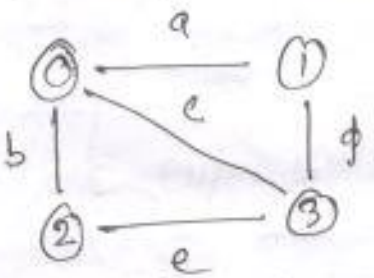
MIN_CUT ~~THEOR~~ ALGORITHM

→ Given undirected graph find the smallest cut (smallest no. of edges that disconnects the graph into two components.

→ The input graph may have the ~~parallel~~ par edges.

→ For example consider the following example, the smallest cut has two edges.



→ Min_Cut for above graph is either {a, d} OR {b, e}.

→ A simple solution use Max-flow-based s-t cut algorithm to find minimum cut. consider every pair of vertices as Source 's' and Sink 't' and call minimum s-t cut algorithm to find the s-t cut return minimum of all s-t cuts. Best time complexity of this algorithm is $O(V^5)$ for a graph.

→ Karger's algorithm for this purpose. Below Karger's algorithm works in $O(E) = O(V^2)$ time.

(1) Initialize connected graph Cg as copy of original graph.

(2) While there are more than 2 vertices

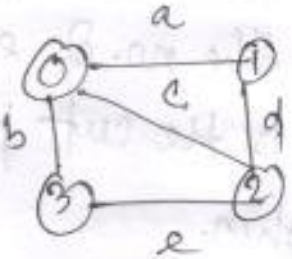
(a) Pick a random edge (u, v) in the connected graph.

(b) Merge u, v into a single vertex (update the graph).

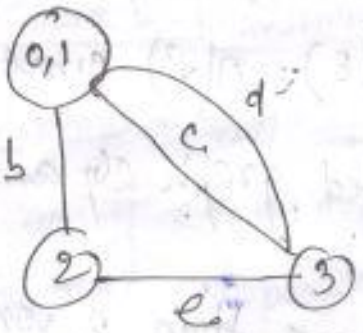
(c) Remove self loops.

(3) Return cut represented by two vertices.

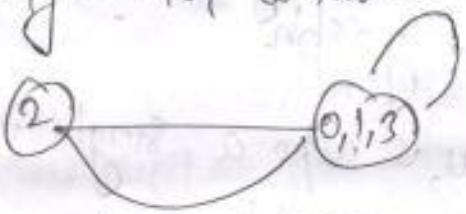
Lets understand the above algorithm



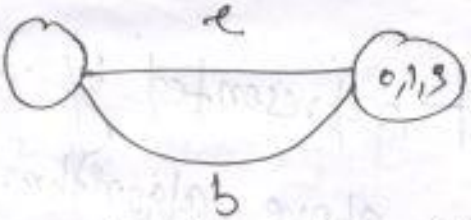
→ Let the first randomly picked vertex be 'a' which connects vertices 0 and 1. We remove this edge and contract the graph (combine 0 and 1).



→ Let the next randomly edge be d. We remove this edge and combine vertices (0,1) and 3



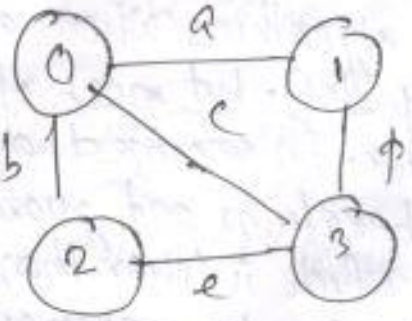
→ We need to remove self loop



→ Now the graph has two vertices, so the algorithm stops. The no. of edges in the resultant graph is the cut produce by Karger's algorithm.

→ Karger's algorithm is a Monte Carlo algo and cut produced by it may not be minim

For example pick the edge in following order



pick edge b remove it and fuse the two vertices into one

Pick edge 'd' remove it and fuse two vertices into one.

The cut produced by Kargers algorithm is not minimum.

Amortized Analysis

→ Amortize analysis is used for algorithms where an occasional operation is very slow, but most of the other operation are faster. In amortized analysis we analyze a sequence of operations and guarantee a worst case average time which is lower than the worst case time of a particular expensive operation.

→ The example data structures whose operations are analyzed using Amortize analysis are Hash Tables, Disjoint set, and splay Trees.

