

MATH-578A: Homework # 1

Due on Tuesday, March 10, 2015

Saket Choudhary
2170058637

Contents

| | |
|------------------------|---|
| Question # 1 | 3 |
| Question # 2 | 4 |
| Question # 3 | 5 |
| Question # 4 | 6 |
| Question # 5 | 7 |
| Question # 6 | 8 |
| Question # 7 | 9 |

Question # 1

Definition: $SP(i) = \max k < i$ such that $P[1..k] = P[i - k + 1..i]$

String: CACGCAACGA

NOTE: Iteration indexed at 0. So $SP[0] = 0$ (By Definition) and hence the loop iterations start from 1 and go till $n-1=9$;

| Iteration | $SP[i]$ | All other SP values examined | # of times inner while loop executed |
|-----------|---------|------------------------------|--------------------------------------|
| 1 | 0 | - | 0 |
| 2 | 1 | - | 0 |
| 3 | 0 | $SP[0]$ | 1 |
| 4 | 1 | - | 0 |
| 5 | 2 | - | 0 |
| 6 | 1 | $SP[0]$ | 1 |
| 7 | 1 | - | 0 |
| 8 | 1 | $SP[0]$ | 1 |
| 9 | 0 | - | 0 |

Question # 2

$S = CACGGCACGG$

NOTE: Indexing starts from 0. By definition $Z[0] = |S| = 10$

The 'cases' are chosen out of:

Case 1. $k > r$. The index for which Z value is being calculated is greater than the right most ending of all the previous (till $k - 1$) Z boxes calculated. Since this is as good as having no pre-calculated Z scores, this case leads to explicit character comparison (starting at k) till a mismatch occurs.

Case 2. $k \leq r$ The current position k is inside one of the previously calculated Z boxes. Hence there exists a corresponding position $k' = k - l + 1$ where l is the left ending of the Z box with its right ending at r , such that $S[k'] = S[k]$. There is a corresponding one to one match for $S[k'..r - l + 1]$ with $S[k..r]$ and we define this to be another box β with $\beta = r - k + 1$ and hence $Z[k]$ can be calculated utilising this information.

The following three cases arise:

Case 2a. $Z'_k < |\beta|$ So starting at k' the length of largest substring that matches the prefix of S is less than size of that β box starting at k' . Since this β box appears starting from k too and $Z'_k < |\beta|$ implies $Z_k = Z'_k$. Total comparisons:

1. Comparison: $k \leq r$
2. Assignment/Calculation: $k' = k - l + 1$
3. Lookup: Z'_k
4. Assignment/Calculation: $|\beta| = r - k + 1$
5. Comparison: $Z'_k < |\beta|$
6. Assignment: $Z_k = Z'_k$

No character comparisons are involved. All the above 'comparisons' are constant time.

Case 2b. $Z'_k > |\beta|$ So the substring starting at k' matches a prefix of S and has length equal to the β box. If we call the box with its leftmost end= l and rightmost end= r as α , then we know that $S[r+1] \neq S[l+\alpha+1]$ otherwise α would not have been the largest such box. Thus, $Z_k = \beta$ Thus no character comparisons involved in this case too.

The comparisons involved:

1. Comparison: $k \leq r$
2. Assignment/Calculation: $k' = k - l + 1$
3. Lookup: Z'_k
4. Assignment/Calculation: $|\beta| = r - k + 1$
5. Comparison: $Z'_k > |\beta|$
6. Assignment: $Z_k = Z'_k$

Again, all the operations are constant time.

Case 2c. $Z'_k = |\beta|$

The substring starting at k might have a matching prefix in S , and hence explicit character comparisons are required from $r + 1$ to $q \geq r + 1$ till the first mismatch occurs. These iterations are bound by $O(|S|)$ since the maximum possible mismatches are $O(|S|)$.

The comparisons involved:

1. Comparison: $k \leq r$
2. Assignment/Calculation: $k' = k - l + 1$
3. Lookup: Z'_k
4. Assignment/Calculation: $|\beta| = r - k + 1$
5. Comparison: $Z'_k == |\beta|$
6. Iteration for explicit character comparison: $while(Z[r + 1] == Z[\alpha + 1])...$, bounded by $O(|S|)$

Except character comparison step, rest all steps are constant time.

| i | $Z[i]$ | l_i | r_i | Case |
|-----|--------|-------|-------|------|
| 1 | 2 | 1 | 0 | 1 |
| 2 | 3 | 3 | 1 | 1 |
| 3 | 4 | 3 | 0 | 1 |
| 4 | 5 | 4 | 0 | 1 |
| 5 | 6 | 10 | 5 | 1 |
| 6 | 6 | 10 | 0 | 2a |
| 7 | 6 | 10 | 1 | 2a |
| 8 | 6 | 10 | 0 | 2a |
| 9 | 6 | 10 | 0 | 2a |

Question # 3

In order to determine if α is a circular rotation of β , we make the following observations:

1. All $|\beta| + 1$ possible length $|\beta|$ substrings possible of length $\beta\beta$ represents all possible circular rotations of β . This is intuitive, since a circular rotation would involve concatenating the start of string to its end
2. The next step involves searching for P in TT . This is possible in linear time using either Z algorithm or any other linear time exact matching algorithm.
3. If P appears in TT , the Z values the indices of its start/end can be determined by querying all those points in the Z value array, which exceed $|\alpha|$

The pseudocode is listed as Algorithm 1.

