Hashing

Birthday Problem:

n people $1 \cdot (1-\frac{1}{365}) \cdot (1-\frac{2}{365}) \cdots (1-\frac{n-1}{365})$

when $n=23$, then prob $\approx 1/2$

Balls into Bins:

linearity of expectation: for random X and Y, $E[X+Y] = E[X]+E[Y]$

n balls into m bins, expected # of empty bins = consider 1st bin. If it's empty, then all n balls missed.

$P(\text{ball miss}) = 1-\frac{1}{m}$

$P(\text{all n ball miss}) = (1-\frac{1}{m})^n$ by lin. expectation: empty bins $m(1-\frac{1}{m})^n$

# of bins w/ 3 balls: By indp, $P(\text{bin has 3 specific balls}) = (\frac{1}{m})^3(1-\frac{1}{m})^{n-3}$

Consider the 1st bin:

$P(\text{3 balls go in}) = \binom{n}{3}(\frac{1}{m})^3$

$P(\text{n-3 balls miss}) = (1-\frac{1}{m})^{n-3}$

now choose 3 balls into bin: $\binom{n}{3}$ choices of 3 balls.

$P(\text{1st bin has 3 balls}) = \binom{n}{3}(1-\frac{1}{m})^{n-3}(\frac{1}{m})^3$

w. lin expect: $m\binom{n}{3}(1-\frac{1}{m})^{n-3}(\frac{1}{m})^3$

Bloom Filters: trade correctness with space/time!

m bits in hash table, k hash functions $(f_1, \dots, f_k)$

if $y \in S$, then $f_1(y), \dots, f_k(y) = 1$ | if $y \notin S$, $f_1(y), \dots, f_k(y) = 0$

Ex: $m=7$, $k=2$, $S = \{x_1, x_2\}$ => $f_1(x_1)=1$ $f_1(x_2)=5$ $f_1(x_3)=5$; $f_2(x_1)=4$ $f_2(x_2)=4$ $f_2(x_3)=1$

① insert x_1:

A:	0	1	2	3	4	5	6
	0	1	0	0	1	0	0

② insert x_2:

A:	0	1	2	3	4	5	6
	0	1	0	0	1	1	0

③ find x_3: false positive!

$P(\text{false positive})$: occurs when all k indices checked are 1s. $P(\text{bit}=0) = (1-\frac{1}{m})^{kn}$

$P(\text{k indices are 1}) = (1-(1-\frac{1}{m})^{kn})^k$

Another example: Partitioned Bloom Filter: m total bits, k mini tables. i^{th} hash only hashes to mini table.

Runtime: insert/query $O(k)$, space $O(m)$

$fpp \approx 1-e^{-kn/m}$

hash/chain: insert/query $O(1+\alpha m)$

Space $O(\text{elements})$, 0 fpp.

So what?

↑ m => ↑ space => ↓ FP

↑ k => ↑ # of 1's => ↓ # of 0's => ↑ computation time

Still can be good since ? Chances to reject x by hashing to 0.

Hashing w/ Chaining

$O(1)$ insertion, $O(1+\frac{n}{m})$ lookup → $\frac{n}{m}$ expected elem. hashing in $O(1)$

Store heads of linked list at every index

n elem, m linked lists

however, want $\frac{n}{m} \approx 1$, but don't know n in advance. So use:

Table Doubling

$n < m$ maintained as invariant, when $n=m$,

when $n=m$, reconstruct entire dataset with size $2m$ and rehash all n elem. to new table.

Expected hash = $O(1)$, Traverse L.L $O(\frac{n}{m}) = O(1)$

Amortized inserts = $O(1)$ pf: "by budget", "cheap" (no table-doub = $O(1)$, w/ t-d = $O(n)$)

insertion store time credit for "expensive" insertion. Every insert takes $2n$, finance with $4 \cdot \frac{n}{2} = 2n$ from prev. $\frac{n}{2}$ insert.

Fingerprinting: Ctrl+F!

① hash sets of $|P|$ consecutive char into a value by taking mod w/ respect to some prime p.

