bottom up DP using loops v/s memoization

- no recursion easier to maintain, debug
- does not use stack space, can't overflow.
- small problems are solved first & immediately. so, at all times, we have *some* result and the memoization can reuse those results

Hash Table $\alpha = \frac{n}{m}$ # of keys / table size

INSERT
SEARCH
DELETE

simple Uniform Hashing
all keys are equally likely to hash into each of m slots $(P = \frac{1}{m})$

Division hashing
hash = h(x) / P   P = Prime number
$P > m$
not close to 2^n

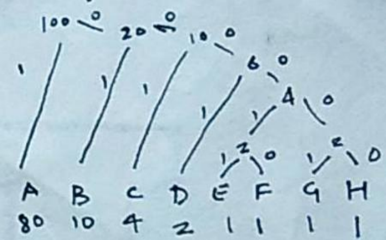
Multiplication hashing
A = const. in [0, 1]
hash = $\lfloor m (h(x) A \bmod 1) \rfloor$
* fractional part $kA - \lfloor kA \rfloor$

$m = 2^P$, P-bit hash
if w = word size, then
$A = \frac{s}{2^w}$ $0 < s < 2^w$

multiply k by $s = A 2^w = r_1 2^w + r_0$
resulting p-bit hash is p MSB of r_0

Universal hashing
pick a fn uniformly at random from a family s.t. $\Pr_{h \sim H} [h(k) = h(\ell)] \leq \frac{1}{m}$

Lossless compression
Huffman tree ← greedy
sort chars in desc. order of frequen
merge min 2



$\Theta(n \log n)$ using pqueue

Priority Queue
INSERT put leaf O(log n)
MAX $\Theta(1)$
EXTRACT-MAX swap w last O(log n)
INCREASE-KEY move up O(log n)

MAX-HEAPIFY O(log n)
BUILD-MAX-HEAP O(n)
$\frac{n}{2}$ to 1 heapify

i = left to right-1
mul left.first x i.second x right.second
+ opt-mul (left, i)
+ opt-mul (i+1, right)

$\underbrace{A_1 A_2 A_3}\ \underbrace{A_4 \ldots A_k}$
cost-of-combination + cost-of-parts
↑
DP

$f(n) \in O(g(n))$
iff $\exists\ c, n_0$ $0 \leq f(n) \leq c\, g(n)\ \forall n \geq n_0$

divide & conquer
- BSEARCH O(log n)
- INSERTION $O(n^2)$
- SELECTION $O(n^2)$
- MERGE-SORT O(n log n)
- QUICK-SORT $O(n^2)$
- HEAP-SORT O(n log n)

Minimum spanning Tree

spanning tree s.t. $\sum_{u,v \in T} w(u,v)$ is min.

Greedy approach

while not done

add a safe edge to MST

Safe edge = (u,v) such that $A \cup \{(u,v)\}$ is subset of some MST

Boruvka $(E \log V)$

for each vertex, pick an edge of minimum weight adjacent to it

contract all connected components

Prims $V \log V + E \log V$

$O(E \log V)$

start with a vertex.

pick min weight ~~edge~~ vertex

repeat

update distance to all adjacent vertices

pick min weight vertex

Relaxation step distance to tree

if $v \in Q$ & $w(u,v) < v.key$

$v.\pi = u$

$v.key = w(u,v)$

Kruskal's $E \log E + E \log \alpha(V)$

$= O(E \log V)$

each vertex is a component

repeat

pick min weight edge connecting 2 components

Light edge lemma: if $A \subseteq$ some MST s.t.

