Minimal Perfect Hash Functions Based on Random Graphs Containing Cycles

Fabiano C. Botelho
Nivio Ziviani

Department of Computer Science
Federal University of Minas Gerais, Brazil

International Conference on the Analysis of Algorithms
Maresias, Brazil, April 14, 2008

Objective of the Presentation

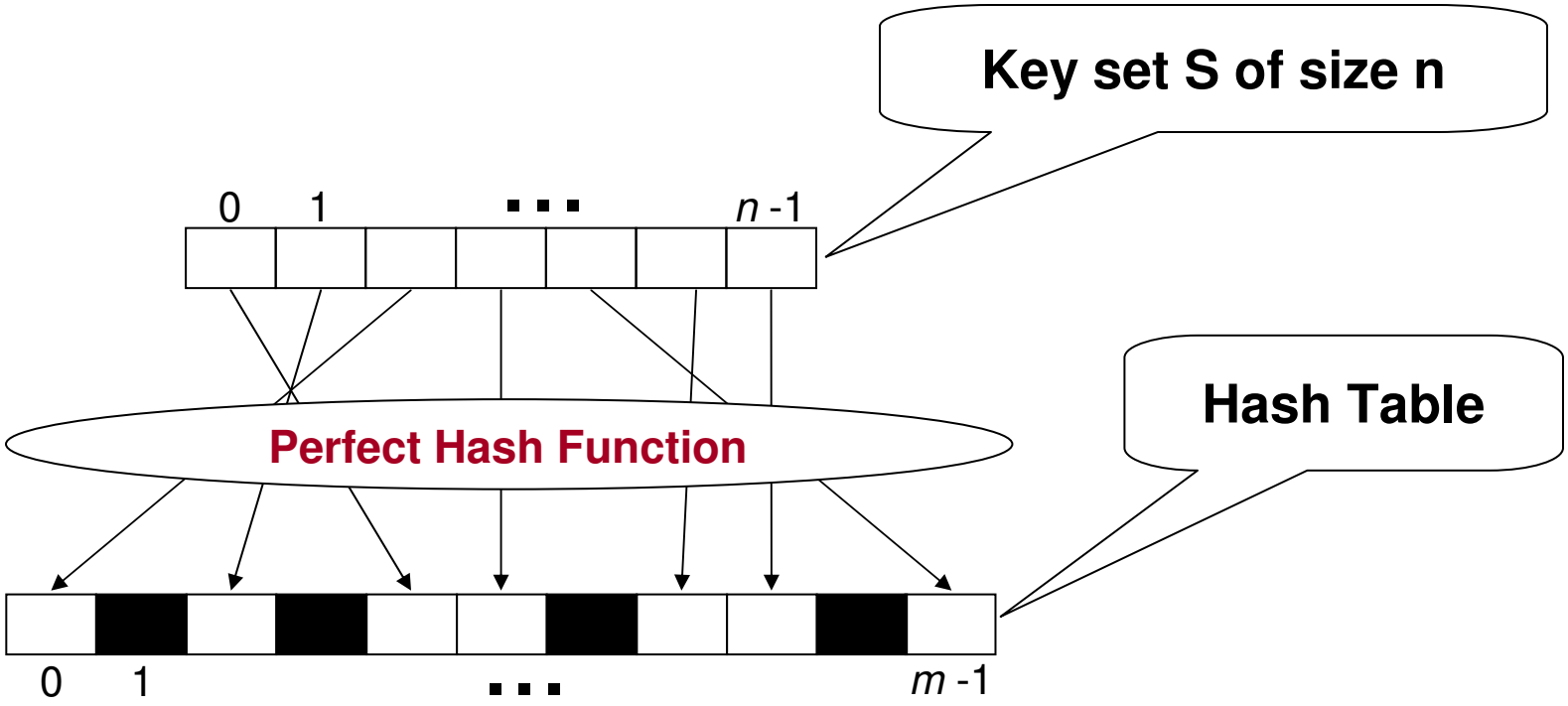
Show how to generate minimal perfect hash functions based on random graphs containing cycles

Some Results

Algorithms	Generation Time(sec)	Storage Space (bits/key)
Czech, Havas and Majewski (IPL, 92)	17.03 ± 3.24	45.47
Botelho, Kohayakawa and Ziviani (WEA,05)	12.83 ± 1.58	21.76
Botelho, Pagh and Ziviani (WADS, 07)	15.04 ± 2.80	3.3
Botelho, Pagh and Ziviani (to be submitted)	10.92 ± 1.18	3.3

