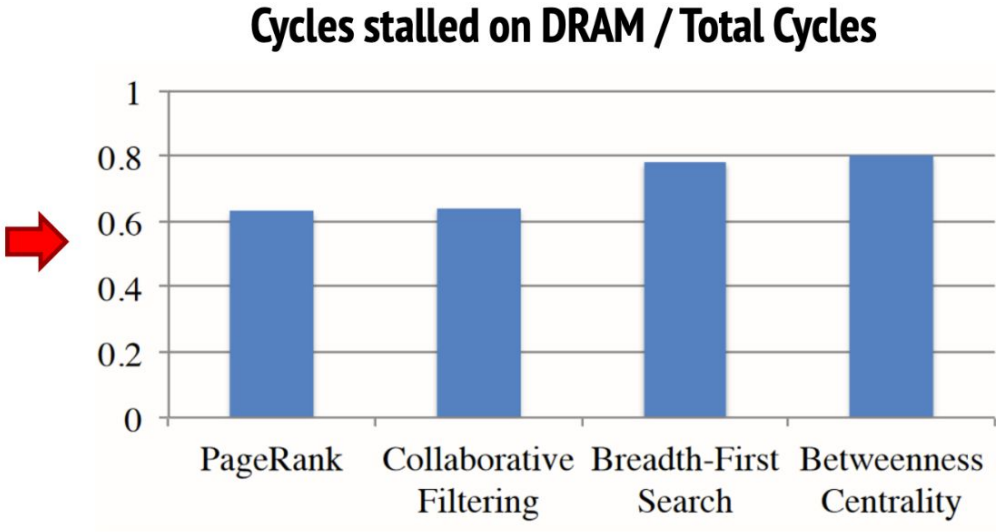


EdgeList sorted by DstIDs

Compressed Sparse Column (CSC)
Incoming Neighbors

Irregular Accesses Lead to Poor Locality



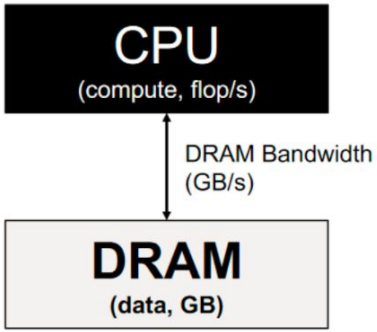


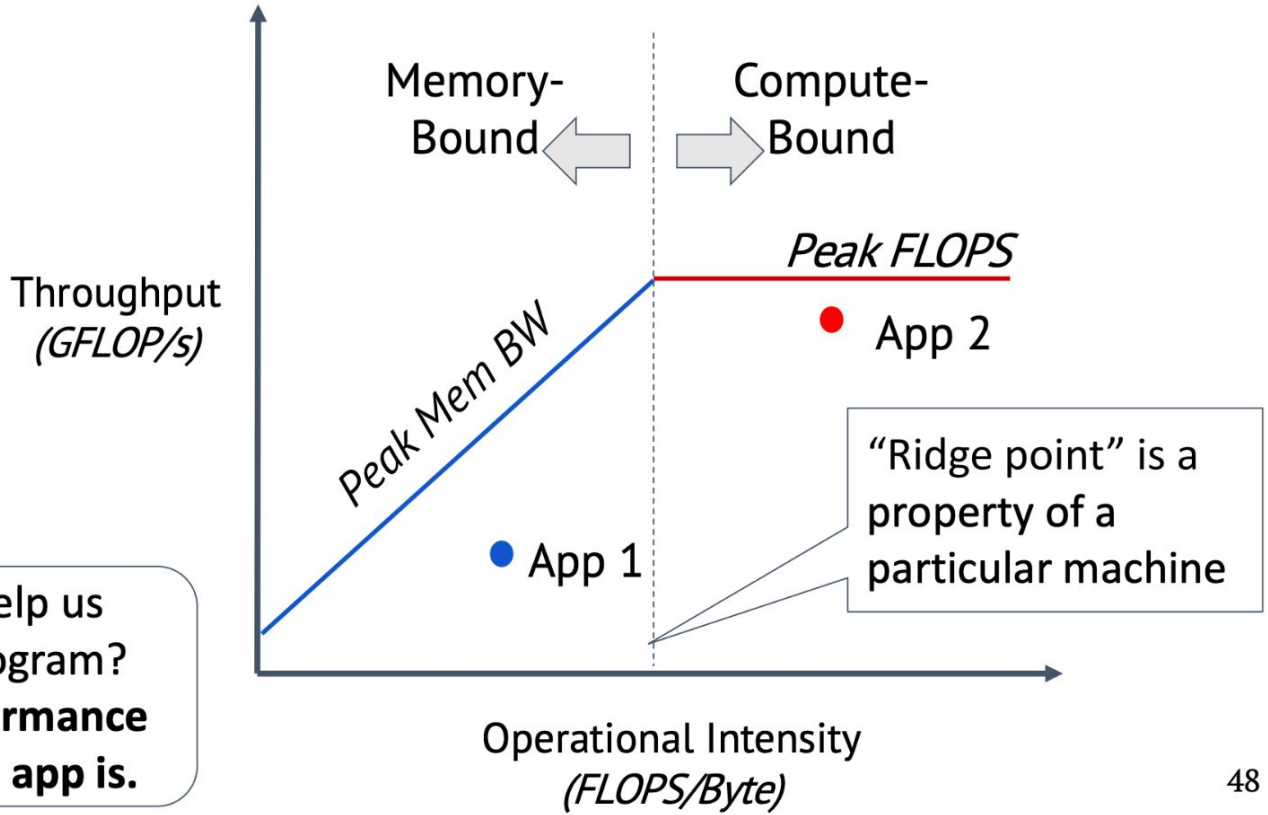
Problem: Sparse representations make processing large graphs feasible, but graph processing still entails a large working set with poor locality