② Compare hashes to hash value of P to see if there's a match.

Ex Let $|P|=5$ and $p=13$. Find fingerprint for 31415 2.

$31415 \bmod 13 = 7$

$14152 \bmod 13 = 8$

fingerprint: 7,8 ↑ Successive calc = $O(1)$

Optimize!

> Don't naively take mods

> Every time read char in a file, change at leftmost digit(a) removed and rightmost digit (b) is inserted.

> Can get new hash in $O(1)$:

$N' = [10 \cdot (N - 10^{|P|-1} \cdot a) + b] \bmod p$ (old hash)

↑ can be bad!

> collisions: string s hash to same value of p, but $s \neq P$

> more accurate by: hash sets of $|P|$ by taking mod $p_1, p_2, \dots$ (multiple)

> tradeoff? space - accuracy

Set Resemblance: are 2 documents similar to each other?

Resemblance: in 2 sets A and B $R(A,B) = \frac{|A \cap B|}{|A \cup B|}$ (or Jaccard coef.)

$R(A,B) = 0$ when disjoint, $=1$ when same.

$O(n^2)$ naively compare each A to B)

$O(n)$ w/ hashing

Random Permutations π_i: some rand. perm.

need "black box" to output permutation (i.e. $BB(1,x) = \pi_1(x)$, $BB(50,x) = \pi_{50}(x)$)

$\pi_i(A)$ = set of elements from $BB(i,x)$ for every x in A.

Shingling: hash k consec. words/elem into 64-bit hash to get smaller set.

down sim => set resemb.

To compare doc, compare document sketches, $\{\min\{\pi_1(S_D)\}, \dots, \min\{\pi_n(S_D)\}\}$

Ex) $\pi_1(S_D) = \{7, 14, 12, 3\}$ and $\pi_2(S_D) = \{1, 7, 11, 14\}$

Sketch is min, so $\{3, 1\}$.

Ex) $A = \{1,2,3,4\}$, $B = \{1,2,5,6\}$ | $A = \{1,2,3,4\} \Rightarrow A' = \{102, 33, 50, 740\}$, $B = \{1,2,5,6\} \Rightarrow B' = \{102, 33, 61, 34\}$

Visually, shared are $\{1,2\}$

now rand. perm. Hash each elem w/ some hash || Now compare minimum value of each set

$33 = 33$, so return 1. if diff. hash, could return 0.

Performing 1): set resemb.

if resem between 2 Doc $R(A,B)=0$, then set resem algo gives correct estimate of resemb:

$R(A,B)=0$, then $\frac{|A \cap B|}{|A \cup B|} = 0 \Rightarrow |A \cap B| = 0$.

thus 2 disjoint sets, since disj, then $\min(\pi_i(A)) \neq \min(\pi_i(B))$, ($\pi_i(A)$ and $\pi_i(B)$ don't share elem)

no min => # match / #perm = 0.

Primality Testing (quickly check if # is prime)

Fermat's Little Thm:

if p is prime and $1 \leq a < p$, $a^{p-1} = 1 \bmod p$

ex. if $p=7$ and $a=3$, $3^{7-1} = 3^6 = 729 = 1 \bmod 7$

$a^{pq} = a \bmod pq$

Carmichael num: some composite n that are a-pseudoprime for all a. {ex: 516 1105, 1729, ... inf many}

Fermat Test:

① For prime candidate n, pick $1 < a < n$

② Calculate $a^{n-1} \bmod n$

③ if $a^{n-1} = 1 \bmod n$, then n is a-pseudoprime. else n is composite.

ex) $n=299$, $a=116$

$116^{299-1} = 116^{288} = 1 \bmod 299$.

Thus 299 is 116-pseudoprime

Rabin-Miller Testing: more complex prime test

fact 1: if p is odd prime, $x^2 = 1 \bmod p$ only has 2 solutions mod p, $x = \pm 1 \bmod p$

fact 2 (contrapos): If there is a nontriv sq. root $1 \bmod n$ (x s.t. $x \neq \pm 1 \bmod n$ but $x^2 = 1 \bmod n$), then n is composite.

