Perfect hashing using graph theory

Jan Kaštil

TID 12.11. 2008

Motivation

- Dictionary problem
  - Old and important
  - How to efficiently access data in dictionary
  - Sorted lists, hash tables , etc.
- Hash Table
  - Collision solving
  - Memory overhead

Hash function

- Lets U be universe of all possible keys
- Let S be set of all actual keys
- Let M be output interval:

$$S \subset U$$
$$|S| \ll |U|$$
$$|M| \geqslant |S| \; yet \; |M| \ll |U|$$

- Hash function h is mapping: U → M
- Synonyms: $h(x_1) = h(x_2) \wedge x_1 \neq x_2$
- Collision for two synonyms: $x_1 \in S \wedge x_2 \in S$

Good hash function

- Uniformly distributed output
  - Birthday paradox

$$|S| = 1,25\sqrt{|M|}$$

Then probability of collision is about 50%

- Fast computation
- Application specific need
  - Cryptographic properties
  - Order preserving

Perfect hash

- Lets have set K_m such that

$$m \in M$$
$$K_m = \{x \mid x \in S \wedge h(x) = m\}$$

than hash function is l-perfect, if

$$\forall m \in M : |K_m| \leqslant l$$

- Perfect Hash is 1-perfect hash
- Minimal Perfect Hash is Perfect Hash with

$$|M| = |S|$$

Needed Graph Theory

- Graph is ordered pair G(V,E) where V is set of vertices and E is set of edges.
  - Edge e is n-tuple $e=(v_1, v_2, \ldots, v_n) \wedge \forall i \leqslant n : v_i \in V$
  - Edge of graph is couple
  - Edge of hypergraph (r-graph) is r-tuple
- Degree of Vertex $Deg(V) = \left| \left\{ x \middle| x \in E \wedge \exists i : x_i = V \right\} \right|$
- Random graph is Graph, created from graph with empty set of edges by adding new edges connecting randomly chosen vertices.

Needed Graph Theory 2

- Loop in graph is the edge e:

$$\exists i \leqslant r \wedge \exists j \leqslant r : v_i = v_j$$

- Acyclic graph is loopless graph containing edge with at least one vertices with degree 1 and after its removing, the remaining graph is still acyclic. Graph without edges is acyclic.
- Probability that random graph or hypergraph is acyclic depends on the rate between |V| and |E|.