Cache miss latency *cannot be hidden by anything else in the program*. Each miss incurs DRAM latency!

Right Figure from "Optimizing Cache Performance for Graph Analytics" ArXiv v1;

The Roofline Model



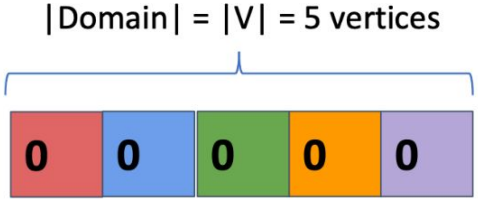


What does Roofline help us understand about a program?
Tell us what limits performance & how close to peak an app is.

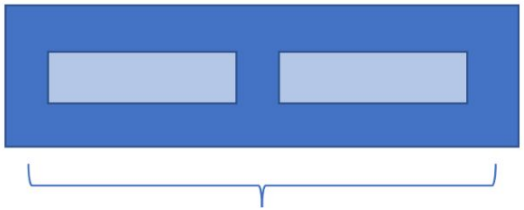
Propagation Blocking: Reorganize Input to Make Memory Being Randomly Written Fit in Cache

0	1
2	0
1	0
0	2
2	3
0	4
0	3

COO
(EdgeList)



Bad for the cache: the size of the *domain* of vertex data array entries is |V|, but the cache holds only |C| << |V| entries

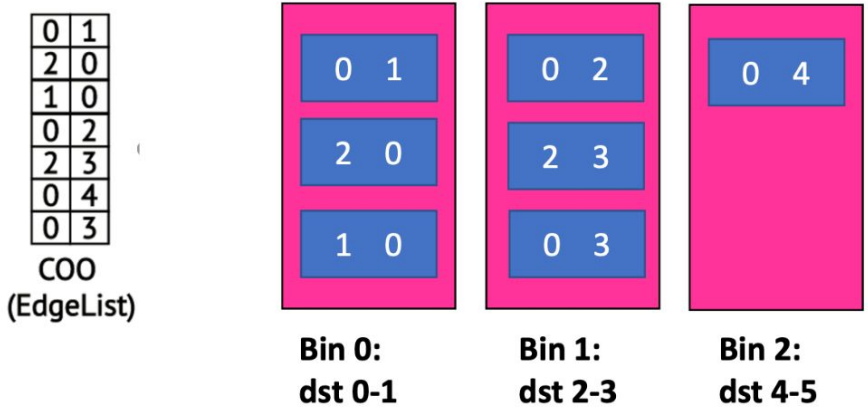


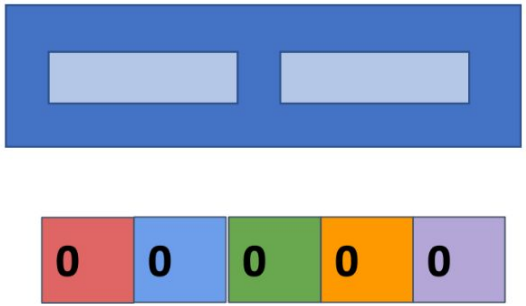
Recall: irregular accesses into vertex data array based on e.dst *which are essentially random*

Key idea in propagation blocking: Limit the domain of updates to a *sub-space* of vertices, **V***, so that |V*| <= |C| and do multiple sub-spaces of V*s, so that all V*s together = V

Propagation Blocking: Reorganize Input to Make Memory Being Randomly Written Fit in Cache

Create "Bins" that hold input elements (edges from the edge list)

