Task 5 Report:
Spelling checking & correction Algorithms

1. terminology

D = dict size

L = dict average word len

Q = docu size

N = docu average word len

M = docu average word edit dist

$R|_{R=k}$ = ratio of words with dist=k∈{0, 1, 2, 3, 3+} for docu

T = duplicate rate of docu words

Possible(len(word), dist) (53 * (n+dist) + 26)

= estimate possible dist=1 #words

HashTable = separating chain based hash table

$P_{larger}|_{dist=k}$=possible combination of word(len=n)with dist=k >D

2. Assumption

• (*) xor hash can perfectly distribute n keys for n bucket → insert and compare the bucket for separating chain ∈O(1) i.e. E(collision=k)∈O(1)

• (*) assume L == N for easier srtrcmp analysis

• asssume malloc() free()∈O(1)

3. HashTable Operation

- hash by xor_hashing (key)∈O(len(key))
- HashTable has ∈O(strcmp(key, query) + hash) as (*)
- HashTable put ∈O(strcmp(key, query) + hash) as (*)
- HashTable get ∈O(strcmp(key, query) + hash) as (*)
- make HashTable(List) by inserting each word in List
  ∈O(len(List) * HashTable put(List->word))

4. Search Operation

- strcmp(dict word, docu word)∈O(MIN(L, N))
- cal edit dist(key, query) ∈ O(L * N)
  ∈ $O(N^2)$ by (*)
- linear match word(dict, query) by comparing query with each word in dict
  ∈O(D * strcmp(dic->word, query))
- linear search word(dict, query) by calculate edit dist(dict->word, query) and chose one with min edit dist and return first found when multiple choices faces ∈$O(D * N^2)$
- generated dist1(query) find “deleted, replaced, inserted” possible combinaitons
  ∈O(Possible(query))
- generated dist>1(query, dist=k) based on previous generated combination
  $\in O(\prod_{k=1}^{k=dist} Possible(query, k-1))$
- linear search word for dist(dict, query, dixt=k) by comparing query with each word in dict and return when dist=k matched found ∈$O(D * N^2)$

5. Small tricks

• when linear search |diff(key, query)| > 3 implies edit dist > 3

• when linear search cal edit dist(key, query) for a given dist_bound, terminate calculation early when cur all elements in calculating row and column(from left top to right bottom layer by layer) > dist_bound.

• store search result to prevent extra time spent on duplicate queries

• assign pointer to HashTable key but not strcpy() to save malloc() and free() time.

6. Task 3 analysis:

Approaches:

1) linear search look up:

step1: create HashTable(hist) for storing search history ∈ O(1)

step2: for each word in docu **Loop(O(Q))**

(1) if query was not searched before ∈O(N + MIN(L, N))

(2) linear match word ∈$O(D* MIN(L,N)|_{search\ once}$[1])

(3) if this is a new search store {query:match}
∈$O(N + MIN(L, N)|_{search\ once})$

∴ Total(approach1 average/worst)

∈O(Q*(N+ MIN(L, N) + MIN(L, N)*D$|_{search\ once}$))

∈O(Q*D*N) as (*)

2) hash table look up:

step1: make HashTable(dict) for List(dictionary)
∈O(D * (N + MIN(L, N)))

step2: for each word in docu **Loop(O(Q))**

(1) HashTable(dict) has word ∈O(N + MIN(L, N))

∴ Total(approach2 average/worst)

∈ O(D(N + MIN(L, N)) + Q(N + MIN(L, N)))

∈ O((D + Q)*N) as (*)

7. Task 3 space analysis

Space(approach1) = HashTable(hist) + List(dist) + List(docu) ∈O(Q + D)

---

1 search once for duplicate query

Space(approach2) = HashTable(dist) + List(dist) + List(docu) $\in O(Q + D)$

8. Task 3 Comparisons

For space both approaches are the same but approach 2 requires less time complexity.

∴ prefer approach 2 than 1 as $O(Q+D) \leqslant O(Q * D)$

9. Task 4 analysis:

1) linear search look up:

step1: create HashTable(hist) for storing search history $\in$ O(1)

step2: for each word in docu **Loop(O(Q))**

(1) if query was not searched before $\in O(N + MIN(L, N))$

(2) linear search word $\in O(D * N^2|_{\text{search once}})$

(3) if this is a new search store {query:match} $\in O(N + MIN(L, N)|_{\text{search once}})$

∴ Total(approach1)

$\in O(Q*(N + MIN(L, N) + D * N^2|search\ once))$

$\in O(Q*D*N^2)$ as (*)

space $\in O(Q+D)$

2) linear combined with hash table look up:

step1: make HashTable(dict) for List (dictionary) $\in O(D * (N + MIN(L, N)))$

step2: create HashTable(hist) for storing search history $\in$ O(1)

step3: for each word in docu **Loop(O(Q))**

(1) if searched before then **output** and next[2] $\in T*O(N + MIN(L, N))$

(2) if HashTable(dict) has query then **output** and next $\in (1-T)* \boldsymbol{R}|_{R=0} *O(N + MIN(L, N))$

(3) if HashTable(dict) has generated dist1(query) then **output** and next $\in (1-T)* \boldsymbol{R}|_{R=1} *O(N * (N + MIN(L, N)))$

(4) if Posssible combination for dist=2 < D[3]

if HashTable(dict) has generated dist>1(query, 2) then **output** and next $\in (1-T)* \boldsymbol{R}|_{R=2} *O(N^2 * (N + MIN(L, N)))$

else

linear search word for dist(dict, query, 2) then **output** and next $\in (1-T)* \boldsymbol{R}|_{R=2} *O(D * N^2)$

(5)repeat previous with dist=3

generate and search: $\in (1-T)*\boldsymbol{R}|_{R=3} *O(N^3 *(N+MIN(L, N)))$

linear search: $\in (1-T)* \boldsymbol{R}|_{R=3} *O(D * N^2)$

(5) if this is a new search store {query:match} $\in O(N + MIN(L, N)|_{\text{search once}})$

∴ Total(approach2)

$\in O(D * N + Q * (T*N +$

$(1-T)*(R|_{R=0} *N +$

$R|_{R=1} *N^2 +$

$MIN((1-P_{larger}|_{dist=2})* R|_{R=2} *N^3, P_{larger}|_{dist=2}*D* N^2) +$

$MIN((1-P_{larger}|_{dist=3})* (N^3 + R|_{R=3} *N^4),$

$P_{larger}|_{dist=3} *(N^3 + D* N^2)))))$

average: with constants T, P dropped

$\in O(D*N + R_0*Q*N + R_1*Q*N^2 + R_2*MIN($

$Q*N^3+R_3*MIN(Q*N^3+Q*N^4,Q*N^3+D*Q*N^2), D*Q*N^2))$

worst $\in O(D*N + Q*N^3+ D*Q*N^2))$

that is dist=2 generated and looked up but no matched and then linear search

space $\in O(Q+D)$

10. Task 4 analysis

If we assume O(string) $\in$ O(1) then O(approach1) $\in$ O(QD), O(approach2) $\in$ O(Q+D). From the above analysis, ***Approach2 is better than Approach1 by experimental result and prevent linear search if generating and hash table lookup can save time, given R has a larger proportion for k $\in${0, 1, 2} than k=3 or greater than 3.***

jabberwocky.txt	Approach1	Approach2
100K	4s	0.12s
250K	6s	0.3s
1M	8s	1.6s
5M	12s	4.0s

by dimefox user+sys

[2] P(searched before) = T/Q print search result and "continue;" for next docu word

[3] ***$(1-P_{larger}|_{dist=2})$***