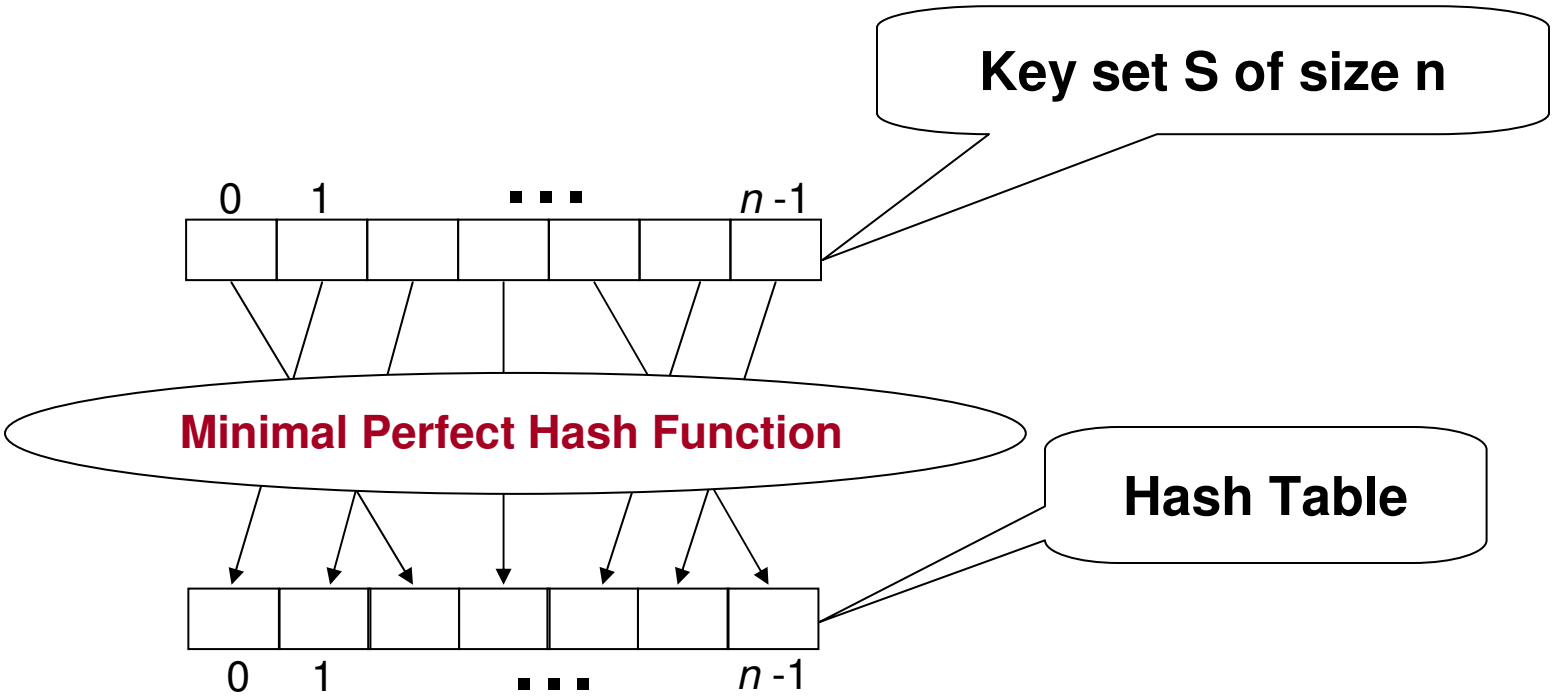
A set of 3,541,615 URLs

Perfect Hash Function



$$S \subseteq U, \text{where} |U| = u$$

Minimal Perfect Hash Function



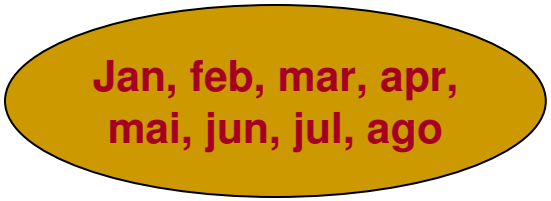
$$S \subseteq U, \text{where} |U| = u$$

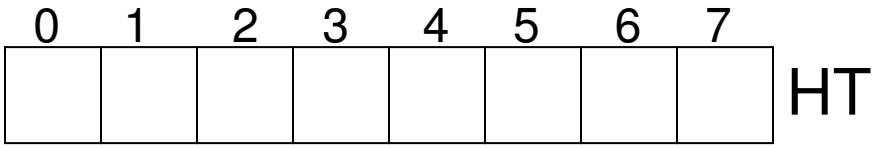
The Algorithm

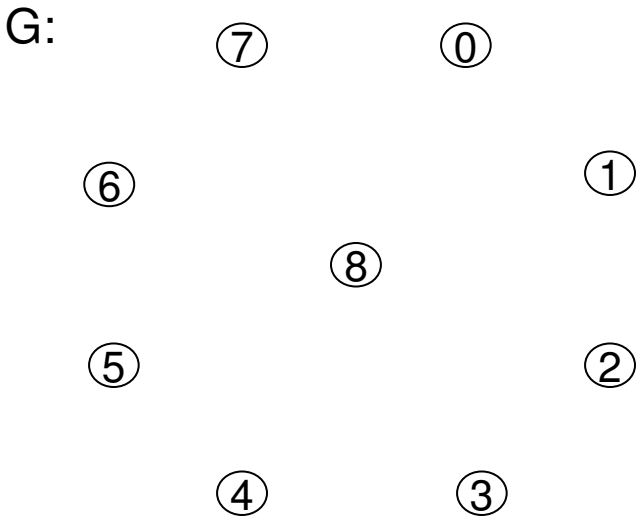
Botelho, Kohayakawa and Ziviani (WEA 2005):

- Improves space requirement of the algorithm by Czech, Havas and Majewski (IPL1992).
- Generates functions stored in O(n log n) bits
- **Is not completely analyzed**

