Details of Probing Techniques

Probing is necessary when *collision* occurs: where do we *insert* the new item that has hashed into an already occupied position?

In this lecture we re-examine in more detail *linear probing* to expose its potential problems.

Linear Probing Again

The algorithm here is to search sequentially from the already occupied position until we find a vaccant one, and we palce the new item there. For a “typical” example that illustrates the problem with this, see figure 20.4 in the text.

Advantage: If a vacant cell exists, it will be found.

Disadvantages:

This can take a long time, on the average 1/2 the table size. As hashing derives its main advantage from the desire to have *near constant time* insertions and deletions, this is bad news.

Another problem is *clustering*, the tendency of items to group *contiguously* together, thereby exaberbating the first problem.

Bernoulli Trials

We want an abstract model of simplified probing problems.

Such models can also reveal deficiencies of simplification.

Consider the repeated tossing of a *biased* coin. Each toss is a *trial*, and is independent of any other trial. At each trial we get H(ead) with probability p and T(ail) with probability $q = (1 - p)$.

So, a “typical” experiment of N trials may be represented as an N-length string of successive *outcomes* like $HHTHTTTHTH \ldots HTT$.

Such experiments are called *Bernoulli Trials*.

Bernoulli Trials cont’d

We are interested in the *expected waiting time* for the *first* occurrence of, say, an H. Call H a “success”, and conversely T is a “failure”.

Obs I:
The strings that represent the *Event* that, say, H occurs for the first time on the third trial, all have to begin with $TTH \ldots$. The experiment must be such that there are 2 failures followed by a success. The probability of this is therefore
$Prob(fail) * Prob(fail) * Prob(succeed) =$
$Prob(T) * Prob(T) * Prob(H) = q * q * p$.

Bernoulli Trials cont'd

Obs II:
Generalizing, the probability of the Event $Succ(k)$ that the first success is at the $k+1$-st trial is $q^k * p$.

Obs III:
Therefore the *expected number* of trials to the first success is $\sum_{k=0}^{k=N}(k+1) * Prob(Succ(k)) = \sum_{k=0}^{k=N}(k+1) * q^k * p$

Bernoulli Trials cont'd

To simplify the evaluation of $\sum_{k=0}^{k=N}(k+1) * q^k * p$ we make the assumption that N (the number of trials, length of the experiment) is large.

Then we might as well evaluate instead $\sum_{k=0}^{k=\infty}(k+1) * q^k * p$. Factoring out p yields $p * \sum_{k=0}^{k=\infty}(k+1) * q^k$.

Bernoulli Trials cont'd

The usual evaluation of $\sum_{k=0}^{k=\infty}(k+1) * q^k$ is by *generating functions*, but we will do it from first principles.

Observe that each term is in fact $(k+1) * q^k = \frac{d}{dq}q^{k+1}$, the sum can be re-written $\frac{d}{dq}\sum_{k=0}^{k=\infty} q^k$.

But the summand is an infinite convergent geometric series with sum $\frac{1}{1-q}$.

Bernoulli Trials Waiting Time Theorem

So the sum is $\frac{d}{dq}(\frac{1}{1-q})$, which is $\frac{1}{(1-q)^2}$. But $1-q=p$, so (remembering the factored out p), we have the result that the sum evaluates to $\frac{1}{p}$

Theorem 1 *Bernoulli Waiting Time*
The expected number of trials to the first success in Bernoulli trials with probability of success p in each trial is $\frac{1}{p}$.

Linear Probing — Approximate Analyis

Assume (1) large table (2) each probe independent of any other. Each probe is a new “trial”. Thus we have Bernoulli Trials in which “success” is finding an unoccupied cell.

Let the *Load Factor* (= fraction of table already occupied) now be denoted by λ; so $0 \leq \lambda \leq 1$.

Theorem 2 *(Linear Probes, Bernoulli Assumption)*
Under the assumptions (1) and (2), the average number of seeks for an unoccupied cell to insert an item is $\frac{1}{1-\lambda}$.

Proof: *With this load factor, the probability of a cell being empty is* $1-\lambda$*. Thus by the Bernoulli Trial Waiting Time Theorem, the expected number of seeks to find one is* $\frac{1}{1-\lambda}$.

A Closer Look

How realistic is this Bernoulli Trial model?

Independence of Trials is not realistic when *primary clustering* (se fig 20.5 of text) occurs.

When this happens, large contiguous blocks of occupied cells increase the seek time after collisions within such blocks, and whenever a probe is unsuccessful, its probability of success in the next probe is lower than if it had not failed in linear sequential search.

Linear Probe Seek Time w/o Independence

The analysis without the independence assumption (no longer a Bernoulli experiment) is complex and tedious. The result (no proof here):

Theorem 3 *Linear Probe Average Seek Time*
The average number of probes to locate an empty cell to insert an item is $\frac{1}{2}(1+\frac{1}{(1-\lambda)^2})$.

Finding an item

Two cases for finding an item: (1) unsuccessful — the item was not there (2) successful — the item is found.

Case (1): This is the same as the seek time for inserting the item, because to insert the item we have to first find if it is there, and only insert it if it is not. These are given by the theorems above.

Case (2): Success. The analysis of this resembles the analysis of of the cost of seeking a item in *chaining* treated in the last lecture. The key idea here is the same, but we describe it again.

Finding an item — success case

Suppose there are already K items in the table. Then it must be the case that there is a *history* of inserting the $1st$, $2nd$, $3rd$, …, j *th*, …, K *th* items in succession.

Concentrate on what happened when the j *th* item was being entered.

At that time T, we had to *insert* it when the load factor was $\lambda(j) = (j-1)/N$ where N is the table size.

Later, after all K items have been inserted, to find this j *th* item the cost of doing so is *the same as the cost of inserting it at time T* because to find it we would have to go through the same probes as we did at time T to insert it.

Finding an item — success case cont'd

So, to find the overall *average* cost of successfully locating the K items, we must average over the costs of inserting the $1st$, $2nd$, $3rd$, …, j *th*, …, K *th* items.

This is $\frac{1}{K}\sum_{j=1}^{K} I(j)$

where I(j) is the cost of inserting the j *th* item (when the load factor was $\lambda(j) = j/N$).

But from the theorem above we know that $I(j) = \frac{1}{2}(1 + \frac{1}{(1-\lambda(j))^2})$. So we must evaluate

$\frac{1}{K}\sum_{j=1}^{K} \frac{1}{2}(1 + \frac{1}{(1-\frac{j}{N})^2})$.

Finding an item — success case cont'd

Evaluating $\frac{1}{K}\sum_{j=1}^{K} \frac{1}{2}(1 + \frac{1}{(1-\frac{j}{N})^2})$.

Let λ be the load factor after the K *th* element was entered, i.e. $\lambda = K/N$.

Then $\frac{j}{N}$ is simply the load factor for the time when the *j-th* item was entered. So we can approximate the sum by the *continuous* integral

$\frac{1}{\lambda}\int_{x=0}^{\lambda} \frac{1}{2}(1 + (\frac{1}{1-x})^2)dx$

which can be verified to be $\frac{1}{2}(1 + \frac{1}{1-\lambda})$

Finding an item — success case cont'd

Theorem 4 {*Avge Search Costs in Insertion/Deletion w Linear Probing*}
If the load factor is λ:

An unsucessful search cost = insertion cost = $\frac{1}{2}(1 + (\frac{1}{1-\lambda})^2)$.

A successful search cost = $\frac{1}{2}(1 + \frac{1}{1-\lambda})$.