Week 3: Balls and Bins

Jan 29, 2018
CS 5330

Last Time:

- QuickSort
- Average-Case Analysis Stable Marriage
- The Coupon Collector's Problem.

Today: Hashing

- Balls-in-Bins
- Hashing with Chaining
- Linear Probing
- Cuckoo Hashing

Puzzle of the Day
→ 2 envelopes
→ one has 2x money
→ Pick an envelope at random

Should switch?

let A = money in envelope

$E[\text{other envelope}] = \frac{1}{2} \cdot 2A + \frac{1}{2} \cdot \frac{A}{2} = \frac{5}{4} A$

Always switch!

What is wrong?

Balls-and-Bins

Setup: b bins, n balls } place each ball in a bin chosen uniformly at random

Basic Facts

1) Let X_j = # balls in bin j.

Let $X_{ji} = 1$ if ball i is in bin j.

$X_j = \sum_i X_{ji}$

$$E[X_{ji}] = Pr[X_{ji}] = 1/b$$

$$E[X_j] = E[\Sigma X_{ji}] = n/b$$

Ex: if $b = 2n$, $Pr(X_j \geq 1) \leq E[X_j] \leq 1/2$

↑ Markov

2) $$Pr(X_j = 0) = \left(1 - \frac{1}{b}\right)^n \leq e^{-n/b}$$

Ex: if $n = (c+1)b\log b$, then $Pr(X_j = 0) \leq e^{-(c+1)\log b}$

$\leq 1/b^{c+1}$

$$Pr(\text{any bins empty}) \leq \sum_j Pr(X_j = 0) \leq \frac{b}{b^{c+1}} \leq 1/b^c$$

More Balls and Bins

Max load of a bin?

fix k balls; all choose bin j

$$\Pr(\text{bin } j \text{ has} \geq k \text{ balls}) = \binom{n}{k}\left(\frac{1}{n}\right)^k$$

$$\leq \left(\frac{en}{k}\right)^k \left(\frac{1}{n}\right)^k$$

$$\leq \left(\frac{e}{k}\right)^k$$

$$k^* = \frac{6 \log n}{\log \log n}$$

$$\left(\frac{k^*}{e}\right) \geq \frac{2 \log n}{\log \log n} \geq \sqrt{\log n}$$

$$\left(\frac{k^*}{e}\right)^{k^*} \geq \left(\sqrt{\log n}\right)^{\frac{6 \log n}{\log \log n}} \geq 2^{\frac{6 \log n}{\log \log n} \cdot \frac{1}{2} \log \log n}$$

$$\geq 2^{3 \log n} \geq n^3$$

$$\Pr(\text{any bin} \geq k^*) \leq \sum \frac{b}{n^3} \leq \frac{1}{n^2}$$

if $b \leq n$

$$\Pr(\text{bin } j \text{ has} \geq \ell \text{ balls}) = \binom{n}{\ell}\left(\frac{1}{n}\right)^{\ell}\left(1-\frac{1}{n}\right)^{n-\ell}$$

$$\geq \left(\frac{n}{\ell}\right)^{\ell}\left(\frac{1}{n}\right)^{\ell}\left(1-\frac{1}{n}\right)^{n}$$

$$\geq \left(\frac{1}{e}\right)^{\ell}\left(\frac{1}{4}\right)$$

$$\ell^* = \frac{\log n}{3\log\log n}$$

$$\ell^{\ell} \leq (\log n)^{\frac{\log n}{3\log\log n}} \leq 2^{\frac{\log n}{3\log\log n}\cdot\log\log n} \leq n^{1/3}$$

$$\Rightarrow \Pr(\text{bin } j \text{ has} \geq \ell^*) \geq \frac{1}{4n^{1/3}}$$

$$\Rightarrow E[\text{bins} \geq \ell^*] \geq \frac{n^{2/3}}{4}$$

What is the probability?