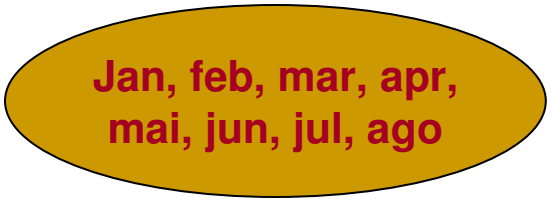
First Step - Mapping

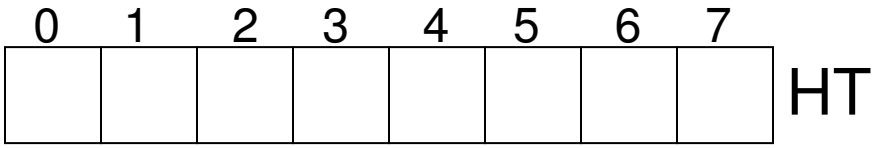




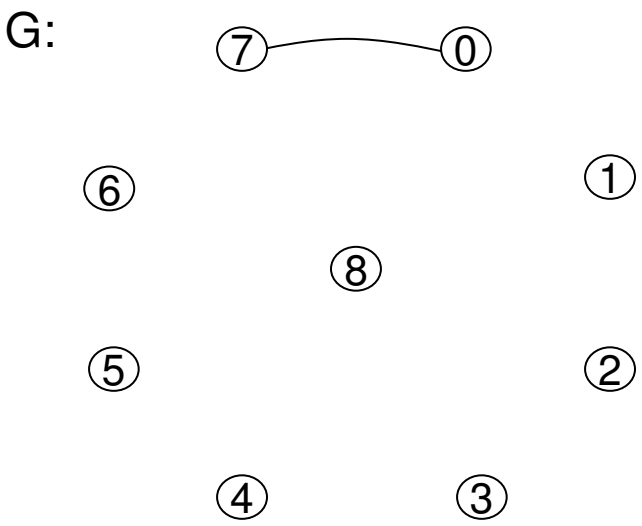


First Step - Mapping

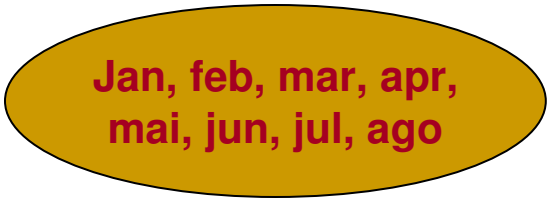


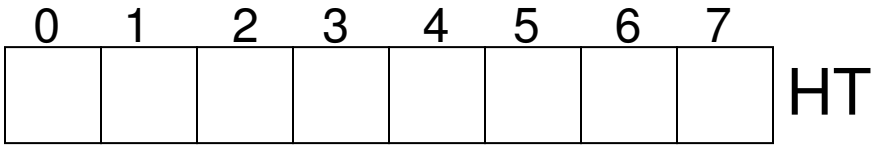


h_1(**jan**) = 7 h_2(**jan**) = 0



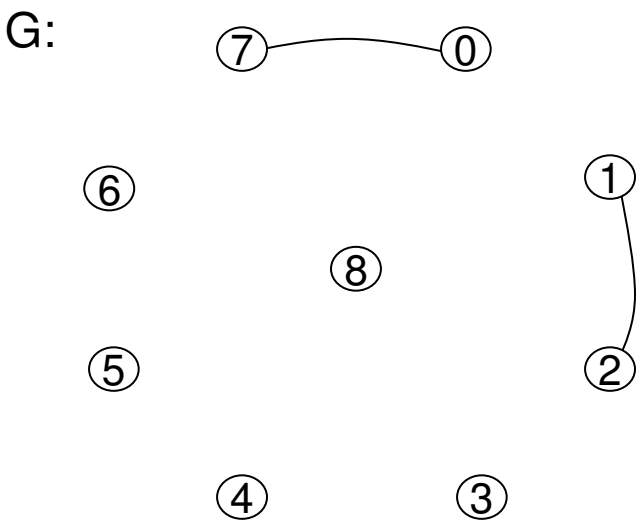
First Step - Mapping





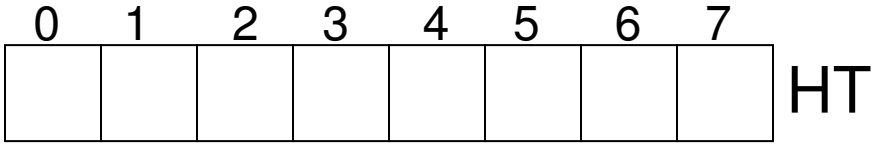
h_1(**jan**) = 7 h_2(**jan**) = 0

h_1(**feb**) = 1 h_2(**feb**) = 2



First Step - Mapping

Jan, feb, mar, apr, mai, jun, jul, ago



G must be simple and must have at most 50% of edges in cycles.

