Data Structures and Algorithms Dr. L. Rajya Lakshmi

- How many distinct probe sequences are possible with
 - Linear probing?
 - The initial probe decides the subsequent probes
 - Quadratic probing?
 - Here also, the subsequent probes are dependent on the initial probe
 - Double hashing?
 - When N is a prime or a power of 2, double hashing uses $\theta(N^2)$ probe sequences, since each possible (h'(k), h''(k)) pair yields a distinct probe sequence
- For the analysis, we assume uniform hashing
- Uniform hashing: the probe sequence for each key is equally likely to be any one of N! permutations of (0, 1, . . ., N-1)
- When the values of parameters are selected appropriately, double hashing performs close to ideal scheme of uniform hashing

- Analysis is in terms of load factor $\alpha = n/N$ (n is the number of items in the hash table and N is the capacity of the hash table)
- With open addressing, $n \le N$, so $\alpha \le 1$
- Assume that we are using uniform hashing
- The probe sequence (h(k,0), h(k,1), . . ., h(k, N-1)) used for key k is equally likely to be any of the permutation of (0, 1, . . ., N-1)

Theorem: Given an open addressing hash table with load factor $\alpha = n/N < 1$, the expected number of probes in an unsuccessful search is at most $1/(1-\alpha)$, assuming uniform hashing.

Proof:

Every probe except for the last one accesses an occupied bucket that does not contain the required item

The last bucket accessed is an empty one

X denotes the number of probes made in an unsuccessful search

A_i: the event that an ith probe occurs and it is to a occupied bucket

- Now consider the event $\{X \ge i\}$
- 1st probe occurs and it is to a occupied bucket, 2nd probe occurs and it is to a occupied bucket, . . ., (i-1)th probe occurs and it is to a occupied bucket
- The event $\{X \ge i\}$ is the intersection of the events $A_1, A_2, \ldots A_{i-1}$
- $\{X \ge i\}$ is $A_1 \cap A_2 \cap \ldots \cap A_{i-1}$

For a collection of events $A_1, A_2, \ldots A_{i-1}$ the following relation holds:

$$Pr\{A_1 \cap A_2 \cap ... \cap A_{i-1}\} = Pr\{A_1\}. Pr\{A_2 \mid A_1\}. Pr\{A_3 \mid (A_1 \cap A_2)\}...$$

 $Pr\{A_{i-1} \mid A_1 \cap A_2 \cap ... \cap A_{i-2}\}$
 $Pr(A_1) = n/N$

Consider the event that jth probe occurs and it is to an occupied bucket, given that the first j-1 probes were to occupied buckets, j > 1

In the jth probe we would find one of the remaining (n-(j-1)) items in one of the remaining (N-(j-1)) buckets (uniform hashing)

Probability of the event is (n-(j-1))/(N-(j-1))

Observing that n < N, $(n-j)/(N-j) \le n/N$, for all j s.t. $0 \le j < N$

 $Pr\{A_2 | A_1\} = n-1/(N-1); Pr\{A_3 | (A_1 \cap A_2)\} = n-2/(N-2), ..., Pr\{A_{i-1} | A_1 \cap A_2 \cap ... \cap A_{i-2}\} = n-i+2/(N-i+2)$

Since the event $\{X \ge i\}$ is $A_1 \cap A_2 \cap \ldots \cap A_{i-1}$

$$\Pr\{X \ge i\} = \frac{n}{N} \cdot \frac{n-1}{N-1} \cdot \frac{n-2}{N-2} \cdot \dots \cdot \frac{n-i+2}{N-i+2} \text{ (for all i s.t. } 0 \le i \le N)$$

$$\le \left(\frac{n}{N}\right)^{i-1}$$

$$= \alpha^{i-1}$$

When a random variable takes values from the set of natural numbers $N = \{0, 1, ...\}$, we have a formula for its expectation:

$$E[X] = \sum_{i=0}^{\infty} i. \Pr\{X = i\}$$

$$= \sum_{i=0}^{\infty} i. (\Pr\{X \ge i\} - \Pr\{X \ge i + 1\})$$

$$= \sum_{i=1}^{\infty} \Pr\{X \ge i\}$$

Using the above mentioned relation

$$E[X] = \sum_{i=1}^{\infty} \Pr\{X \ge i\}$$

$$\leq \sum_{i=1}^{\infty} \alpha^{i-1}$$

$$= \sum_{i=0}^{\infty} \alpha^{i}$$

$$= \frac{1}{1-\alpha}$$

- Unsuccessful search runs in O(1) if α is constant.
- If the hash table is half full, then the average number of probes in an unsuccessful search is 2
- If hash table if 90% full, then the average number of probes in an unsuccessful search is 10

Corollary: Inserting an item into an open addressing hash table with load factor α requires at most $1/(1-\alpha)$ probes on average assuming uniform hashing.

Proof:

An item is inserted into a hash table if and only if there is room, that is $\alpha < 1$

Inserting an item: an unsuccessful search followed by placing the item into the empty bucket found

The expected number of probes is at most $1/(1-\alpha)$

Theorem: Given an open addressing hash table with load factor α < 1, the expected number of probes in a successful search is at most

$$\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$$

assuming uniform hashing and assuming that each key in the table is equally likely to be searched for.