$X_j = 1$ if bin j has $\geq \ell^*$

$$\mathrm{Var}[X_j] = E[X_j^2] - E[X_j]^2 \leq E[X_j] \leq 1$$

$$\mathrm{Var}[X] = \sum \mathrm{Var}[X_j] + \sum_{i \neq j} \mathrm{Cov}(X_i, X_j)$$

$\mathrm{Cov}(X_i, X_j) \leq 0$ if X_i is 1, less likely for X_j to be 1

$$\leq \sum \mathrm{Var}[X_j] \leq n$$

$$\Pr[X = 0] \leq \Pr(|X - \mu| \geq \mu)$$

$$\leq \frac{E[(X-\mu)^2]}{\mu^2}$$

$$\leq \frac{\sum \mathrm{Var}[X_j] + \sum_{i \neq j} \mathrm{Cov}[X_i, X_j]}{\mu^2}$$

$$\leq \frac{n}{\left(\frac{1}{4} n^{2/3}\right)^2}$$

$$\leq 4 / n^{1/3}$$

Conclusion: $wp \geq 1 - \frac{4}{n^{1/3}}$, some bin has $\geq \ell^*$ balls

Hashing with Chaining

The simplest form of hashing is to resolve collisions with chaining. Each key x is hashed using hash function f to a table $A[1, \ldots, m]$. Each slot in the table stores a linked list containing all the keys hashed to that slot. To insert an item, simply compute $f(x)$ and add the key to the front of the linked list in slot $A[f(x)]$. This takes $O(1)$ time (assuming you can compute the hash function in $O(1)$ time). To search for an item, compute $f(x)$ and then search through the entire linked list at $A[f(x)]$. The cost of a search depends on the size of the linked lists.

Using the balls in bins analysis, you should be able to show that:

- If the hash table has size m and there are n items in the table, then the expected cost of an insert is $O(n/m)$.
- If $n = m$, the expect cost of an insert is $O(1)$.
- Using the analysis above, you can show that if $m = n$, then with high probability, every linked list has length $O(\log n / \log \log n)$, and so each search operation terminates in time $O(\log n / \log \log n)$ with high probability.
- You should be able to show that if $m = n$, then the following statement is *false*: with high probability, each linked list has length $O(1)$.
- In fact, using the analysis above, you can show that in expectation, many search operations will be cost $\Omega(\log n / \log \log n)$, and you can calculate the probability that at least one searc operation has cost $\Omega(\log n / \log \log n)$.

