Hash tables

$h_0(x) = (h(x) + f_0) mod\, m,\ \ f_i = i$ # Linear probing

$f_i = ih^{+}(x),\ \ \ h^{+}(x) = R - (x\, mod\, R),\ \ \ prime\ R < m$

$f_i = i^2$ # Quadratic probing

For quadratic probing hash table of size $m > 3$, first $\lfloor \frac{m}{2} \rfloor$ probes would be distinct.

$\lambda = \frac{m}{n}$ # Load factor

Ideal: prime m, $\lambda = 1$ for open hashing, $\lambda = \frac{1}{2}$ for closed hashing.

Use separate chaining (linked list) when don't know # of insertion/deletion
Else, use closed hashing

Computational complexity

$f(x)\ is\ positive\ if\ f(n) > 0$
$f(x)\ is\ eventually\ positive\ if\ f(n) > 0\ for\ n \geq n_0$

$f(n) = O(g(n)) \Leftrightarrow f(n) \leq cg(n)\ for\ n \geq n_0$

$f(n) = \Omega(g(n)) \Leftrightarrow f(n) \geq cg(n)\ for\ n \geq n_0 \Leftrightarrow g(n) = O(f(n))$

$f(n) = \Theta(h(n)) \Leftrightarrow c_1 h(n) \leq f(n) \leq c_2 h(n)\ \ for\ n \geq n_0\ and\ positive\ c_1,\ c_2 \Leftrightarrow g(n) = \Theta(f(n))$

Growth rates:

c
$\log n$
$\log^2 n$
n
$n \log n$
n^2
n^3
2^n
$n!$
n^n

$f(n) = X(f(n))$

$f(n) = O(g(n), g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$

$f_1(n) + f_2(n) = O(max(g_1(n) + g_2(n))\ \ (min\ for\ \Omega,\ first\ for\ \Theta)$

f*f=X(g*g)

$f(n) = o(g(n)) \Leftrightarrow f(n) = O(g(n)),\ f(n) \neq \Theta(g(n))$

$f(n) = \omega(g(n)) \Leftrightarrow f(n) = \Omega(g(n)),\ f(n) \neq \Theta(g(n))$

$if\ \lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, then:

$0 \leq c < \infty \Rightarrow f(n) = O(g(n))$

$0 < c \leq \infty \Rightarrow f(n) = \Omega(g(n))$

$0 < c < \infty \Rightarrow f(n) = \Theta(g(n))$

$c = 0 \Rightarrow f(n) = o(g(n))$

$c = \infty \Rightarrow f(n) = \omega(g(n))$

Use L'Hopital's rule in the proof

$$\sum_{1\leq i\leq n} i = \frac{n(n+1)}{2}$$

$$\sum_{1\leq i\leq n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{1\leq i\leq n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$$

D_n # domain of inputs of size n for the algorithm

$C(I)$ # cost

$Pr(I)$ # probability I is in input

$R(n)$ # complexity for any input of size n

$$R_b(n) = \min_{I\in D_n} C(I)$$

$$R_w(n) = \max_{I\in D_n} C(I)$$

$$R_a(n) = \sum_{I\in D_n} Pr(I)C(I)$$

S_i # statement cost

$$T(n) = \sum_{1\leq i\leq n} cost(S_i)$$

$$T_w(n) = \sum_{i=1}^{n} C = Cn$$

$$T(n) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} + \sum_{k=1}^{n}\right)C = C\sum_{i=1}^{n}(i + n) = C\left(\frac{n(n+1)}{2} + n^2\right)$$

Trees

$Depth\ =\ |path\ from\ root\ to\ x|$
$Height\ =\ |longest\ path\ from\ x\ to\ any\ leaf|$
$Tree\ height\ =\ Tree\ depth$
$Height\ of\ empty\ tree\ =\ -1$
All nodes in a $k-ary$ try have at most k children ($k = 2 \Leftrightarrow binary\ tree$)
Complete binary tree is left-justified
$|h(T_L(x)) - h(T_R(x))| \leq 1$ # Balanced binary tree
Skew tree has one child in non leaf nodes
Full binary tree has 2 children on levels $[0, h-1]$

Traversals:
Preorder (root, L, R)
Postorder (L, R, root)
Inorder (L, root, R)
Level-order (L->R by level)

For el i in array binary tree: parent $\frac{i-1}{2}$, left $2i+1$, right $2i+2$
$i \in Leaf \Leftrightarrow 2i \geq n-1$

Binary Search Trees

$left\ nodes < root \leq right\ nodes$
Inorder traversal = sorted order
Can deserialize preorder traversal
Can balance by rebuilding from inorder serialization
$c_{i,j} = \min_{i \leq k \leq j} \{c_{i,k-1} + c_{k+1,j} + \sum_{l=i}^{j} p_l\}$ # min avg search cost
$c_{i,i} = p_i$
$c_{i+1,i} = 0$
Approach to compute $c_{1,n}$:

1. Compute $c_{i,i}\ for\ all\ i$
2. Compute $c_{i,j}$ in increasing difference of $(j-i)$

$t_{i,j} = k \Leftrightarrow x_k$ is the root of optimal BST
$\{x_i, x_{i+1},..., x_k, x_{k+1},..., x_j\}$
$for\ i = 1\ to\ n\ do$:
$c_{i,i} = p_i$

$$t_{i,i} = i$$

$$Cost = \sum_{i=1}^{n} p_i (d_{epth\, i} + 1)$$