

Analysis of Open Addressing

- Simplifying assumption: $h(\text{key}, \text{probe \#})$, random and uniform.
- Probability that **at least i probes** lead to **already occupied slots** ?

$$q_i = \left(\frac{n}{m}\right)^i = \alpha^i$$

- $1 +$ Expected # probes in **unsuccessful search**:

$$1 + \sum_{i=1}^{\infty} i \underbrace{\Pr[\text{exactly } i \text{ probes}]_{p_i}} = 1 + \sum_{i=1}^{\infty} q_i = 1 + \sum_{i=1}^{\infty} \alpha^i = \frac{1}{1-\alpha}$$

Why?	p_1	p_2	p_3	\dots	q_1
		p_2	p_3	\dots	q_2
			p_3	\dots	q_3
	p_1	$2p_2$	$3p_3$	\dots	$\sum ip_i = \sum q_i$

100

More open addressing

- What about **successful search** ?
Depends on the element: element inserted **earlier** will be **easier to find** !

- Assume uniform distribution on the element we search for.
If element was inserted at $(i+1)$ -th step, expected number of probes

$$\text{was } \leq \frac{1}{1-\frac{i}{m}} = \frac{m}{m-i}$$

- Condition on i , take expectation:

$$\leq \frac{1}{n} \sum_{i=0}^{n-1} \frac{m}{m-i} = \frac{m}{n} \sum_{i=0}^{n-1} \frac{1}{m-i} = \frac{1}{\alpha} [H_m - H_{m-n}]$$

But: $\ln i \leq H_i \leq \ln i + 1$

Thus expected #probes:

$$\leq \frac{1}{\alpha} [\ln m + 1 - \ln(m-n)] = \frac{1}{\alpha} \left[\ln \frac{m}{m-n} + 1 \right] = \frac{1}{\alpha} \left[\ln \frac{1}{1-\alpha} + 1 \right]$$

Note the difference between this result and the (intuitive) "half of insertion time" !

101

Comments

- Good hash functions are hard to invent
- Analysis was based on heavy (and incorrect) assumptions
- Adversary can make any specific hash function “fail”
- Existence of “universal hash functions”
- Difference in performance between chaining and open addressing

107

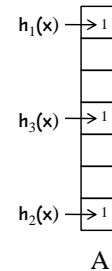
Bloom filter

- Standard hashing – no mistakes
 - » If the element is in the set – result “yes”
 - » If the element is not in the set – result “no”
- Can we reduce the space if we are willing to “make mistakes”
 - » If the element is in the set – result “yes”
 - » If the element is not in the set – result “no” with high probability (low probability of false positive)
- Examples of where this might be useful:
 - » Dictionary – will miss small fraction of misspellings
 - » Distributed cache – with small probability will try to extract non-existent page from a remote cache
 - » No space to store the actual objects (or their long unique IDs)

108

Bloom filters continued

- Setting:
 - » n elements
 - » k different (independent) hash functions
 - » Each element is hashed into range 1 to m .
 - » Bit vector $A[]$ of length m , initially all bits are zero.
- To hash an element x :
 - » For each hash function h_i , set to $A[h_i(x)]=1$
- To check whether element x is in the set:
 - » Return “yes” if for all i , $A[h_i(x)]=1$
 - » “no” otherwise
- Observe that there is a possibility of false positive



109

Bloom Filter Analysis

- After all elements are hashed, probability of specific bit in A still being zero:

$$p = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m} \quad \text{since } \left(1 - \frac{1}{m}\right)^m \approx e^{-1}$$

- Probability of false positive is

$$(1 - p)^k$$

- What is the “right” number of hash functions (k) ?

» Two competing factors:

- Larger k increases probability of finding zero if the element is not in the set (good)
- Too large k will fill A with too many ones (p too small - bad)

110

Analysis continued

- Rewrite false positive probability:

$$(1-p)^k \approx (1 - e^{-kn/m})^k = \exp(k \ln(1 - e^{-kn/m}))$$

- We will try to minimize the argument of the exponent.

Using
$$p \approx e^{-kn/m}$$

rewrite the exponent as a function of p:

$$k \ln(1 - e^{-kn/m}) = -\frac{m}{n} \ln p \ln(1 - p)$$

- From symmetry considerations, the minimum is at $p=0.5$:

$$k = \frac{m}{n} \ln 2$$

- For this value of k, the false positive rate is

$$(0.5)^k = (0.62)^{m/n}$$

111

Example

- Consider $m/n=10$
- If $k=1$, then false positive probability is 0.1

- If

$$k \approx (m/n) \ln 2 \approx 7$$

then false positive probability is:

$$(0.5)^k = (0.62)^{m/n} \approx 0.008$$