Linear Probing

→ Hash $h: U \to [0, \ldots, m-1]$

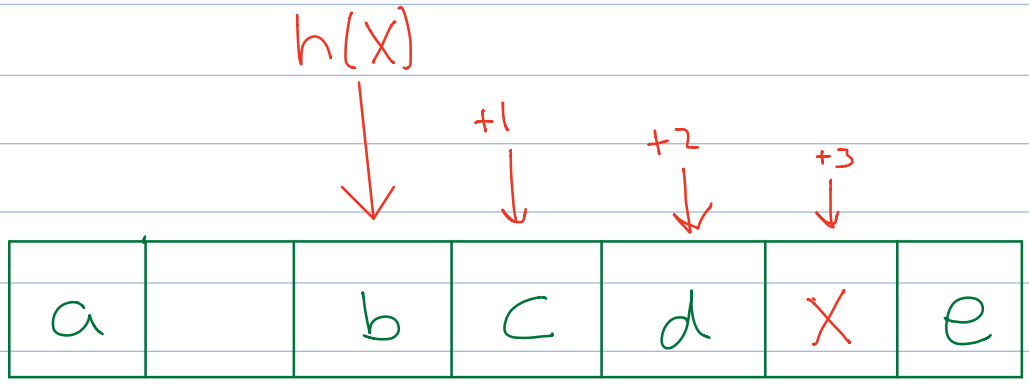
insert(X): try slot $h(X)$

$h(X)+1 \mod m$

$h(X)+2 \mod m$

⋮

until find empty slot



search(X): try slot $h(X)$

$h(X)+1 \mod m$

$h(X)+2 \mod m$

⋮

until find empty slot or X found

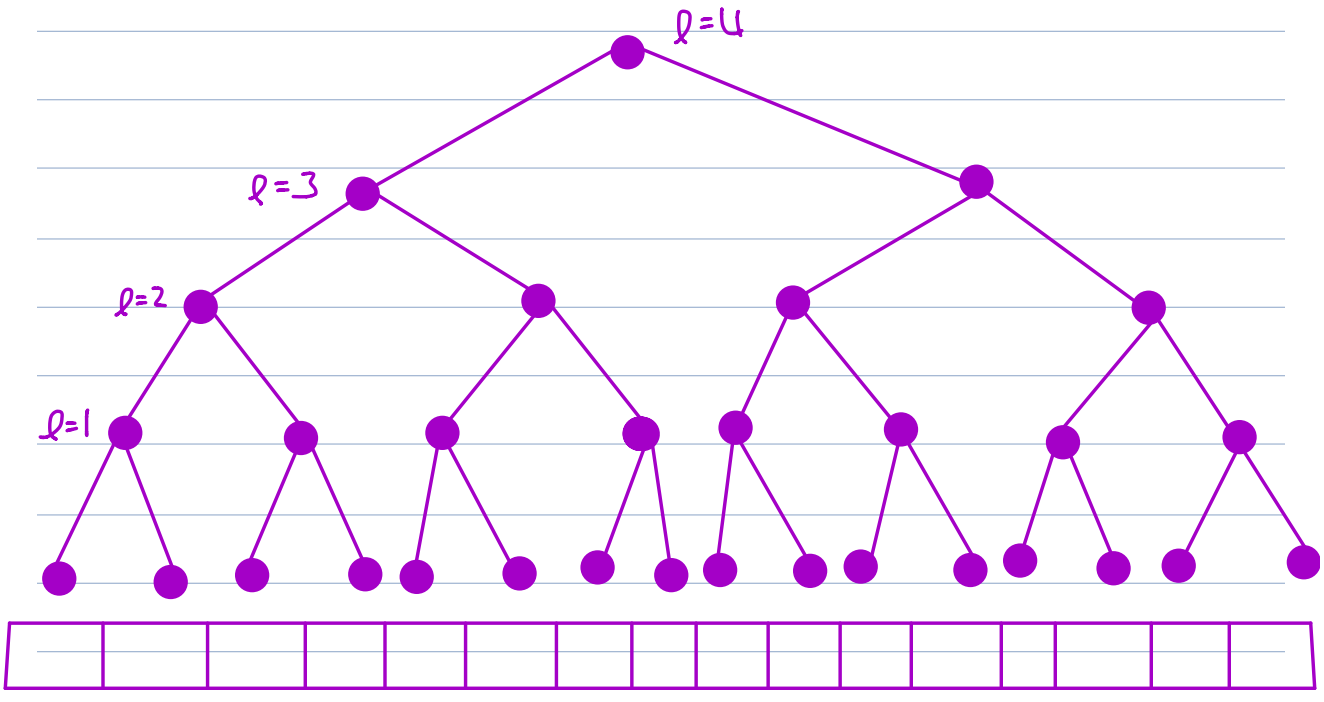
Clusters = BAD

→ E[cluster size] = ??

Goal: show that for item q (all other items inserted) cluster containing q is $E[\text{size}] = O(1)$.

Assume: $m \geq 4n$ (m: array size; n: # of keys inserted)

Step 1) Build tree over array



Each leaf = one slot in array

A level ℓ-node "owns" a 2^ℓ-sized subarray

$\rightarrow$ height $= \log(m) = \log(4n) = \log(n) + 2$

How many keys hash to node u?
(note: $h(x) \in u \not\Rightarrow x$ inserted in u.)

If u is level ℓ:

$$\Pr(X \in u) = \frac{2^\ell}{m} \leq \frac{1}{4} \cdot \frac{1}{n} \cdot 2^\ell$$

$$E[\#\text{ hash to } u] = E\left[\sum \tfrac{1}{4n} 2^\ell\right] = \tfrac{1}{4} \cdot 2^\ell$$

Define: u is crowded if $> \frac{3}{4} 2^\ell$ hash to u in level ℓ

Claim: u is unlikely to be crowded (postponed)

