IN4301 Advanced Algorithm - Homework Assignment 3

Lu Dai (4506677) L.Dai-1@student.tudelft.nl

1.
1) Brute-force algorithm is to check all possiblities. We use $\sum_i$ denotes the i^{th} symbol in set $\sum$, t denotes the representative, s_i represents i^{th} sequence is S

```
Boolean Brute-force Search(S, ∑){
  for each ∑_i in ∑ do t[1] = ∑_i
    for each ∑_i in ∑ do t[2] = ∑_i
      ...
        for each ∑_i in ∑ do t[m − 1] = ∑_i
          for each ∑_i in ∑ do t[m] = ∑_i
            for each s_i in S:
              if (dis = distance(t, s_i) > d ) break
              if ( dis ≤ d ) return TRUE   // all distance ≤ d
  return FALSE
}
```

2) Use $|\sum|$ to denote the number of symbol set $\sum$. The distance computation for all the n sequencies requires nm (calculated position by position). The run time for the Brute force algorithm is $O(\ |\sum|^m nm)$, which can be simplified as $O(\ (|\sum|^m)$.

2.
For the m columns, check the same column of the n sequences each time, the greedy decision is to select the symbol which has the highest frequency to the current position. If for some positions, all the symbols from different sequences are the same, then the decision reaches the optimal.

3.
If there is a representative t for set S, then for each s_i in S, it holds that $dist(s_i, t) \leq d$. The total distance is $n * dist(s_i, t) \leq nd$ which indicates nd is the maximum of positions where not all S sequences have the same symbol, as shown in the table below. Red symbols in each sequence stand for symbols that are different from the representative.
Intuitively, for $m \leq nd$, the maximun positions where not all sequences have the same number will not exceed nd.
For $m > nd$, if the positions where not all sequences have the same number is larger than nd, then there is at least one sequence which has a distance to the representative is larger than d. This means there exists no representative for set S.

Representative	T_1	$\cdots$	T_d	T_{d+1}	$\cdots$	T_{2d}	$\cdots$	$T_{(n-1)d+1}$	$\cdots$	T_{nd}	T_{nd+1}	$\cdots$	T_m
S_1	S_{11}	$\cdots$	S_{1d}	$S_{1(d+1)}$	$\cdots$	$S_{1(2d)}$	$\cdots$	$S_{1[(n-1)d+1]}$	$\cdots$	$S_{1(nd)}$	$S_{1(nd+1)}$	$\cdots$	S_{1m}
S_2	S_{21}	$\cdots$	S_{2d}	$S_{2(d+1)}$	$\cdots$	$S_{2(2d)}$	$\cdots$	$S_{2[(n-1)d+1]}$	$\cdots$	$S_{2(nd)}$	$S_{2(nd+1)}$	$\cdots$	S_{2m}
.	.												
.	.												
.	.												
S_n	S_{n1}	$\cdots$	S_{nd}	$S_{n(d+1)}$	$\cdots$	$S_{n(2d)}$	$\cdots$	$S_{n[(n-1)d+1]}$	$\cdots$	$S_{n(nd)}$	$S_{n(nd+1)}$	$\cdots$	S_{nm}

4.
If there exists representative t for set S, each s_i in S has distance $dist(s_i, t) \leq d$ which means there are at most d different positions between s_i and t. Thus, when we randomly pick a sequence, if we change the d different positions, the sequence becomes the representative. Thus, at most d positions need to be changed.

5.
A recurssive algorithm can be defined as following: the sequences are treated as the string type for describing purpose.

> As there are at most l changes and one position is changed at each level of the tree. Thus, search tree's depth is l.
> Changes on $0, 1, \ldots, l$ (up-to l) positions are possible. To find all the situations, the algorithm follows sequential orders of the m positions $0, 1, 2, \ldots, m$ where changes at position 0 means nothing changes.
> For the first level of the tree, there are m+1 branches include m possible positions and one situation where nothing changed. The branches of different nodes in the next level of tree is decided on the position choosed at father nodes. For example, if we choose to modify symbol at postion m, then we could choose to modify from the remain positions from 0 to m-1.
> For each position, we have $|\sum|$ possible symbols to select where n is the maximum number of different symbols in the same column.

l: allowed changes t: candidate representative randomly selected from S
$|\sum|$: the number of symbols in $\sum$ $\sum_i$: the i^{th} symbol in $\sum$

```
String Representative(m, l, t, S, d){
// base case
  if(l<=0){
    for(each s_i in S){if(dist(s_i, t) > d) return NULL}     //not all distance satisfy ≤ d
    return t                                                //all satisfy, return representative t
  }
```

```
  // recursive and for-loop
  for (p = 0 to m){                                        // m+1 possibilities (keep or change on position 1 to m)
    if(p == 0)  a = representation(p-1, p-1, t, S, d)     //no further changes
    for (update = ∑_1 to ∑_|∑| ){                          //select symbol to update candidate
      t[p] = update
      if(p ≥ l) a = representation(p-1, l-1, t, S, d)
      else b = representative(p-1, p-1, t, S, d)
    }
  }
  return a
}
```

6.

As the algorithm enumates all the possible situations/combinations of positions need to be changed. And for each position, we try every possible symbols which may be selected. Thus, the algorithm tries every possibility for changing the candidate representative in l changes (theoretically similar to Brute-force algorithms). As it is indicated, at most l changes will lead to a correct result. Thus, it means our algorithms will finally find a representative after checking with every possibility. Above all, those can prove the correctness for the algorithm.

7.

$$T(m,l) = \begin{cases} = nm & l \leq 0 \text{ (calculating distance position by position)} \\ = (m-l)|\sum|T(m-1,l-1) + l|\sum|T(p-1,p-1) \leq m|\sum|T(m-1,l-1) & l > 0, \text{ p: position choosed to change} \end{cases}$$

The run time is $O(m^l|\sum|^l * nm)$, thus $O^*(m^l|\sum|^l)$. As shown, the algorithm satisfies $O(p(n) * f(l))$ with the exponential part bounded by fixed parameter l. Thus, the algorithm is fixed-parameter tractable.