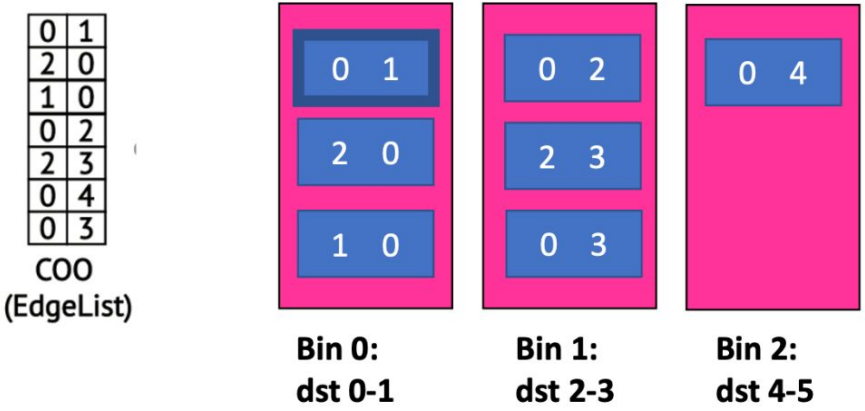




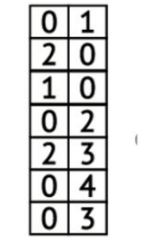
Remember: dstData[e.dst] ++
and e.dst is random, from edge list

Execute the kernel for one bin at a time

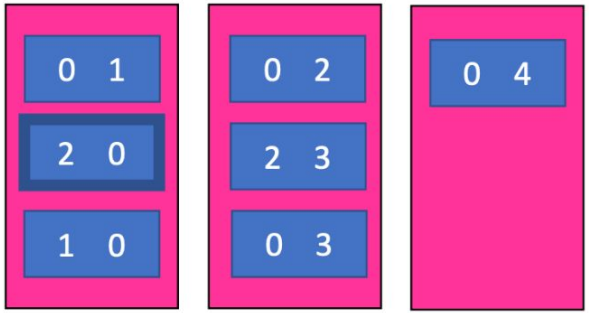
Propagation Blocking: Reorganize Input to Make Memory Being Randomly Written Fit in Cache



Propagation Blocking: Reorganize Input to Make Memory Being Randomly Written Fit in Cache



COO
(EdgeList)

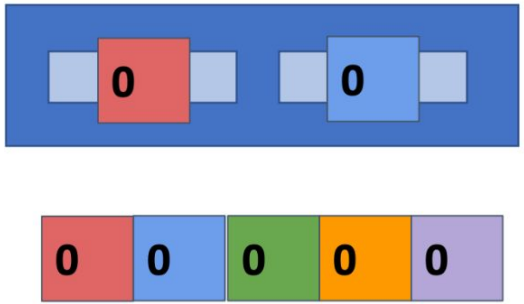


Bin 0:
dst 0-1

Bin 1:
dst 2-3

Bin 2:
dst 4-5

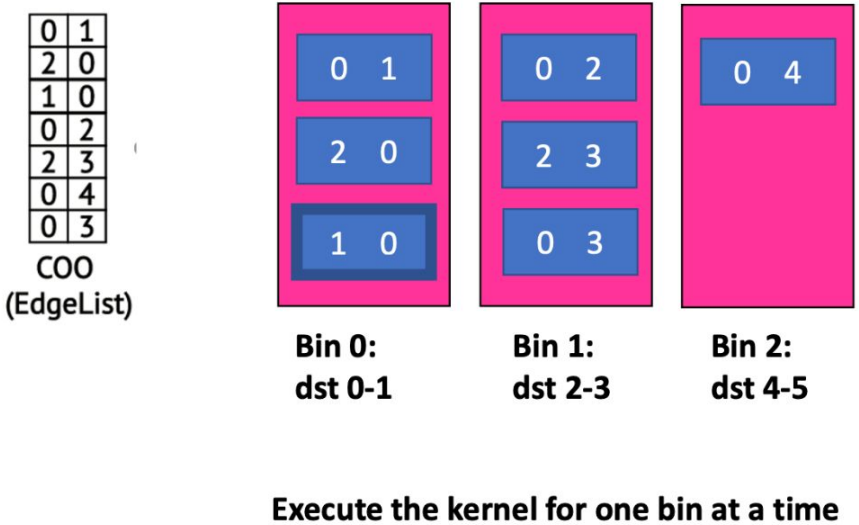
Execute the kernel for one bin at a time



dstData

Remember: dstData[e.dst] ++
and e.dst is random, from edge list

Propagation Blocking: Reorganize Input to Make Memory Being Randomly Written Fit in Cache



Propagation Blocking: Reorganize Input to Make Memory Being Randomly Written Fit in Cache