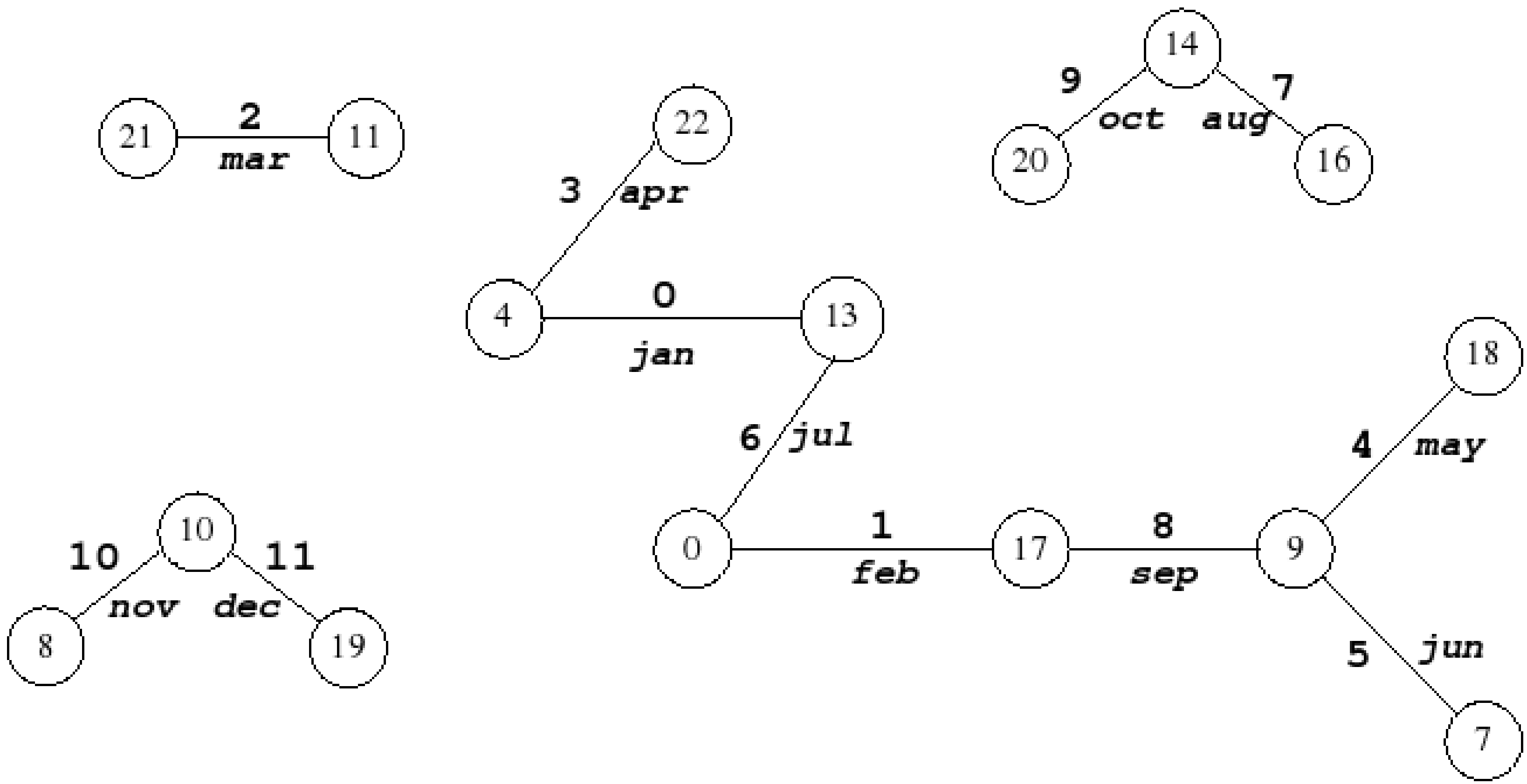
Hash functions & Random graphs

- Lets have r **randomly** chosen hash functions h_i and set of the keys S.
  - Create the set V $|V|=|M|$
  - Set of edges $E=\{(h_1(x), h_2(x), \ldots, h_r(x)) | \forall x \in S\}$
- Such graph is random because hash functions are random

Graph and Information

- We can create a mapping from E to any interval just by storing pairs <Edge, Number>
  - This mapping could be perfect hash, but
  - How efficiently found the right edge?
- Add information to the vertices
  - Fast to retrieve
- Acyclic graph can have simple function (plus) to compute label of edge from the labeled vertices

Illustration



	0	4	7	8	9	10	11	13	14	16	17	18	19	20	21	22
g	0	6	10	0	7	10	0	6	0	7	1	9	1	9	2	9

Algorithm CHM for Perfect Hashing

- Presented by Czech, Havas and Majevski
- Uses two random hash functions
- Random graph formed by hashes and set S
  - Has to be acyclic graph (loopless, not multigraph)
- If random graph doesn't meet criteria, hashes are generated again.
- To found graph in sufficiently small number of steps, graph should have $|V| \geqslant 2{,}09|E|$
- CHM is order preserving
- CHM requires about $|S|\log(|S|)$ bits

BMZ algorithm

- Why graph has to be acyclic?
  - For fast computation of values of vertices
  - To generate order preserving hash function
- BMZ does not require acyclic graph
  - Maximal half of the graph can be in the cycles
  - “right” graph can be found in reasonable time:

$$|V| \geqslant 0{,}93|E|$$

- Only difference against CHM is in labeling vertices in the cycles

BDZ algorithm

- Uses r hash functions (best results for r = 3)
  - Each hash maps keys into subset of V
    - No loops can exists
- Generated r-graph has to be acyclic
- Values of edges do not represent the output of PHF, but points to the right hash
  - Only linear space requirements

Literature

Z.J. Czech, G. Havas, and B.S. Majewski. Fundamental study perfect hashing. Theoretical Computer Science,182:1–143, 1997.

F.C. Botelho, Y. Kohayakawa, and N. Ziviani. A practical minimal perfect hashing method. In Proc. of the 4th International Workshop on Efficient and Experimental Algorithms (WEA’05), pages 488–500. Springer LNCS vol. 3503, 2005.

F. C. Botelho, R. Pagh, N. Ziviani. Simple and space-efficient minimal perfect hash functions. 10th International Workshop on Algorithms and Data Structures (WADs'07), Springer-Verlag Lecture Notes in Computer Science, vol. 4619, Halifax, Canada, August 2007, 139-150.

Z.J. Czech, G. Havas, and B.S. Majewski. An optimal algorithm for generating minimal perfect hash functions., Information Processing Letters, 43(5):257-264, 1992.