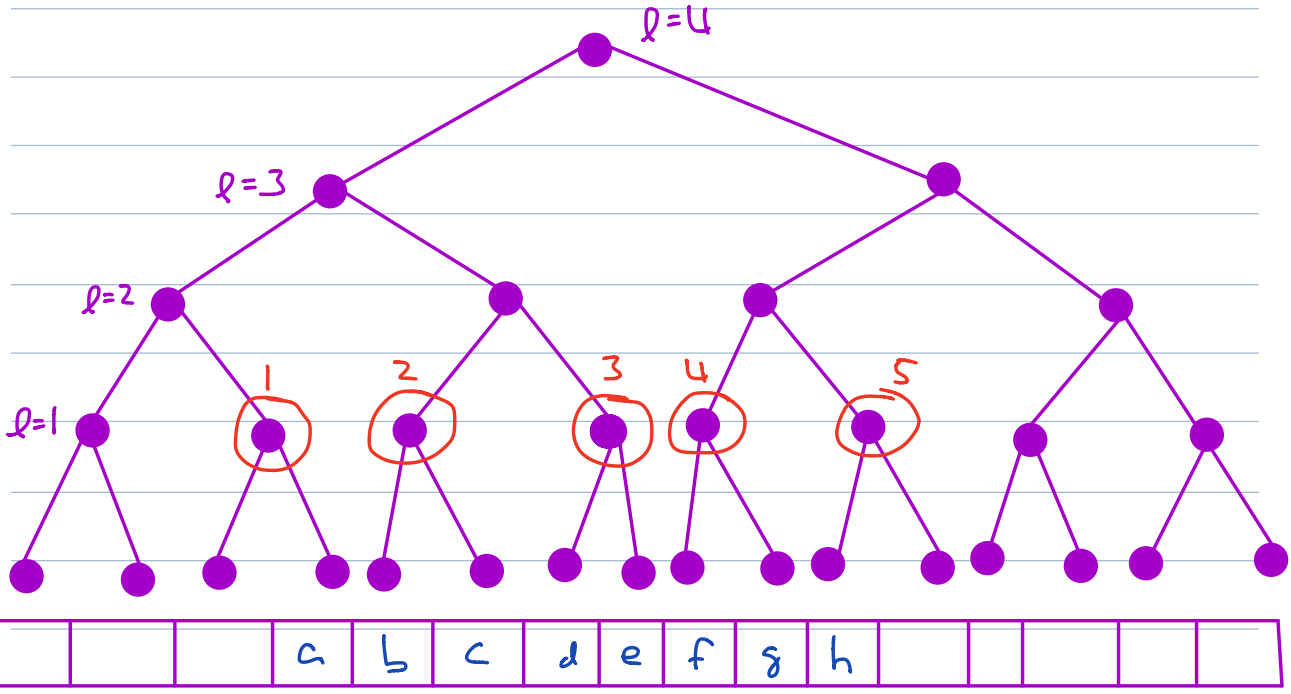
Let $p_\ell = \Pr(\text{node } u \text{ at level } \ell \text{ is crowded})$

Let $S = [s_1, s_2]$ be a cluster of length 2^ℓ

→ $A[s_1 - 1]$ is empty

→ $A[s_2 + 1]$ is empty

→ $A[j]$ is full, $s_1 \le j \le s_2$



→ S in at most $\underline{5}$ level $\ell-2$ nodes

Claim: ≥ 1 of first $\underline{4}$ level $\ell-2$ nodes for S are crowded

Why?

1) All keys in first 4 nodes hash to first 4 nodes (cannot hash earlier: $A[5,-1]$ is empty)

2) $\geq 3 \cdot 2^{\ell-2}+1$ keys hash to first 4 nodes
   - $\rightarrow \geq 1$ in node 1
   - $\rightarrow$ nodes 2, 3, 4 are full $\rightarrow$ covered by cluster
   - $\rightarrow \geq 3 \cdot 2^{\ell-2}+1$ in nodes 1,2,3,4 $\Rightarrow \geq 3 \cdot 2^{\ell-2}+1$ hash

3) If all 4 <u>not</u> crowded:
   $\leq 4 \cdot \frac{3}{4} \cdot 2^{\ell-2} \leq 3 \cdot 2^{\ell-2}$
   hash to first 4 nodes

Contradiction $\Rightarrow \geq 1$ first 4 nodes crowded

Long streak $\Rightarrow$ crowded nodes

Assume q is inserted in cluster of size $[2^{\ell}, 2^{\ell+1}]$

→ ≤ 9 level $\ell-2$ nodes contain cluster

→ Assume q is hashed to node u_j at level $\ell-2$

Claim: $N = [u_{j-8}, u_{j+3}]$ → 12 level $\ell-2$ nodes
≥ 1 node in N is crowded

Why?

