SILVESTRI LUCA 502524

Ex 1

$p -> pitom
it -> pitom
to -> pitom
pl -> pitom
om -> pitom, zoom
$d -> dad, daddy
da -> dad, daddy
ad -> dad, daddy
dd -> daddy
dy -> daddy
$z -> zoom
zo -> zoom
oo -> zoom

Soft intersection over 2-gram of P gives { pitom : 2, zoom : 1 }
Since |P| = 4 we get L - K · e = 4 - 2·1 = 2 so only 'pitom' satisfy the condition
~~Edit distance check~~ But the edit distance between P and 'pitom' is 2, so also 'pitom' is discarded
and no results are returned

Ex 2

rsync : sia C in possesso di F_old, S di F_new, C partiziona F_old in blocchi di dimensione B = 3,
ne calcola la funzione hash ed invia tali valori ad S

"can" -> h1, "e g" -> h2, "att" -> h3, "o o" -> h4, "rso" -> h5

S calcola un rolling hash di F_new, ogni volta che un valore corrisponde lo invia a C e salta B
caratteri in avanti, altrimenti invia il primo carattere del blocco

risposta di S : m, h3, o, ' ', h1, e, ' ', r, h3, o

zsync : S partiziona F_new in blocchi di dimensione B, ne calcola hash e salva tali valori su file .zsync
C richiede tale file e calcola rolling hash di F_old, per poi inviare ad S una bitmap dei valori condivisi

hash F_new : "mat" -> h1, "to " -> h2, "can" -> h3, "e r" -> h4, "att" -> h5, ~~"to$" -> h6~~
"o$$" -> h6

~~bitmap di risposta : 011011~~

~~S utilizza quindi algoritmo delta compression su F_new a partire da blocchi condivisi~~

~~to canatto$$| matto cane ratto -> < 0,0,m >~~
~~to canatto$$m|atto cane ratto -> < 7,4,' ' >~~
~~to canatto $$matto |cane ratto -> < 15,3,e >~~
~~to canatto $$ matto cane| ratto -> < 5,1,r >~~
~~to canatto $$ matto cane r|atto -> < 11,4,d >~~

~~C riceve tale elenco di triple e a partire da esso e dall'informazione sui blocchi condivisi ricostruisce F_new~~

Ex 3

Outdegree (15) = 11 ; Ref = 1
Copy - list : 011011110
Copy - blocks : 0, 0, 1, 0, 3
Additional docID : 5, 12, 20, 24, 33

Ex 5

La struttura dati per la risoluzione di 1-edit match su query consiste, oltre al dizionario di partenza D, di un dizionario ausiliario D1 contenente tutte le possibili stringhe ottenibili dagli elementi di D a seguito della cancellazione di un carattere : tali stringhe punteranno ciascuna al proprio (o propri) elemento di origine in D

La query di una stringa P, $|P| = p$, consiste di $2p + 2$ ricerche
1 Una ricerca esatta di P in D
2 Inserzione : una ricerca esatta di P in D1
3 Cancellazione : p ricerche ~~di~~ in D, una per ciascuna stringa ottenibile da P a seguito di una cancellazione
4 Sostituzione : p ricerche in D1, una per ciascuna stringa ottenibile da P a seguito di una cancellazione

Ex 4

Ricordando che $J(P, Q) = |P \cap Q| / |P \cup Q|$ ricaviamo
$J(A, B) = 3/8$, $J(B, C) = 1/9$, $J(A, C) = 0$

Sapendo che, data una permutazione casuale π degli elementi $\{0, 1, \dots 11, 12\}$, ed indicando con $\pi_m(A) = \min \pi(A)$, $P(\pi_m(A) = \pi_m(B)) = J(A, B)$, possiamo stimare la Jaccard similarity tra i due insiemi utilizzando u volte una permutazione differente, ~~osservando se $\pi_m(A) = \pi_m(B)$~~ ottenendo $\Phi_u(A) = \{\pi1_m(A), \dots, \pi u_m(A)\}$, $\Phi_u(B) = \{\pi1_m(B), \dots, \pi u_m(B)\}$ e calcolando $|\Phi_u(A) \cap \Phi_u(B)| / u$ dove l'intersezione è operata componente per componente

Sia $\pi1(x) = 5x + 1 \bmod 12$ allora $\pi1(A) = \{6, 9, 7, 5, 10\}$, $\pi1(B) = \{11, 4, 9, 7, 0, 5\}$, $\pi1(C) = \{11, 2, 8, 1\}$
quindi $\pi1_m(A) = 5$, $\pi1_m(B) = 0$, $\pi1_m(C) = 1$ ciascuno diverso dall'altro

Sia $\pi2(x) = 7x + 3 \bmod 12$ allora $\pi2(A) = \{10, 7, 9, 11, 6\}$, $\pi2(B) = \{5, 0, 7, 9, 4, 6\}$, $\pi2(C) = \{5, 2, 8, 3\}$
quindi $\pi2_m(A) = 6$, $\pi2_m(B) = 0$, $\pi2_m(C) = 2$ ciascuno diverso dall'altro

La stima della Jaccard similarity tra gli insiemi dati è dunque pari a zero per ciascuna coppia
valore corretto per $J(A, C)$, approssimativo per $J(B, C)$, 'scorretto' per $J(A, B)$
La cardinalità esigua degli sketch, in questo caso, impedisce una stima accurata della Jaccard similarity

Ex 2 (cont.)

bitmap di risposta : 011011 c'era anche "att" = b_5

S calcola delta compressione dei blocchi non condivisi di F_new a partire dall'informazione nella bitmap

to cano $$ | mate ratt -> <a, 0, m>
to cano $$m | ate ratt -> <6, 1, t>
to cano $$ mat | e ratt -> <0, 0, e>
to cano $$ mate | ratt -> <11, 1, r>
~~to cano $$ mate r | att -> <5, 2, t>~~

C riceve tale elenco di triple, opera la decompressione e sfruttando l'informazione precedentemente ottenuta ricostruisce F_new