h_1(**jan**) = 7 h_2(**jan**) = 0

h_1(**feb**) = 1 h_2(**feb**) = 2

h_1(**mar**) = 8 h_2(**mar**) = 1

h_1(**apr**) = 3 h_2(**apr**) = 4

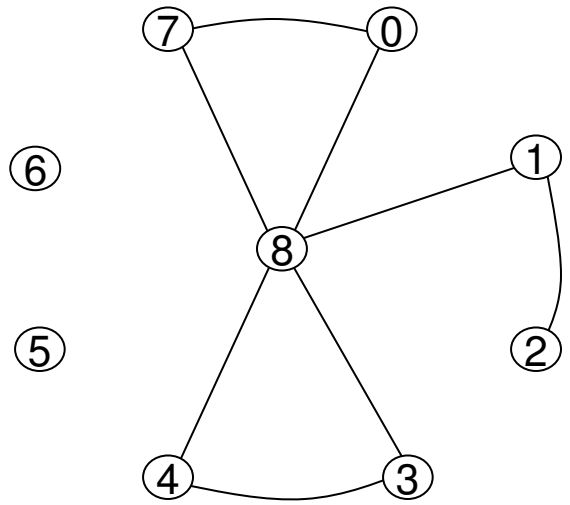
h_1(**mai**) = 4 h_2(**mai**) = 8

h_1(**jun**) = 8 h_2(**jun**) = 0

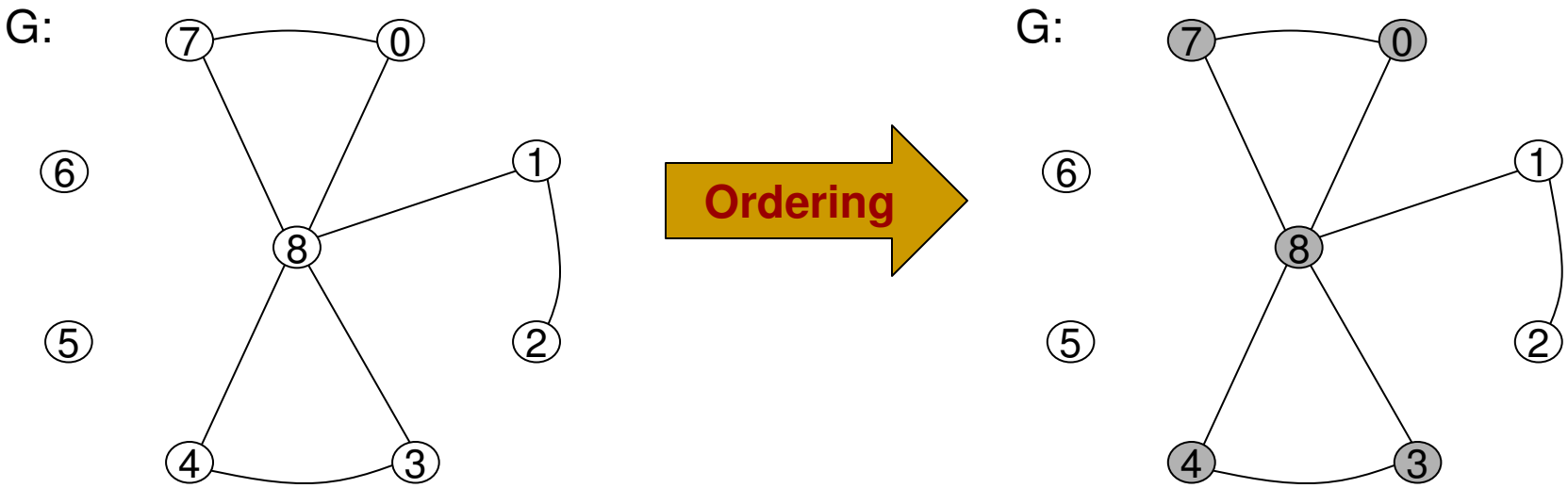
h_1(**jul**) = 3 h_2(**jul**) = 8

h_1(**ago**) = 7 h_2(**ago**) = 8

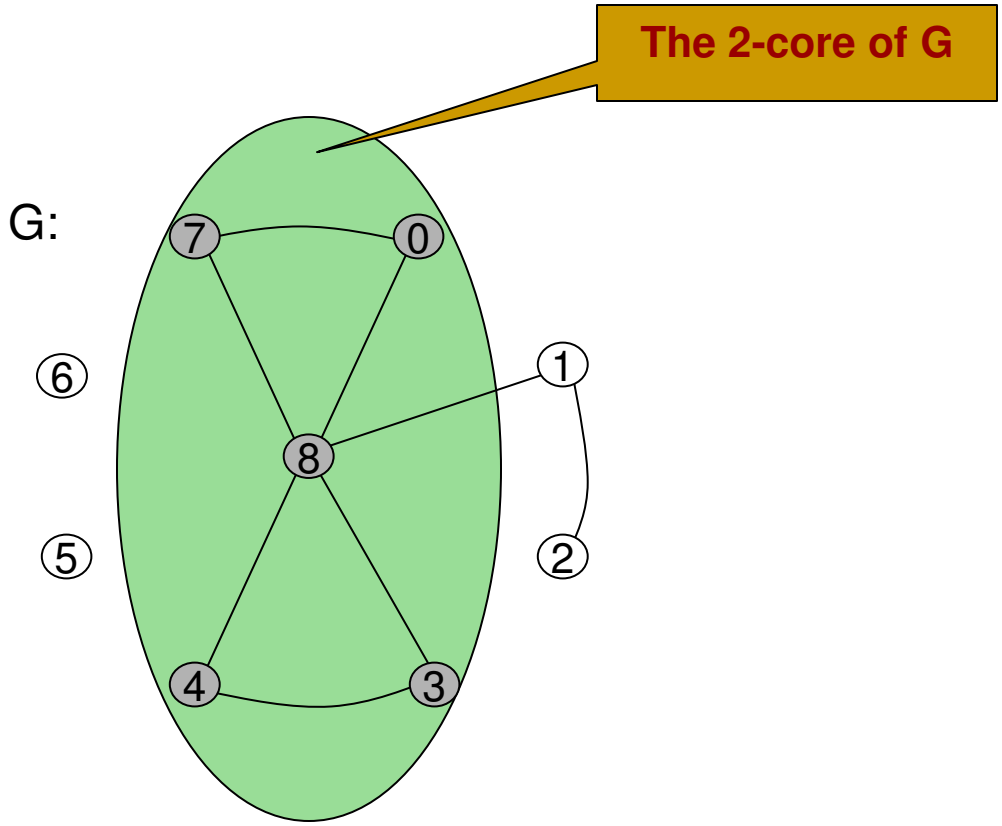
G:



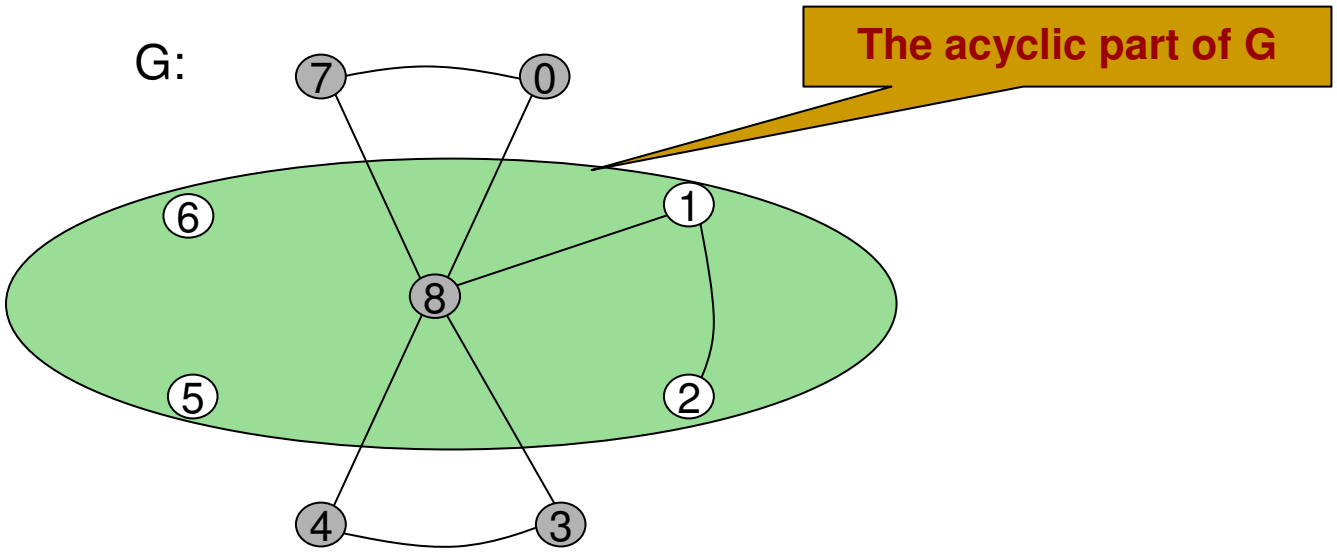
Second Step - Ordering



Second Step - Ordering



Second Step - Ordering



Third Step - Searching

Assign values to the vertices such that:

$$h(e) = g(a) + g(b)$$

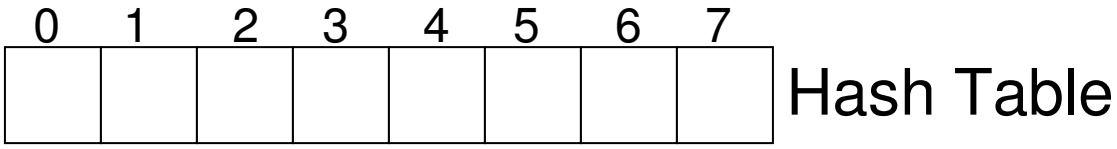
MPHF

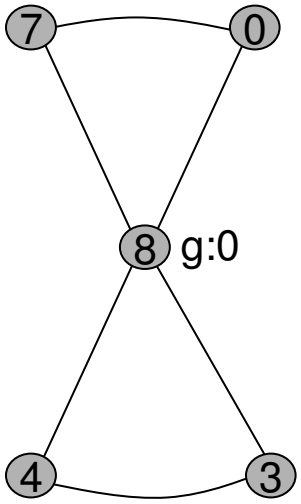
Where:

1. e = {a, b}
2. a = $h_1(x)$
3. b = $h_2(x)$
4. x is in a key of S

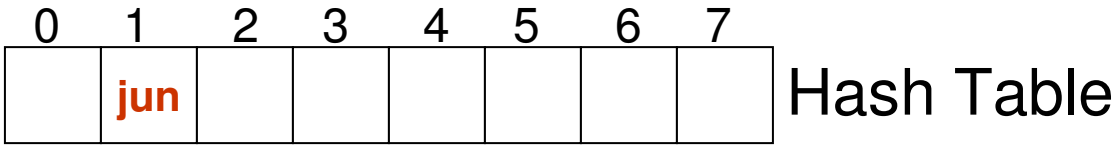
First: Critical part (2-core)
Second: Non critical part (acyclic part)

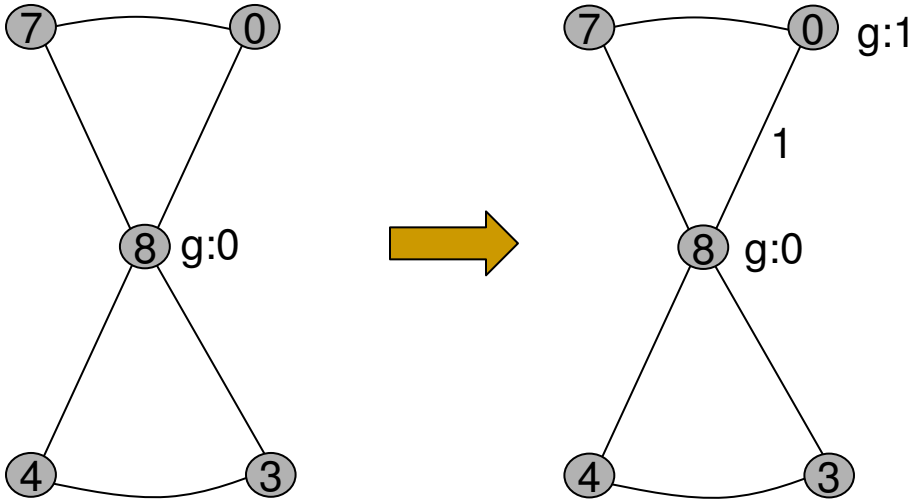
Assignment of Values to Critical Vertices



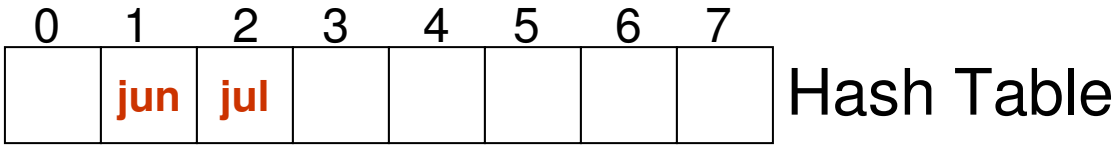


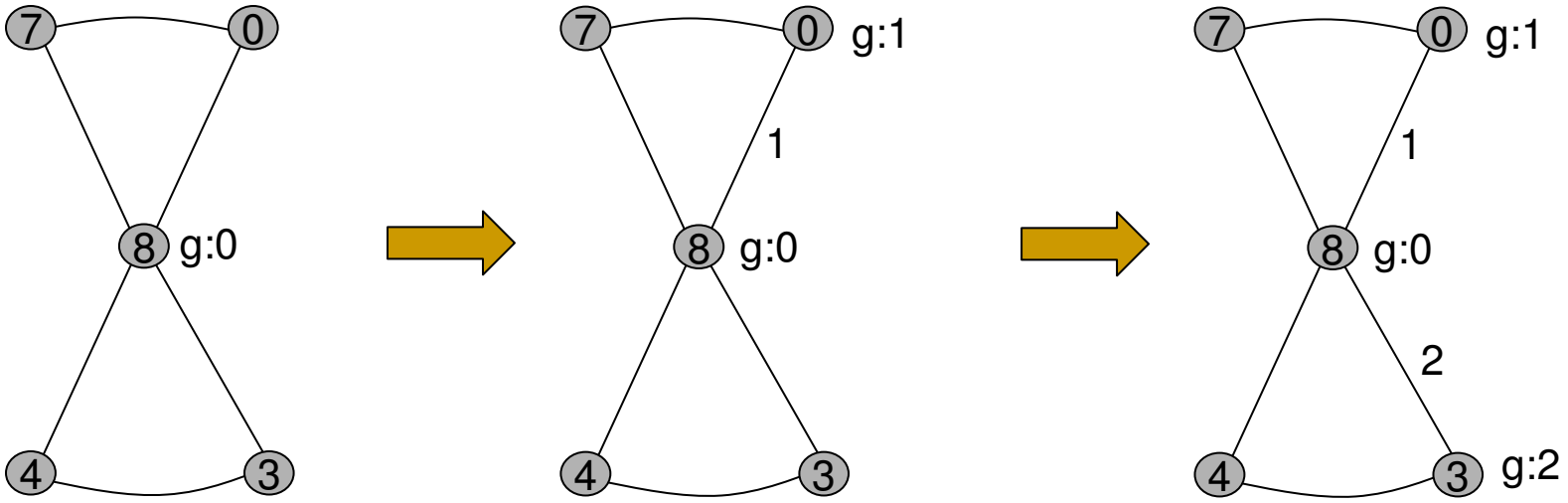
Assignment of Values to Critical Vertices