$(S, V \setminus S)$ cut respects A

then any light edge for $(S, V \setminus S)$ is safe

Dijkstra's

init (G, w, s)

$S = \phi$, $Q = V$ (S = explored, Q = unexplored)

while Q

$u = $ EXTRACT-MIN (Q)

$S = S \cup \{u\}$

for $v \in$ Neighbors of u Relaxation step

if $v.d > u.d + w(u,v)$

$v.d = u.d + w(u,v)$

$v.\pi = u$

OR for each vertex u in topological order

for each $v \in$ neighbors u

if $v.d > u.d + w(u,v)$

$v.d = u.d + w(u,v)$

$v.\pi = u$

Graph $G = (V, E)$

Search

init $u.color = WHITE$, $u.d = \infty$, $u.\pi = NIL$

$\forall u \in V \setminus \{s\}$

$s.color = GRAY$ $s.d = 0$, $s.\pi = NIL$

BFS

```
while Q
    u = DEQUEUE(Q)
    for v in neighbours of u
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

$O(|V| + |E|)$

$O(|V|)$

DFS

```
time++
u.d = time
u.color = GRAY
for each v in neighbors of u
    if v.color == WHITE
        v.π = u
        DFS(G, v)
u.color = BLACK
time++
u.f = time
```

$O(|V| + |E|)$

$O(|V|)$

Parenthesis: if $[u.d, u.f] \subseteq [v.d, v.f]$ then u is descendant of v.

White Path: v is descendant of u

$\Leftrightarrow$ when u turns gray, there is a white path from u to v

edges: undirected graphs only have tree or back edges (no forward edge)

Topological sort (DAG) $O(|V| + |E|)$

output a list of vertices s.t. if uv is an edge then u is before v in the list

DFS(G) $\Rightarrow$

$(-v.f)$ order is topological order

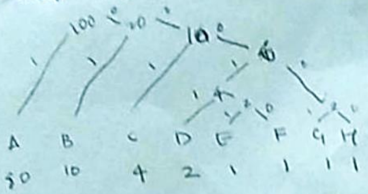
strongly connected component

maximal subset $C \subseteq G$ st. $\forall u, v \in C$ there is a path from u to v

call DFS(G^T) but vertices are in topological order (G) decreasing $v.f$

each tree in DFS forest is a SCC

huffman tree



Division method.

$h(k) = k \bmod m$

avoid : $m = 2^p - 1$ for base 2^p

ex. 127 for ASCII characters

good : prime m not close to power of 2

linear $h(k) + i$

quad $h(k) + c_1 i + c_2 i^2$

double $h_1(k) + i h_2(k)$

uniform hashing

each key k is equally likely to have any one of $m!$ permutations as probing sequence

probing seq $\sim$ Uniform (Permutations)

Multiplication method.

$h(k) = \lfloor m (kA \bmod 1) \rfloor$

$= \lfloor m (\text{frac } kA) \rfloor$

$= \lfloor m (kA - \lfloor kA \rfloor) \rfloor$

m : 2^p (p-bit hash)

w : word size (64 bit = 8 B)

A : $\in [0, 1]$ $\frac{s}{2^w}$ $s \in (0, 2^w)$

$kA = \frac{ks}{2^w} \Rightarrow \frac{1}{2^w}(r_1 2^w + r_0)$

take p MSB of r_0

Universal hashing

$\Pr_{h \sim H} [h(k) = h(l)] \leq \frac{1}{m}$

$E[n_{h(k)}] = \alpha$

$E[n_{h(k)}] = 1 + \alpha$

TO

hash - table

U = universe of keys

K = set of keys used

T = table

m = # slots in T

h = hash function (ideally a random function)

$\alpha = \frac{n}{m}$ load factor n = # keys in T

Simple uniform hashing

$P(\text{key being hashed to slot } i) = \frac{1}{m}$

hash values of different keys are independent

SEARCH : $O(1+\alpha)$

Universal hashing

H = set of hash functions

$\forall k, l \in U$: # hash functions s.t. $h(k) = h(l) \leq \frac{|H|}{m}$

$\Pr_{h \sim H} [h(k) = h(l)] \leq \frac{1}{m}$

if h is sampled from universal family

k not in T $E[n_{h(k)}] \leq \alpha$

k in T $E[n_{h(k)}] \leq 1+\alpha$

ex. p : prime $p > m$

$Z_p = \{0, 1, \ldots p-1\}$ $Z_p^* = \{1, 2, \ldots p-1\}$

$h_{ab} = ((ak+b) \bmod p) \bmod m$ $a \in Z_p^*$, $b \in Z_p$