Algorithm 1 Find circular rotation

Input: Two string α, β and a linear time algorithm say Z algorithm to solve exact string matching problem in linear time

Output: Determine if α is a circular rotation of β

```

 $S \leftarrow \alpha\beta$ 
 $Z_{values} \leftarrow Z(S)$ 
 $N \leftarrow |S|$ 
while  $N \neq 3|S| + 1$  do
  if  $Z_{values}[i] \geq |\alpha|$  then
    return true
  end if
end while
return false

```

Question # 4**Question 7:**

Case 2b of Z algorithm can be split into following sub cases: Case 2b $Z'_k > |\beta|$

Case 2c $Z'_k = |\beta|$

Let r denote the right most edge of the Z box(call it α) such that $k \leq r$. l denotes the left most edge of this Z box. When $Z'_k > \beta$, let $S[r+1] = X$ Let $k' = k - l + 1$ denote the cooresponding position(there is an α box that appears as the prefix of S by definition) in the prefix of S , such that $S[1...k']$ matches $S[l...k]$ and also $S[1...r-l+1]$ matches $S[l..r]$

Consider $r' = r - l + 1$ let $S[r'+1] = Y$, then $X \neq Y$, else the Z box would have been longer than $|\alpha|$, contrary to the definition.

Now consider $Z'_k > |\beta| \implies$ there exists a matching prefix of S for substring starting at k' which also implies that $S[Z'_k + 1] = S[r' + 1] = Y$ because Z'_k will be at least $|\beta| + 1$ in size.

Since $X \neq Y$, $Z_k = |\beta|$, because $|\beta|$ is the length of longest matching prefix given $S[|\beta| + 1] = S[r' + 1] \neq S[r + 1]$

Question 7:

No. there is no extra speedup if we take into consideration all comparisons.

Case 2a, 2b approach: Comparison required: 1 character comparison on failure of conditional check $Z_k < |\beta|$ Case 2a,2b,2c approach: Comparison required: 1 integer comparison $Z_k == |\beta|$

Question # 5**Observations:**

1. The first occurrence of parameters is very flexible, since they can be made to match to any other parameter.
2. Any parameter appearing more than once arises a constraint

Complexity:**Correctness:**

Approach:

Algorithm 2 Find multisets**Input:** String P, T**Output:** Find all p-matches of P in T in $O(P + T)$

```

 $m \leftarrow |P|$ 
 $n \leftarrow |T|$ 
 $lastParameterMap$ 
 $parametersTotal$ 
 $P' \leftarrow null$ 
 $S = P\$T$ 
for  $i \leftarrow 1$  to  $m + n$  do
  if  $isParameter(S[i])$  then
    if  $P[i]$  in  $lastParameterMap$  then
       $parametersTotal[i] \leftarrow parametersTotal[i - 1] + 1$ 
       $lastOccurenceAt \leftarrow lastParameterMap[P[i]]$ 
       $numParamsFromLastOccurrence \leftarrow parametersTotal[i] - parametersTotal[lastOccurenceAt]$ 
       $P' \leftarrow concat(P', numFromLastOccurrence)$ 
    else
       $lastParameterMap[P[i]] = i$ 
    end if
  else
     $P' \leftarrow concat(P', S[i])$ 
  end if
end for
 $z\_values \leftarrow ZAlgorithm(P')$ 
return all positions where  $z\_values \geq m$ 

```

Example: XYabCaCXZddbW

Question # 6**Observations:**

1.

Complexity:**Correctness:****Algorithm 3** Find multisets**Input:** String S, T**Output:** Find all substrings of T that are formed by characters of S*patternMap* \leftarrow *CreateFrequencyOfCharacters*(S)*longestSubstringPossible* \leftarrow []*m* \leftarrow |S|*n* \leftarrow |T|*i* \leftarrow 2*sum* \leftarrow 0**while** *i* \leq *m* **do** **if** *S*[*i*] *in* *sequenceMap.keys()* **then** *sequenceMap*[*S*[*i*]] \leftarrow *sequenceMap*[*S*[*i*]] + 1 **end if** **if** *patternMap*[*S*[*i*]] \geq 1 **and** *sequenceMap*[*S*[*i*]] $<$ *sequenceMap*[*S*[*i*]] **then** *sum* \leftarrow *sum* + 1 **else** *sum* \leftarrow *sum* - 1 **end if** *i* \leftarrow *i* + 1**for** *i* \leftarrow 2 to *n* - *m* **do** *next* \leftarrow *S*[*i* + *m*] **if** *sequenceMap*[*previous*] $>$ *patternMap*[*previous*] **then** *sum* \leftarrow *sum* + 1 **else** *sum* \leftarrow *sum* - 1 **end if** **if** *patternMap*[*next*] \geq 1 **and** *sequenceMap*[*next*] $<$ *sequenceMap*[*S*[*i*]] **then** *sum* \leftarrow *sum* + 1 **else** *sum* \leftarrow *sum* - 1 **end if** *longestSubstringPossible*[*i*] = *sum* *previous* \leftarrow *S*[*i*] **end for****end while***longestSubstringPossible*[1] = *sum**previous* \leftarrow *S*[1]

Question # 7**Observations:**

1.

Complexity:**Correctness:****Algorithm 4** Find occurrence of P in T in linear time using sp values**Input:** Strings P and T**Output:** Find all occurrences of P in T in linear time using *sp* values $S \leftarrow PT$ $sp_{values} \leftarrow SP_{Calculator}(S)$ $N \leftarrow |S|$ $P_{occurrences} = []$ **while** $N \geq |P| + 1$ **do** **if** $sp_{values}[i] \geq |P|$ **then** **if** $S[N] == P[|P|]$ **and** $S[N - |P|] == P[1]$ **then** $P_{occurrences}.push(i)$ $N \leftarrow N - |P|$ **else** $N \leftarrow N - 1$ **end if** **else** $N \leftarrow N - 1$ **end if****end while****return** $P_{occurrences}$