Assignment of Values to Critical Vertices

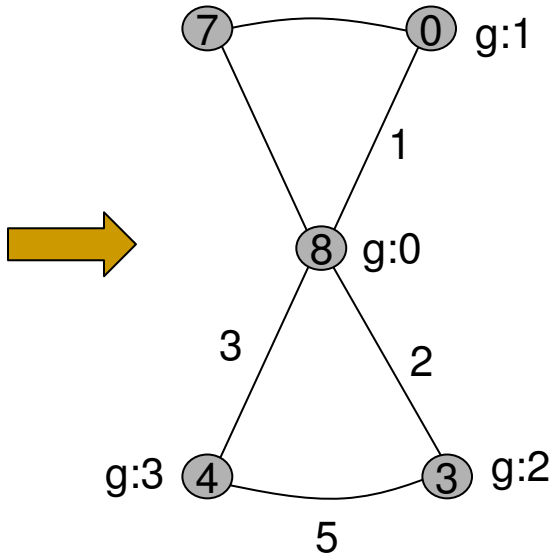




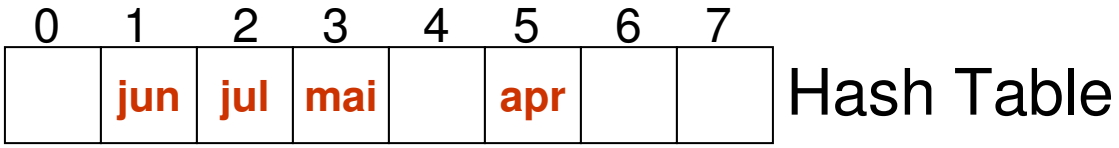
Assignment of Values to Critical Vertices