Steps:

① Find $n-1 = 2^u r$ $(n > 2)$

② For some a, calculate a^r and compute subsequent squares $(a^{2r}, a^{4r}, \dots)$

③ If at any step i we have

a) $a^{2^{i-1}r} \neq \pm 1 \bmod n$

b) and square $a^{2^i r} = 1 \bmod n$

then nontriv sq. rt. $1 \bmod n$ and n is comp => a is a witness.

④ Repeat above for t steps until $a^{2^t r}$

Then by fermat, $a^{2^t r} = a^{n-1} \neq 1 \bmod n$, so n is composite.

- can do test on multiple values of a
- Searching randomly for a witness.
- efficient: randomly selected a will be a witness w/ prob. $\geq \frac{1}{4}$.
  - ↳ if n prime, RM says prime.
  - if n composite, RM labels as compos. w/ prob $1-\frac{1}{4^k}$ for k trials

Ex) is 53 prime?

$n=53$, $a=2$

1) $n-1 = 2^u \cdot r$

$52 = 2^u \cdot r$

2) compute squares:

$\frac{52}{2^1} = 26$, $\frac{52}{2^2} = 13$, $\frac{52}{2^3} = 6.5$ (not whole, so stop)

take 1 before

③ check $b_0 = a^r \bmod n$

$52 = 2^2 \cdot 13$ $t=2$

$52 = 2^t \cdot u$ $u=13$

$b_0 = 2^{13} \bmod 53 = 30 \bmod 53$

want to know if $b_0 \pm 1 \Rightarrow n$ is prime

here, $b_0 = 30$, so not prime

$b_1 = b_0^2$; $30^2 \bmod 53 = -1 \bmod 53$

$b_1 \to +1$: compos

$b_1 \to -1$: prime, it's prime w/ 2 as witness.

Euclid's GCD:

```
func Euclid(a,b):
    if b=0, return(a)
    return (Euclid(b, a mod b))
```

Ex) gcd(10, 45)

$45 = 10 \cdot 4 + 5$ (mod; times 10 into 45; remainder)

$10 = 5 \cdot 2 + 0$ → GCD once 0, gcd is prev remainder.

Extended Euclid's

→ given a,b, lets us calculate gcd(a,b) and x,y s.t. $ax+by=d$.

```
if b=0, then return (a,1,0)
end if k = (a - a mod b)/b = a/b
(d, x, y) s.t. bk+a mod b
return (d, y, x-ky)
```

Extended-Euclid's Example

let a=51 and b=20

a	b	k	x	y
51	20	2	-9	23
20	11	1	5	-9
11	9	1	-4	5
9	2	4	1	-4
2	1	2	0	1
1	0		1	0

from bottom-up: $k = \frac{a}{b}$; $x = \text{prev. } y$; $y = \text{prev } x - \text{current } k \cdot \text{prev. } y$

always 1 since base case.

$51 = 20 \cdot 2 + 11$

$20 = 11 \cdot 1 + 9$

$11 = 9 \cdot 1 + 2$

$9 = 2 \cdot 4 + 1$ ← gcd

$2 = 1 \cdot 2 + 0$

return (d, y, x-ky)

thus $-9(51) + 23(20) = 1$

RSA

algo: Suppose Oliver wants to send Lisa msg.

① Lisa finds 2 large primes $p \neq q$ and computes $n = pq$

② Lisa chooses odd $e < n$ s.t. $\gcd(e, (p-1)(q-1)) = 1$

③ Lisa calculates d to be multip. inv. of e

i.e. $d = e^{-1} \bmod (p-1)(q-1)$ s.t. $de = 1 \bmod (p-1)(q-1)$

[Extended Euclid] ↳ basically $ex + (p-1)(q-1)y = 1$ but $x = d$.

④ Public key is $K_E = (n, e)$

Private $K_D = (p, q, d)$

now Oliver knows $K_E = (n, e)$ and he encodes msg m into ciphertext $c = m^e \bmod n$.

Then Lisa calculates $c^d \bmod n$.