Proof:

A search for key k reproduces the probe sequence that was used while inserting that key into hash table

By the corollary, if the key k was the (i+1)st item inserted into hash table, then the expected number of probes in search for key is: 1/(1-i/N), that is, N/(N-i)

Averaging over all n items in the table,

$$\frac{1}{n} \sum_{i=0}^{n-1} \frac{N}{N-i} = \frac{N}{n} \sum_{i=0}^{n-1} \frac{1}{N-i}$$

$$A = \frac{1}{\alpha} \sum_{k=N-n+1}^{N} \frac{1}{k}$$

$$\int_{p}^{q+1} f(x) dx \le \sum_{k=p}^{q} f(k) \le \int_{p-1}^{q} f(x) dx$$

$$A \le \frac{1}{\alpha} \int_{N-n}^{N} (\frac{1}{x}) dx$$

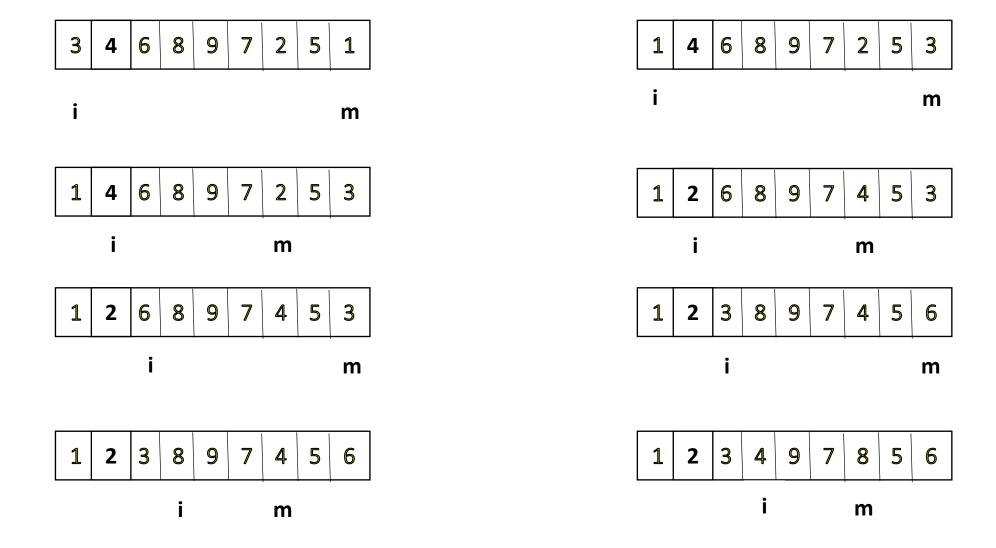
$$= \frac{1}{\alpha} \ln \frac{N}{N-n}$$

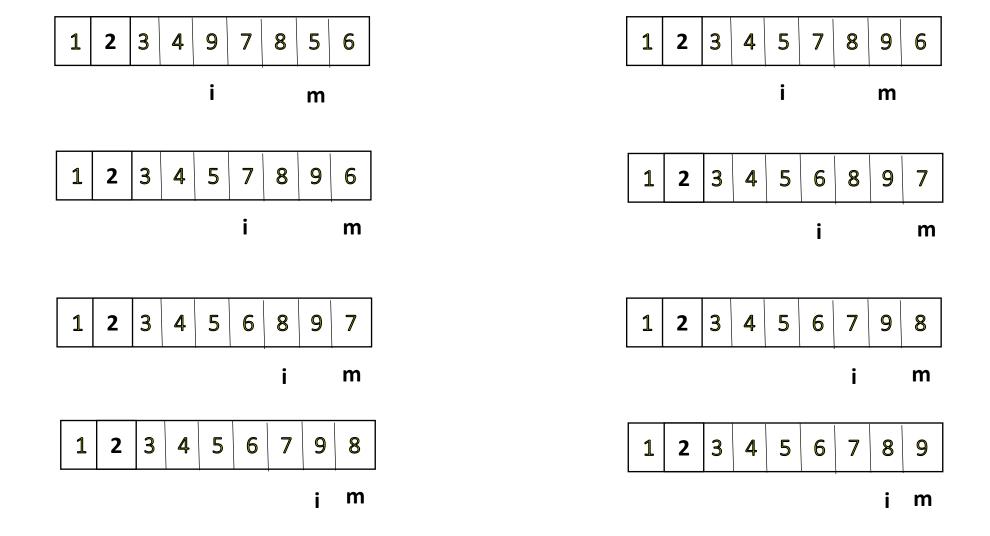
$$= \frac{1}{\alpha} \ln \frac{1}{1-\alpha}$$

- If the table is half full, then the expected number of probes in a successful search is 1.387
- If the table is 90% full, then the expected number of probes in a successful search is: 2.559

- Similar to the insertion, the given input array can be divided into two parts: sorted and unsorted part
- Basic Principle: Take the smallest elements from the unsorted part and move it to the end of the sorted part

```
Algorithm Selection_Sort(A[0..n-1], n)
        for i \leftarrow 0 to n-2 do
                m \leftarrow i
                for j \leftarrow i+1 to n-1 do
                        if A[j] < A[m]
                                 m \leftarrow j
                swap A[i] and A[m]
```





```
Algorithm Selection Sort(A[0..n-1], n)
         for i \leftarrow 0 to n-2 do
                  m \leftarrow i
                  for j \leftarrow i+1 to n-1 do
                           if A[i] < A[m]
                                    m \leftarrow i
                  swap A[i] and A[m]
• T(n) = \sum_{i=0}^{n-2} \sum_{i=i+1}^{n-1} c
        = \sum_{i=0}^{n-2} c(n-i-1) = c \sum_{i=0}^{n-2} (n-1) - c \sum_{i=0}^{n-2} i
         = c(n-1)(n-1) - c(n-2)(n-1)/2 = cn(n-1)/2
```