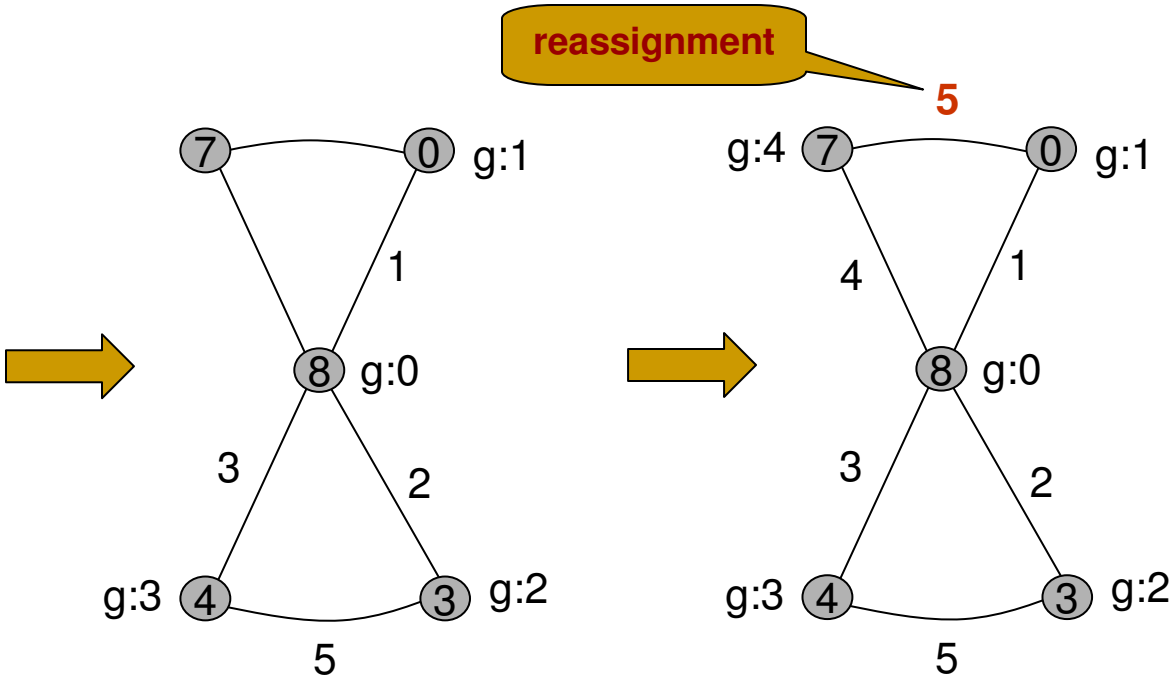
0	1	2	3	4	5	6	7
	jun	jul	mai		apr		

Hash Table

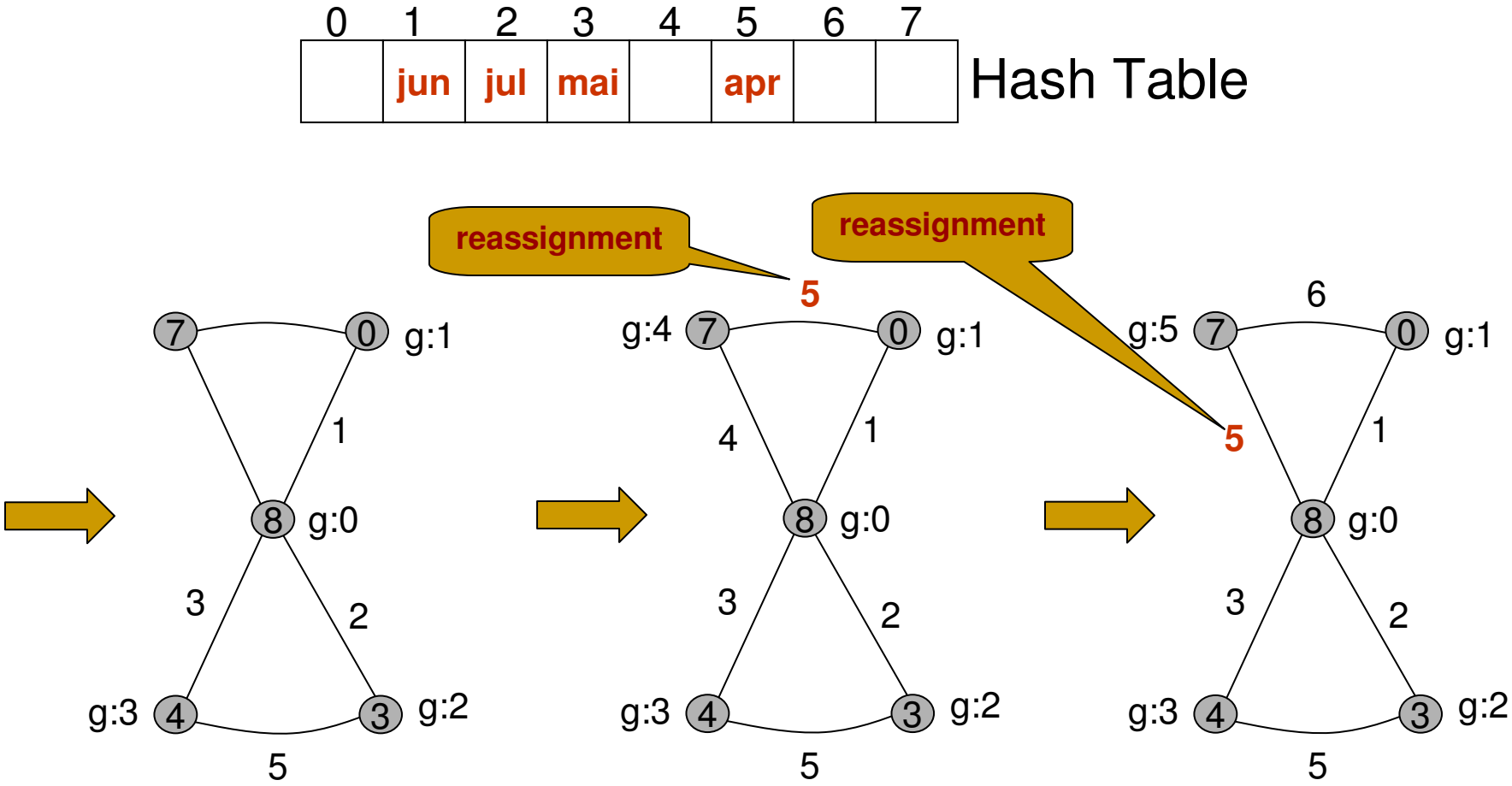


Assignment of Values to Critical Vertices





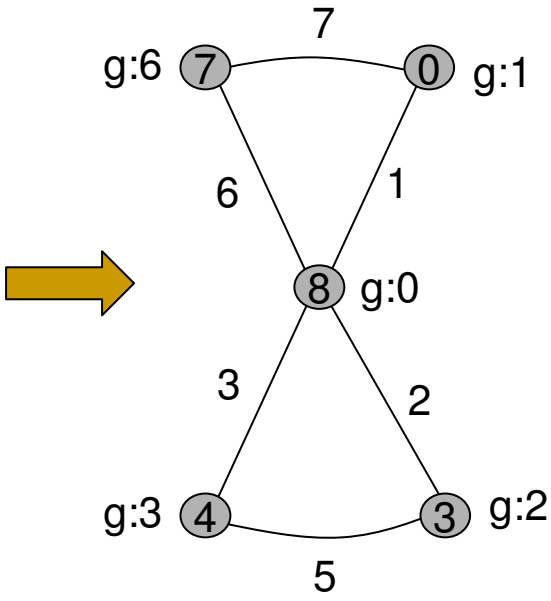
Assignment of Values to Critical Vertices



Assignment of Values to Critical Vertices

0	1	2	3	4	5	6	7
	jun	jul	mai		apr	ago	jan

Hash Table



Used addresses: {1,2,3,5,6,7}

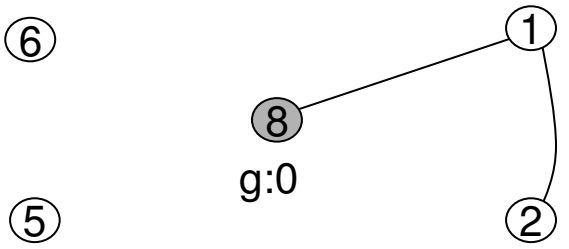
Unused addresses: {0,4}

Assignment of Values to Non-Critical Vertices

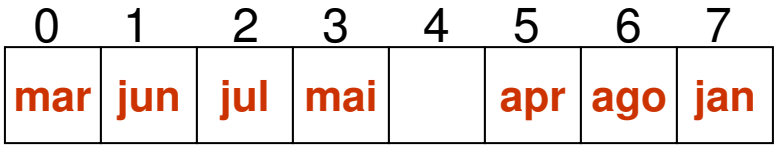
0	1	2	3	4	5	6	7
	jun	jul	mai		apr	ago	jan