Cluster in ≤ 9 nodes at level $\ell-2$

→ Cluster begins in u_{j-8} or later
→ First 4 nodes of cluster end at u_{j+3} or earlier
→ First 4 nodes in $[u_{j-8}, u_{j+3}]$
→ One of first 4 nodes is crowded

Claim: $\Pr[q \text{ is in cluster of size } [2^\ell, 2^{\ell+1}]] \le 12\, p_{\ell-2}$

For each $u \in N$, $\Pr[u \text{ crowded}] \le p_{\ell-2}$

Union bound...

$$E[\text{cost } q] \le \sum_{K=0}^{n} K \cdot \Pr[q \text{ in cluster of size } K]$$

$$\le \sum_{\ell=0}^{\log n} 2^{\ell+1} \cdot \Pr[q \text{ in cluster of size } [2^\ell, 2^{\ell+1}]]$$

$$\le 3 + \sum_{\ell=2}^{\log n} 2^{\ell+1} \cdot 12\, p_{\ell-2}$$

If $p_{\ell-2}$ is small, then $E[\text{cost}]$ is small!

Balls-and-Bins

n balls

b bins $\quad (b = n/2^{\ell-2} \leftarrow$ one bin per $\ell-2$ node$)$

Assume:

Fix one bin Z.

$E[Z] = \frac{n}{b} = M$ $\quad$ bin crowded $M = \frac{n}{n} 2^{\ell-2} = \frac{2^{\ell-2}}{4}$

Markov: $\Pr[Z > 3M] \leq 1/3$

Not good: $\sum_{\ell=2}^{\log n} 2^{\ell+1} \cdot \frac{1}{3} = \Theta(n)$

Chebychev:

$$\Pr[Z > 3M] \leq \Pr[|Z-M| > 2M]$$

$$\leq \frac{E[(Z-M)^2]}{4M^2}$$

Define $X_j = 1$ if ball j in bin
$= 0$ otherwise

$\Pr(X_j) = E[X_j] = 1/b$

$Z = \sum_{j=1}^{n} X_j$

Define $Y_j = X_j - \frac{1}{b}$

$$\sum Y_j = \sum X_j - \frac{n}{b} = Z - \mu$$

$$\Rightarrow E[(Z-\mu)^2] = E[(\sum Y_j)^2]$$

$$E\left[\left(\sum Y_j\right)^2\right] = \sum_{i=1}^{n}\sum_{j=1}^{n} E[Y_i Y_j]$$

1) $i \neq j$: Y_i, Y_j independent

$$E[Y_i Y_j] = E[Y_i]\,E[Y_j] = 0 \cdot 0 = 0$$

$$\left(E[Y_i] = E[X_i] - \frac{1}{b} = \frac{1}{b} - \frac{1}{b} = 0\right)$$

2) $i = j$: $E[Y_j^2] = \frac{1}{b}\left(1-\frac{1}{b}\right)^2 + \left(1-\frac{1}{b}\right)\left(-\frac{1}{b}\right)^2$

$$= \frac{1}{b} - \frac{2}{b^2} + \frac{1}{b^3} + \frac{1}{b^2} - \frac{1}{b^3}$$

$$= \frac{1}{b} - \frac{1}{b^2}$$

$$\leq \frac{1}{b}$$

Conclusion:

$$E\left[\left(\sum_j Y_j\right)^2\right] \le \sum_{j=1}^{n} E\left[Y_j^2\right] \le \frac{n}{5} = M$$

$$\Pr\left(|Z - M| > 2M\right) \le \frac{M}{4M^2} \le \frac{1}{4M}$$

$$\frac{1}{4M} = \frac{1}{4} \cdot \frac{n}{2^{\ell-2}} \cdot \frac{1}{n} = \frac{1}{4} \cdot \frac{1}{2^{\ell-2}}$$

$$\Rightarrow \sum_{\ell=2}^{\log n} 2^{\ell+1} \cdot \frac{1}{4 \cdot 2^{\ell-2}} = \sum_{\ell=2}^{\log n} \frac{1}{4} \cdot 8 \simeq 2 \log(n)$$

Better...