Ex Run RSA; $p=11$ and $q=29$, $e=3$

what is public/private keys? if m=100, c=?

$n = pq = 319$ and $(p-1)(q-1) = 280$

$e=3$ since $\gcd(3, 280) = 1$

Use Extended Euclid's → $d = 187$

a	b	k	x	y
280	3	93	1	-93
3	1	3	0	1
1	0		1	0

$[-93 \cdot 3 + 1 \cdot 280 = 1$

$\Rightarrow -93 = 187 \bmod 280]$

$K_E = (319, 3)$

$K_D = (11, 29, 187)$

$c = m^e \bmod 319 = 100^3 \bmod 319 = 254$

check w/ $c^d \bmod n = 254^{187} \bmod 319 = 100$ ✓

2-SAT

ex $(\bar{a} \vee b) \wedge (b \vee \bar{c}) \wedge (a \vee c)$ (or, and)

1. implication graph
   - → each 2-sat makes 2 implications
     ex) $\bar{a} \vee b$ { if $a=T, b=T$ $(a \Rightarrow b)$; if $\bar{b}=T, \bar{a}=T$ $(\bar{b} \Rightarrow \bar{a})$ }
   - → construct graph w/ v: each literal and neg
     ex) $\bar{a} \vee b$: $a \to b$, $\bar{a} \to \bar{b}$
   - → formula satisfiable if and only if graph doesn't have literal and its negation on SCC.

2. random algo
   - → start w/ random assignment, take formula that's unsat and flip.
   - → repeat till bored; if sol not found return not found. → only false neg possible (no sol when was sol)

$O(n \cdot m)$ (# literal, # clause)

if ∃ sol, algo finds in m expected $O(n^2)$ flips (# vari)

prob of not finding satisfying in $4n^2$ flips lower bounded by $1/4$. Plus sol after kn^2 steps $\leq \frac{1}{k}$

Applic. to 3SAT:
- pick rand assign; for $3n$ steps, choose unsat and flip. if by end of $3n$, none found, start over w/ new rand assign. will find (if one exists) in $(\frac{4}{3})^n$ (local, little)

P vs. NP

P: set of all yes/no problems that can be deterministically solved in polynom time.

NP: set of yes/no problems that can be verified w/ a polynomial length certificate. (nondet. (random) poly time)

P is a subset of NP since can be verified in polynom. by running it.

Problem is NP-complete if $X \in NP$ and $\in$ NP-hard

NP-Hard if every problem $y \in NP$ reduces to x.

Ex of NP: compositness, 3SAT, indp. set, vertex cover.

3-SAT: sol to $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$
- witness: assign of T or F
- poly verif: plug witness into 3SAT

Indpt. Set: is there indpt. set of size k
- witness: set of k vari that form indpt. set
- poly. time. veri: ensure no 2 verti have edge

Reduction: if we can solve B, then can solve A. (transform input of A to input of B.)

→ A is hard: Reduce smth that is hard to it.

Proving

- ∈ P: come up w/ poly algo and correctness (if and only if)
- ∈ NP: state poly-length witness & how to verify
- NP-Hard: reduce known NP-hard problem
  1) Show that reduction is polynomial
  2) prove it's correct
- NP-complete: Both NP and NP-hard.

Heuristic for NP-Hard

Local search:
1. Define a cost function f, lets say goal is to minimize f.
2. Define neighborhood function N. $N(x)$ must be:
   - easily computable from x
   - $N(x)$ is polynomial in input size
   - if $y \in N(x)$ then $x \in N(y)$
3. pick starting point x
4. while there is $y \in N(x)$ s.t. $f(y) < f(x)$, $x=y$
   return sol

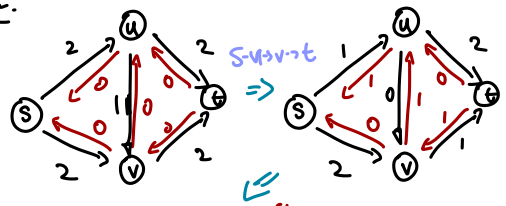
variations:
- Hill Climbing: generic local search
- Metropolis: pick random $y \in N(x)$. If $f(y) < f(x)$, $x=y$. $f(y) > f(x)$, $x=y$ w/ some prob.
- Simu. Annealing: same as metropolis, prob of moving to higher cost neighborhood decreases over time.
- Tabu Search: same as metropolis/SA, but w/ memory so won't get stuck.
- Parallel Search: do more than 1 search, occasionally replace veering doing poorly w/ ones doing well.
- Genetic Algo: keep pop. of searches that changes over time via "breeding" of best performing searches.

