MATH-578A: Homework # 1

Due on Tuesday, March 10, 2015

Saket Choudhary
2170058637

Contents

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

continued on next page...

Question #2

$S = CACGGCACGG$

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

The 'cases' are choosen 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 previoulsy calculated Z boxes. Hence there exists a correpsonding position $k' = k - l + 1$ where l is the left ending of the Z box with it's 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 caculated 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/Caculation: $|\beta| = r - k + 1$
5. Comparison: $Z'_k < |\beta|$
6. Assigment: $Z_k = Z'_k$

No character comparisons are involved.

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 it's leftmost end=l and rightmost end=r as α, then we know that $S[r+1] \neq S[|\alpha|+1]$ 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/Caculation: $|\beta| = r - k + 1$
5. Comparison: $Z'_k > |\beta|$
6. Assigment: $Z_k = Z'_k$

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

The substring starting at k might have a matching prefix in S, and hence explicit character comparions 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/Caculation: $|\beta| = r - k + 1$

[Question #2] continued on next page. . .

Question # 3

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

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

Question # 4

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 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 failore of conditional check $Z_k < |\beta|$ Case 2a,2b,2c appraich: Comparison required: 1 integer comparison $Z_k == |\beta|$

Question # 5

Solution: 1. The first occurence 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
Approach:
Example: XYabCaCXZddbW

Question # 6

Question # 7

continued on next page...

Algorithm 2 Find multisets

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

```
m ⇐ |P|
n ⇐ |T|
lastParameterMap
parametersTotal
P' ⇐ null
S = P$T
for i ⇐ 1 to m + n do
  if isParameter(S[i]) then
    if P[i] in lastParameterMap then
      parametersTotal[i] ⇐ parametersTotal[i − 1] + 1
      lastOccurenceAt ⇐ lastParameterMap[P[i]]
      numParamsFromLastOccurnce ⇐ parametersTotal[i] − parametersTotal[lastOccuredAt]
      P' ⇐ concat(P', numFromLastOccurence)
    else
      lastParameterMap[P[i]] = i
    end if
  else
    P' ⇐ concat(P', S[i])
  end if
end for
z_values ⇐ ZAlgorithm(P')
return all positions where z_values ≥ m
```

---

continued on next page...

Algorithm 3 Find multisets

Input: String S, T
Output: Find all substrings of T that are formed by characters of S

```
patternMap ⇐ CreateFrequencyOfCharacters(S)
longestSubstringPossible ⇐ []
m ⇐ |S|
n ⇐ |T|
i ⇐ 2
sum ⇐ 0
while i ≤ m do
  if S[i]insequenceMap.keys() then
    sequenceMap[S[i]] ⇐ sequenceMap[S[i]] + 1
  end if
  if patternMap[S[i]] >= 1andsequenceMap[S[i]] < sequenceMap[S[i]] then
    sum ⇐ sum + 1
  else
    sum ⇐ sum − 1
  end if
  i ⇐ i + 1
  for i ⇐ 2 to n − m do
    next ⇐ S[i + m]
    if sequenceMap[previous] > patternMap[previous] then
      sum ⇐ sum + 1
    else
      sum ⇐ sum − 1
    end if
    if patternMap[next] >= 1andsequenceMap[next]] < sequenceMap[S[i]] then
      sum ⇐ sum + 1
    else
      sum ⇐ sum − 1
    end if
    longestSubstringPossible[i] = sum
    previous ⇐ S[i]
  end for
end while
longestSubstringPossible[1] = sum
previous ⇐ S[1]
```

continued on next page...

Algorithm 4 Find occurence of P in T in linear time using sp values

Input: Strings P and T

Output: Find all occurences of P in T in linear time using *sp* values

```
S ⇐ PT
sp_values ⇐ SPCalculator(S)
N ⇐ |S|
P_occurences = []
while N ≥ |P| + 1 do
  if sp_values[i] ≥ |P| then
    if S[N] == P[|P|]andS[N − |P|] == P[1] then
      P_occurences.push(i)
      N ⇐ N − |P|
    else
      N ⇔ N − 1
    end if
  else
    N ⇔ N − 1
  end if
end while
return P_occurences
```

continued on next page. . .