On to the fourth moment!

Why not 3??

$$\Pr[|Z-M| \geq 2M] = \Pr[|Z-M|^4 \geq 16M^4]$$

$$\leq \frac{E[(Z-M)^4]}{16M^4}$$

$$E[(Z-M)^4] = E\left[\left(\sum Y_i\right)^4\right] = \sum_v \sum_w \sum_x \sum_z E[Y_v Y_w Y_x Y_z]$$

Case 1) One index is unique: 0

eg: $w \neq v$

$w \neq x$

$w \neq z$

Y_w independent of Y_v, Y_x, Y_z

$$E[Y_v Y_w Y_x Y_z] = E[Y_w] E[Y_v Y_x Y_z]$$
$$= 0 \cdot E[Y_v Y_x Y_z]$$
$$= 0$$

Case 2) Two Pairs: $E[Y_w Y_w Y_z Y_z]$

Y_w^2, Y_z^2 independent $\longrightarrow$ $= E[Y_w^2] E[Y_z^2]$

$$\leq \left(\frac{1}{6}\right)^2$$

Case 3) All identical: $E[Y_u Y_v Y_u Y_v]$

$$= E[Y_u^4] = \left(\frac{1}{b}\right)\left(1-\frac{1}{b}\right)^4 + \left(1-\frac{1}{b}\right)\left(-\frac{1}{b}\right)^4$$

$\left(1-\frac{1}{b}\right) \le 1$, $\frac{1}{b} \le 1$ $\quad \le \frac{1}{b}\left(1-\frac{1}{b}\right)^2 + \left(1-\frac{1}{b}\right)\left(\frac{1}{b}\right)^2$

$$\le 1/b$$

Total: n terms of case 3
$\le n^2$ terms of case 2

$$E\left[\left(\sum Y_i\right)^4\right] \le \frac{n^2}{b^2} + \frac{n}{b} \le 2M^2$$

$$\Pr[|Z-M| > 2M] \le \frac{2M^2}{16M^4} = \frac{1}{8M^2}$$

$$\frac{1}{8M^2} = \frac{1}{8} \cdot \frac{1}{(2^{l \cdot 2})^2} = \frac{1}{8} \cdot \frac{1}{2^{2l-4}}$$

$$E[\text{cost } q] \le 3 + 12 \cdot \sum_{\ell=2}^{\log m} 2^{\ell+1} P_{\ell-2}$$

$$\le 3 + \frac{12}{8} \sum_{\ell=2}^{\log m} 2^{\ell+1} \cdot \frac{1}{2^{2\ell-4}}$$

$$\le 3 + \frac{12}{8} \sum_{\ell=2}^{\log m} \frac{1}{2^{\ell-5}}$$

$$\le 3 + \frac{12}{8} \cdot \frac{32}{2} \le 21 = O(1)$$

$E[\text{cost of insert/access}] = O(1)$

More care = smaller constant

More care $\Rightarrow$ $m = n(1+\varepsilon)$

In practice: Very fast!

Cache performance easily more important than constant factors.

Moral: Use linear probing

(Don't believe what they tell you...)

