

Path charges

Observations:

1. If a node x_i is assessed a path charge, then $p[x_i] \neq x_l$.
 $\implies x_i$ must be assigned a new parent during path compression.
2. x_i 's new parent must have higher rank than its old parent.

Lemma 1. *A node can be assessed at most $B(j) - B(j - 1) - 1$ path charges while its rank is in block j .*

Counting path charges

We can bound the path charges using $N(j)$, the number of nodes with rank in block j :

$$N(j) \leq \sum_{r=B(j-1)+1}^{B(j)} \frac{n}{2^r}$$

for $j = 0$:

$$\begin{aligned} N(j) &= n/2^0 + n/2^1 \\ &= 3n/2 \\ &= 3n/2B(0) \end{aligned}$$

for $j \geq 1$,

$$\begin{aligned} N(j) &\leq \frac{n}{2^{B(j-1)+1}} \sum_{r=0}^{B(j)-(B(j-1)+1)} \frac{1}{2^r} \\ &< \frac{n}{2^{B(j-1)+1}} \sum_{r=0}^{\infty} \frac{1}{2^r} \\ &= \frac{n}{2^{B(j-1)}} \\ &= \frac{n}{B(j)} \leq 3n/2B(j) \end{aligned}$$

So, for any $j \geq 0$, we have $N(j) \leq 3n/2B(j)$.

Summing over all blocks to get $P(n)$, the overall number of path charges,

$$\begin{aligned} P(n) &\leq \sum_{j=0}^{\lg^* n - 1} \frac{3n}{2B(j)} (B(j) - B(j-1) - 1) \\ &\leq \sum_{j=0}^{\lg^* n - 1} \frac{3n}{2B(j)} B(j) \\ &= \frac{3}{2} n \lg^* n \end{aligned}$$

Total runtime

Thus the total number of charges incurred by FIND-SET operations is

$$\begin{aligned} O(\text{block charges} + \text{path charges}) = \\ O(m \lg^* n + 1) + n \lg^* n \end{aligned}$$

which is $O(m \lg^* n)$ since $m \geq n$.

Since there are $O(n)$ MAKE-SET and LINK operations, each with 1 charge, the total time is

$$O(m \lg^* n + n) = O(m \lg^* n)$$

Hashing

A data structuring technique for supporting **Dynamic Dictionary** operations: INSERT, SEARCH, and DELETE.

Given a set of possible keys U , such that $|U| = u$ and a table of m entries, a **Hash function** h is a mapping from U to $M = \{1, \dots, m\}$.

A **collision** occurs when two hashed elements have $h(x) = h(y)$.

Definition 1. *A hash function $h : U \rightarrow M$ is **perfect** for a set S if it causes no collisions for pairs in S .*

For any given S such that $|S| \leq m$ there is a perfect hash function.

For any S such that $|S| > m$ there is **no** perfect hash function.

If $|U| > m$ there is no perfect hash function for all $S \subset U$, s.t. $1 < |S| \leq m$.

Chaining

$h(.)$ - hash function.

A table $T[1..m]$ such that $T[k]$ is a pointer to a linked list of all the elements hashed to $T[k]$.

Insert k : add k to the linked list $T[h(k)]$.

Search/delete k : search (+ delete) in $T[h(k)]$.

The cost is proportional to the length of the link lists.

Hash Functions

$$h(k) = k \bmod m$$

$$h(k) = (ak + b) \bmod m,$$

$$H = \{h(k) \mid 1 \leq a \leq m - 1, 0 \leq b \leq m - 1\}$$

If m not a prime, let $p > m$ be a prime

$$h(k) = ((ax + b) \bmod p) \bmod m$$

Analysis of Hashing with Chaining

Let n be the number of keys stored in the table.

The **load factor** $\alpha = \frac{n}{m}$.

Worst case insert time $O(1)$.

Worst case search/delete time $O(n)$.

For simple probabilistic analysis:

Uniform Distribution Assumption: Keys are hashed to uniformly random and independent locations.

Assume that $h(\cdot)$ can be computed in $O(1)$ time.

Theorem 1. *In a hash table in which collisions are resolved by chaining, under the assumption of simple uniform hashing, any (unsuccessful or successful) search takes $\Theta(1 + \alpha)$ expected time.*

Proof. For $j = 0, 1, \dots, m - 1$, let n_j denote the length of the list $T[j]$.

$$E[n_j] = \alpha = n/m.$$

1. If a key k is not in the table, it is equally likely to hash to any of the m slots. The expected time to search to the end of the list $T[h(k)]$ is α . Including the time to compute $h(\cdot)$ the total expected time is $\Theta(1 + \alpha)$.

2. For a successful search we assume that all keys are equally likely to be searched and compute the average time to search a key.

Assume that a key is inserted at the head of the link list.

Let k_i denote the i th key inserted into the table. For keys k_i and k_j , we define the indicator r.v. X_{ij} to indicate whether $h(k_i) = h(k_j)$.

$$E[X_{ij}] = 1/m.$$

The expected number of elements examined is:

$$\begin{aligned} & E \left[\frac{1}{n} \sum_{i=1}^n \left(1 + \sum_{j=i+1}^n X_{ij} \right) \right] \\ &= \frac{1}{n} \sum_{i=1}^n \left(1 + \sum_{j=i+1}^n E[X_{ij}] \right) \\ &= \frac{1}{n} \sum_{i=1}^n \left(1 + \sum_{j=i+1}^n 1/m \right) \\ &= 1 + \frac{1}{nm} \sum_{i=1}^n (n - i) \\ &= 1 + \frac{1}{nm} (n^2 - n(n + 1)/2) \\ &= 1 + \frac{n-1}{2m} = 1 + \frac{\alpha}{2} - \frac{\alpha}{2n} \end{aligned}$$

Thus total time needed is $\Theta(1 + \alpha)$. \square

Universal Hash Functions

Definition 2. A family H of hash functions from U to M is **universal (2-universal)** if for all $x, y \in U$, such that $x \neq y$, and for a randomly chosen function h from H

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}.$$

Let H be the set of all functions from U to M , then H is universal.

Problem: There are m^u functions from U to M - requires $u \log m$ bits to choose, represent and store as a table.