→ Lets take an example of hash table insertions. How do we decide table size? There is a trade off between space and time, if we make hash table size big, search time becomes fast, but space required becomes high.

→ The solution to this trade-off problem is to use Dynamic Table. The idea is to increase size of table whenever it becomes full. Following are the steps to follow when table becomes full.

① Allocate memory of a larger table of size, typically twice the old table.

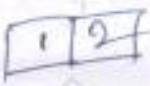
② Copy the contents of old table to new table.

③ Free the old table.

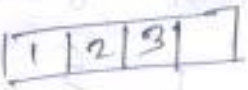
Initially table is empty and size is 0.

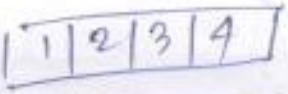
Insert item 1. 

(OF)

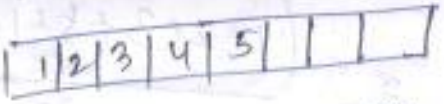
" " 2. 

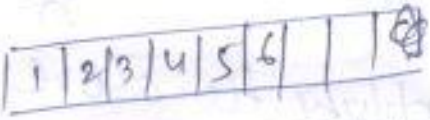
(OF)

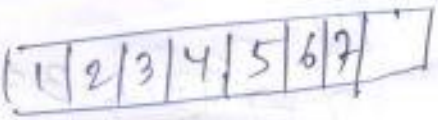
" " 3. 

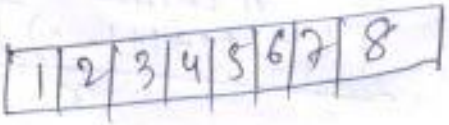
" " 4. 

(OF)

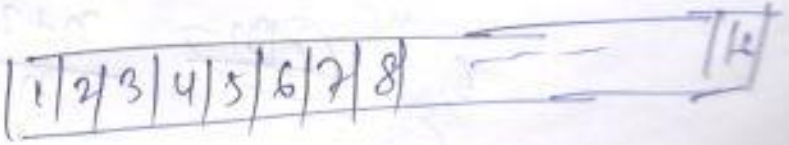
" " 5. 

" " 6. 

" " 7. 

" " 8 

(OF)

" " 9 

Time Complexity

→ The worst case time insertion time is $O(n)$. so to insert n element it's time complexity is $n * O(n) = O(n^2)$.

→ This analysis gives upperbound but not a tight upper bound for n insertions as all insertions don't take $\Theta(n)$ time.

item No	1	2	3	4	5	6	7	8	9
Table size	1	2	4	4	8	8	8	8	16
cost	1	2	3	1	5	1	1	1	9

Amortize cost $= \frac{(1+2+3+5+1+1+9+1\ldots)}{n}$

We can simplify the above series by grouping terms.

$2 \rightarrow (1+1)$, $3 = (1+2)$, $5, 9 \ldots$

n items, $\lfloor \log_2(n-1) \rfloor + 2$

$= \frac{[(1+1+1+1+1\ldots)] + \{1+2+4+\ldots\}}{n}$

$\leq \frac{n+2n}{n}$

≤ 3

$O(1)$

⇒ cost is other than one only if item is of form $2^{(x-1)}+1$ where $x=1,2,3,\ldots$

→ so if no. of items is n, then

$$n = 2^{x-1}+1$$

$$\Rightarrow (n-1) = 2^{x-1}$$

$$\Rightarrow \log_2(n-1) = x-1$$

$$\Rightarrow \boxed{x = \log_2(n-1)+1}$$

So $1+2+4+8 \ldots + \log_2(n-1)$

is a G.P. with $r=2$

$$= \left(\frac{r^n-1}{r-1}\right) : \log_2(n-1)+1$$

$$= \frac{2^{\log_2(n-1)+1}-1}{} =$$

$$= \left(2^{\log(n-1)} \cdot 2\right) - 1$$

$$= (n-1)*2 - 1$$

$$= 2n-3$$

$$\therefore \leq n$$

$\frac{2}{1} \quad \frac{4}{2} \quad \frac{8}{4}$

$\frac{a(r^n-1)}{r-1}$

$= 1+2$

$\log^{a+1} = \log^a \cdot \log$