Cuckoo Hashing

This week, we considered Cuckoo Hashing. The basic idea behind Cuckoo Hashing is that we use to tables A and B of size $m = 2n$ to store n items. For table A, we use hash function f, and for table B, we use hash function g. To insert an item, we proceed as follows, executing `insert(x, A, f)`:

```
insert(item x, Table T, hashfunction h)
      slot = h(x)
      z = T[slot]
      T[slot] = x
      if (z != null)
            if (T == A)
                  insert(z, B, g)
            else if (T == B)
                  insert(z, A, f)
```

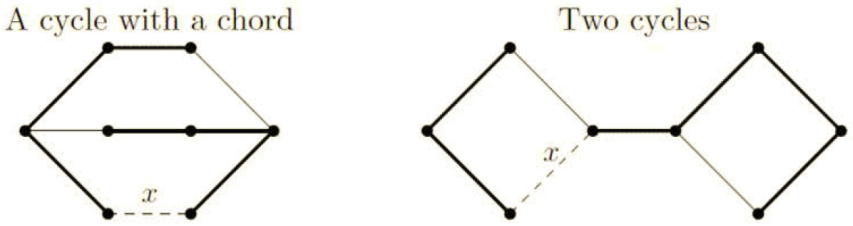
That is, we put the new item in the proper slot in table A. If the slot is already occupied by some item z, then we kick z out of table A and insert it (recursively) into table B. Eventually, if this proceeds for more than $2m$ steps, then we know there is a problem and we abort.

Searching for an item is simple, and requires at most 2 queries:

```
search(x)
      if (A[f(x)] == x) then return A[f(x)];
      if (B[g(x)] == x) then return B[g(x)];
      else return NOT_IN_TABLE;
```

Item x can be in at most two locations: table A or table B, so searching is very fast: $O(1)$ time.

An insertion, on the other hand, can be more expensive. If $m \geq 2n$, we can show that most of the time, each insertion completes in $O(\log n)$ with high probability. But sometimes, on occasion, you may have a problem. For example, imagine you have three keys x, y, z where $f(x) = f(y) = f(z)$ and $g(x) = g(y) = g(z)$. In this case, the insertion will loop forever—or until it aborts. More generally, we have a problem when the Cuckoo Graph has two cycles[1]:



In either of these cases, we will not be able to insert all the items. When this happens, we choose new hash functions and try again to insert all the items. We can show that for a given set of n elements, there is a probability of $O(1/n)$ of failure. (Below, we will suggesting proving the easier bound, i.e., that there is a probability $1/2$ of a failure.) Thus, with probabiliy $1/n$ we will have to re-hash (and re-build) the tables, which has cost $O(n)$. Thus the expected cost due to rebuilding is $O(1)$. Overall, then, we can show that insertions have expected amortized cost $O(1)$.

[1]Figure taken from: `http://infoweekly.blogspot.sg/2010/02/cuckoo-hashing.html`

Analysis

To analyze the performance of Cuckoo Hashing, we consider the Cuckoo Graph which is a bipartite graph defined as follows: there are m left node representing array A and m right nodes representing array B; for every key x, add one edge to the graph connecting node $f(x)$ on the left side to node $g(x)$ on the right side.

Notice that when analyzing the Cuckoo Graph, we are not actually *executing* the insertion algorithm. We are simply looking at the edges that would be created by the keys. By looking at the Cuckoo Graph for n keys, we can determine both whether or not it is possible to insert them all (without an infinite loop) and the expect cost of each insertion.

As an exercise, prove the following facts regarding Cuckoo Hashing. Assume throughout that $m = 4n$.

- Given a Cuckoo Graph containing n edges, show that any path of length p in the Cuckoo Graph that is generated by an insert operation that terminates has a simple subpath (with no repeated nodes) of length at least $p/3$.
- Given a Cuckoo Graph containing n edges, show that for all nodes i and j, for all ℓ, the probability that the graph contains a path of length exactly ℓ from i to j is at most $\frac{1}{4^\ell}\frac{1}{m}$.
- Prove that the expected cost of an insert operation is $O(1)$ (as long as the Cuckoo Hash Table contains at most n items).
- Given a Cuckoo Graph containing n edges, prove that the probability it contains a cycle is at most $1/2$.
- Whenever an insert does not terminate (i.e., ends in an infinite loop), you need to create a new Cuckoo Hash Table, choose new hash functions, and reinsert all n items. (If your insert operation continues for $2m$ steps, you can assume you have entered an infinite loop!). Assume that you continue this process, rehashing as necessary, until you successfully insert all n items. Prove that the expected total cost of this process is $O(n)$ (including the cost of the repeated rehashing).