Hash Table

Unused addresses: {0,4}

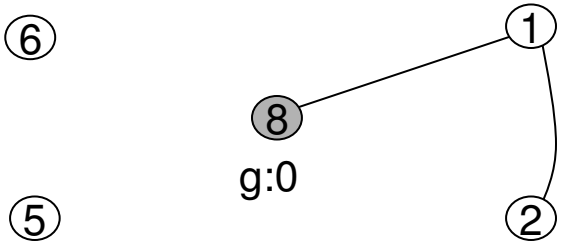


Assignment of Values to Non-Critical Vertices

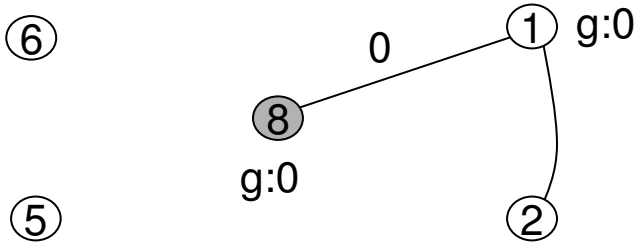


Hash Table

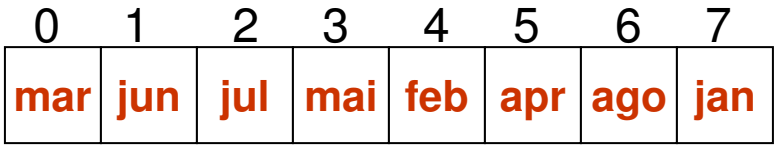
Unused addresses: {0,4}



Unused addresses: {4}

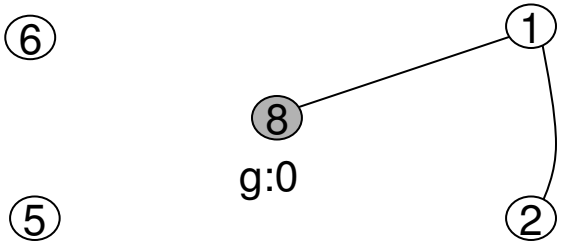


Assignment of Values to Non-Critical Vertices

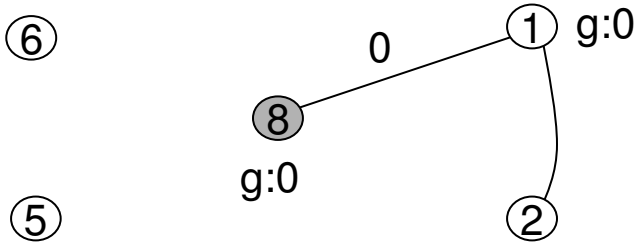


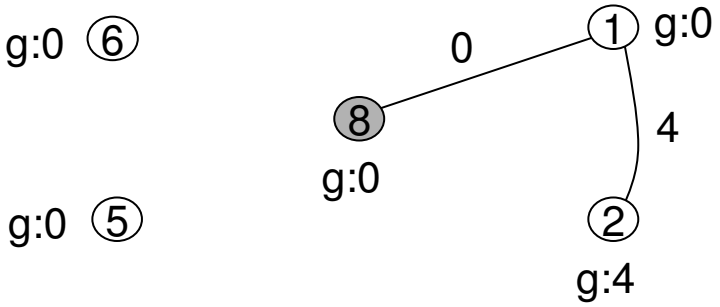
Hash Table

Unused addresses: {0,4}



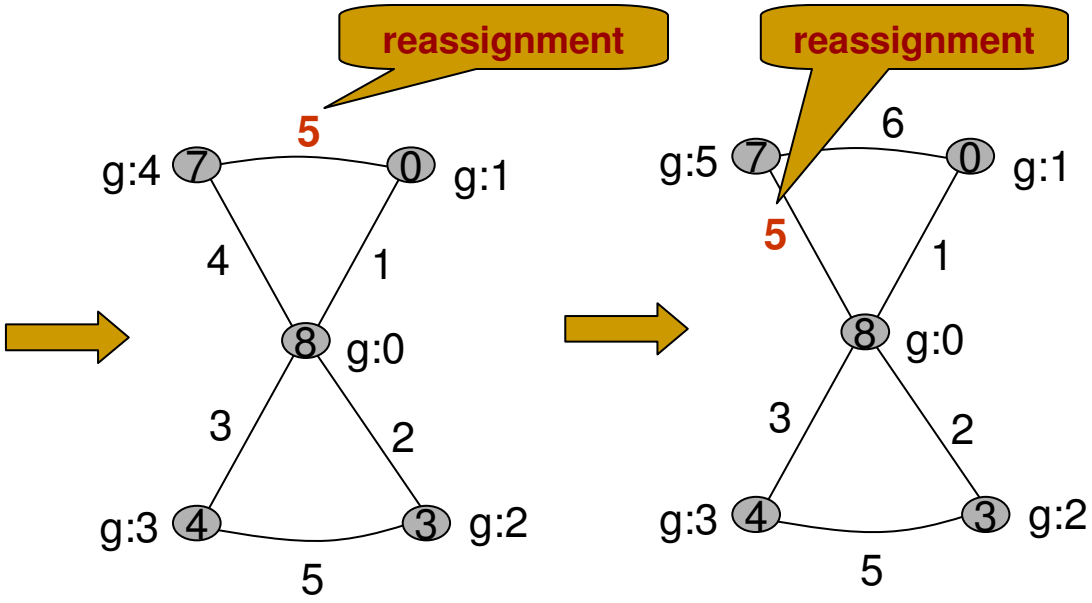
Unused addresses: {4}





Unused addresses: {}

The Open Problem



- The maximal value assigned to an edge is:

$$A_{max} \leq 2 | V_{crit} | - 3 + 2N_t$$

- The number of back edges of G is:

$$N_{bedges} = |E_{crit}| - |V_{crit}| + 1$$

The Open Problem

- Conjecture: $N_t \leq N_{bedges}$

$$A_{max} \leq 2|V_{crit}| - 3 + 2N_{bedges}$$

$$A_{max} \leq 2|V_{crit}| - 3 + 2(|E_{crit}| - |V_{crit}| + 1)$$

$$A_{max} \leq 2|E_{crit}| - 1$$

- If $|E_{crit}| \leq 0.5n$ then $A_{max} \leq n - 1$ and a MPHF is generated in linear time.

- The only problem is left open is:

Prove that $N_t \leq N_{bedges}$

Experimental Evidences

- Experimental evidences that $N_t \leq N_{bedges}$:
- Recall: $N_{bedges} = |E_{crit}| - |V_{crit}| + 1 = 0.501n - 0.401n + 1 = 0.1n + 1$.

n	Expected N_{bedges}	Maximal Value of N_t
10,000	0.1n + 1	0.067n
100,000	0.1n + 1	0.061n
1,000,000	0.1n + 1	0.059n
2,000,000	0.1n + 1	0.059n
...	...	...

Approach Used for Constructing Minimal Perfect Hash Functions

- MOS - Mapping, Ordering and Searching:
  - **Mapping**: transforms the key set from the original universe to a new universe.
  - **Ordering**: places the keys in a sequential order that determines the order in which hash values are assigned to keys.
  - **Searching**: attempts to assign hash values to the keys.

At Most 50% of Edges in the 2-Core

- The number of vertices is a linear function on the number of edges: |V(G)|=cn
- Pittel and Wormald (2005), present detailed results for the 2-core of the giant component of a random graph G.

$\|V_{crit}\|$	$\|E_{crit}\|$	c
0.399n	0.498n	1.153
0.400n	0.500n	1.152
0.401n	0.501n	1.151
0.401n	0.501n	1.150
0.401n	0.502n	1.149

- We determined empirically that c = 1.15.