Lecture 8, Thursday 4/26/01

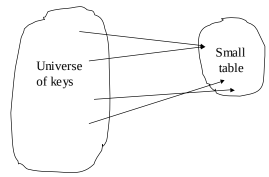
Hashing (Chapter 12)

- Heaps support:
  - » Insert
  - » Delete
  - » Max/min
- How about Search ?
  (How would you implement "find" in a heap ?)
- Possible solutions:
  - » Ordered array: slow "insert" - $\Omega(n)$.
  - » Ordered list: both find and insert are slow !

74

Direct address table

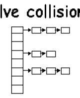
- Maintain table T[i] $T[i] = \begin{cases} x & \textit{if } x \in T, key(x) = i \\ \varnothing & \textit{otherwise} \end{cases}$
- Disadvantage: too much memory !
- Idea: maintain small table:



75

Collisions

- | Table | << | Universe of keys | - collisions !
  (collision= two keys map into the same slot in T)
- How to resolve collisions:
  - » Chaining:



What is "next" ?

  - » Open addressing: if A[h(x)] full - try next slot.

76

Analysis of Chaining

- Assume each key equally likely hashed to any slot.
- n keys, m slots; $\alpha = \frac{n}{m} =$ "load factor"
- Expected length of a chain: $\sum_{1}^{n} \frac{1}{m} = \frac{n}{m} = \alpha$

  $\Rightarrow$ Access time = $O(1+\alpha)$
- Unsuccessful search:
  Expected length of a randomly chosen list +1: $O(1+\alpha)$

77

Successful Search

- Expected time to find *i*-th element = time to insert *i*-th element
- *Assume that the key being searched for is equally likely to be any one of the keys stored.*
- Conditioned on "key was the *i*-th element inserted",

  expected time = $\left(1+\frac{i-1}{m}\right)$

  overall: $\frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{i-1}{m}\right) = 1+\frac{1}{nm}\sum_{i=1}^{n}(i-1)$

  $$= 1+\frac{1}{nm}\frac{n(n-1)}{2} = 1+\frac{\alpha}{2}-\frac{1}{2nm} = O(1+\alpha)$$
- Intuition: need to search 1/2 of a list on the average.

78

Open Addressing

- If A[h(x)] full, try "next" slot.
- Linear probing:
  - » pick some integer b relatively prime to size of table m.
  - » For i = 0, 1, 2, 3, ... try to place x in position:
    h(x) + b·i mod m
  - » Bad idea: results in large clusters.
    Increased search time and insert time as $\alpha \to 1$.
- Double hashing: works well in practice.
  - » Pick two hash functions h_1, h_2
  - » For i = 0, 1, 2, 3,... try to place x in position:
    $h_1(x) + i \cdot h_2(x)$ mod m

79

Analysis of Open Addressing

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

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

- Expected # probes in unsuccessful search:

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

$$Why? \quad \begin{array}{cccc|c} p_1 & p_2 & p_3 & \cdots & q_1 \\ & p_2 & p_3 & \cdots & q_2 \\ & & p_3 & \cdots & q_3 \\ \hline p_1 & 2p_2 & 3p_3 & \cdots & \sum i p_i = \sum q_i \end{array}$$

80

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 was $\le \frac{1}{1-\frac{i}{m}} = \frac{m}{m-i}$
- Condition on i, take expectation:

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

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

Thus expected #probes:

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

81