Ford-Fulkerson:

1. Let residual graph start as original
2. at each step from $s \to t$ in orig graph, increase flow.
   - in resid. graph, take path, reverse flow, and add edge.
3. repeat until no paths in residual graph

$O(Ef^n)$: every time augment path, increase flow
- augment path, at most $O(f^n)$, f^n: value of max fl.
- find any path in $O(E)$ w/ DFS
- BFS instead, $O(VE^2)$, called Edmonds Karp

F-F:



when flow backward, mins.

Approximation Algs.

k-approx: cost of solution produced by approx algo c is within a factor of k of the cost of an optimal sol.

examples:

> minimum vertex cover: Given graph $G(V,E)$, find min vertices $S \subseteq V$ s.t. $\forall (u,v) \in E$, have at least 1 of $u \in S$ or $v \in S$.

2-approx: choose edge at random, add both to cover, delete vertices (and edge) from graph and repeat. 2-approx since at worst, add 2 vert. per edge to cover. In opt, only 1 vertex per edge.

$|OPT| \leq |S| \leq 2|OPT|$

> max cut: $G(V,E)$, 2 disjoint subsets S and T s.t. $S \cap T = \emptyset$, $S \cup T = V$, and num of edges that crosses cut is maximized. Partition into 2 sets w/ max edges.

1/2 approx algo:
1. randomly assign node to S or T s.t. each edge has 1/2 prob. of crossing the cut.
2. Starts w/ all vertices on one side of cut, switch vertex if it increases num of edges crossing cut till sol'n can't inc. Local search algo.

> Euclidean Traveling Salesman: In set of points (x_i, y_i) in a Euclidean plane, find the tour of the min. length that travels thru all cities.

greedy: run DFS on MST and skip vertices that have already been visited (shortcutting). 2-factor since DFS at most visits edge 2x. Shortcutting reduces travel time due to triangle ineq. Opt. tour is at least size of MST since tour must be spanning tree (w/ last edge back to origin)

$\text{tour}_{\text{greedy}} \leq 2T(\text{after shortcut}) \leq 2\,OPT$

can improve to 3/2 approx, or Christofides-Serdyukov algo.

> max-SAT: most clauses that are satisfiable?

Flip random coin. If has k literals, clause is true w/ prob. 1/2, so prob. not satisf. is $1-2^{-k}$, where k = # literals.

Linear Programming

geometrically:
- > Lin eq $c_1x_1 + \dots + c_nx_n = C$ determines hyperplane in n-dim space.
- > constraint $c_1x_1 + \dots + c_nx_n \leq C$ defines halfspace
- > set of points satisfying all constraints = intersec. of half-spaces = polyhedron.
- > optimum happens at the corner → simplex algo.

LP-proof: > s.t. objective is what you want to min/max
> s.t. constraints define over what you're optm.

Integer Linear Problem: Find an integer assign. that maximizes a given objective function while respecting given constraints.
- variables are integral - NP hard

Max Flow Min Cut:

- cut partitions into S and V-S.
- weight of cut = sum of all edges crossing

Min cut ≥ max flow

If I have cut, all units of flow must pass thru S and V-S. If weight of cut is C, then at most C units of flow can pass between s and t. S-T = bottleneck.

Min cut ≤ max flow

At end of F-F, look at all nodes from S and all nodes that reaches t in residual graph. no path from s to t or else algo hasn't term. Some cut of graph and weight is max flow, so min cut can be bigger.

intuition: max flow push across the smallest capacity cut.

Dijkstra's Routing: