Introduction to Scalable Systems

Optimisation for Cache

→ Array merging

```
struct { double A, B; } arr[2048];

for (i=0; i<2048; i++)
    sum += arr[A] * arr[B]
```

Data Structures & Algorithms

→ Saving storage for sparse matrices

- → Each non-zero element is added as a tuple in a list.
- → This tuple contains the row, coln., & value.
- → Storing row & coln. overheads is better than storing all 0s.

→ Compressed Sparse Row (CSR) representation

- → Store as three arrays :
  - → $V[nnz]$: non-zero values in row-major order
  - → $C[nnz]$: coln. offset within row group for a non-zero value.

no. of non-zero elements → $R[M+1]$: stores cumulative count of non-zero elements till $(i-1)^{th}$ row

dimension of the matrix

- → Relatively cache friendly

Hashing Analysis

→ Using chain hashing : Expected no. of keys in a successful search

$$= \frac{1}{n} \sum_{j=1}^{n} \left[1 + E\left(\sum_{j=i+1}^{n} X_{ij} \right) \right]$$

$$= 1 + \frac{n-1}{2b}$$

no. of elements stored

$$= O(1+\alpha), \text{ where } \alpha = n/b$$

no. of blocks

B-Trees

→ Depth of the tree = $\left\lceil \log_{\lfloor \frac{m}{2} \rfloor} (\text{size}) \right\rceil + 1$

Graphs

→ Diameter = Distance of the longest shortest path!

$= \max (d(u,v)) \ \forall u, v \in V, u \neq v$

→ Clique : A subgraph (which is a connected graph

→ Maximal clique : Clique with most no. of nodes possible.

→ Graph traversals

→ BFS

→ Start at source vertex & use a queue

→ $O(|V|)$ for adjacency matrix

better for dense graphs due to cache locality

edge degree

→ $O(d)$ for " list

better for sparse graph

→ Total time

w = no. of vertices in connected component

→ $O(w|V|)$ for adjacency list

→ $O(w+E)$ " " matrix

no. of edges in connected component.

→ DFS (Use stack instead of queue)

→ Requires $O(n)$ space (for recursion stack)

→ Same TC as BFS

Algorithms

→ Dijkstra's algorithm

→

Method of storage	TC
Array of weights	$O(\|V\|^2 + \|E\|)$
Using min. heap to store edge weights	$O((\|V\| + \|E\|)\log\|V\|)$
Using Fibonacci heap	$O(\|E\| + \|V\|\log\|V\|)$

worse than array implementation when $|E| >> |V|$, i.e for an almost connected graph