COMP6714 Assignment

Student Number: z5148637

21 November 2019

1 Q1

1.1

Listing 1: Get Pivot

```
getPivot(a, start, end)
  while end > start + 1:
    pos = start
      for segStart = start to end, step 5:
        segEnd = segStart + 5
        if segEnd > end:
          segEnd = end
        for i = segStart to segEnd - 1:
          for j = i + 1 to segEnd:
            if a[i] > a[j]:
              temp = a[i]
              a[i] = a[j]
              a[j] = temp
        mid = (segEnd + segStart) / 2
        tmp = a[pos]
        a[pos] = a[mid]
        a[mid] = tmp
        pos++
    end = pos
  return a[start]
```

Listing 2: Partition

```
Partition(a, start, end, pivot)
  i = start
  while i <= end:
    if a[i] < pivot: a[start] = a[i]:
      i++
      start = start + 1
    else if a[i] = pivot:
      i++
    else:
      temp = a[end]
      a[end] = a[i]
      a[i] = temp
      end = end - 1

  for i = start to end:
    a[i] = pivot
  pivotStart = start
  pivotLen = end - start + 1
  return pivotStart, pivotLen
```

Listing 3: Intersection

```
answer = []
Intersect(A, B)
  if (A is empty or B is empty):
    return []
  else:
    pivot = getPivot(A, 1, len(A))
    aPStart, aPLen = Partition(A, 0, len(A), pivot)
    bPStart, bPLen = Partition(B, 0, len(B), pivot)
    for i = 1 to min(aPLen, bPLen):
      answer.append(pivot)
    if aPStart > aStart and bPStart > bstart:
      tempA = A.slice(1, aPStart - 1)
      tempB = B.slice(1, bPStart - 1)
      p1 = Intersect(tempA, tempB)
      answer.append(p1)
    if aPStart + aPLen < aEnd and  bPStart + bLen:
      tempA = A.slice(aPStart + aPLen, aEnd)
      tempB = B.slice(bPStart + bPLen, bEnd)
      p2 = Intersect(tempA, tempB)
      answer.append(p2)
```

1.2

In Intersect function, when use slice method to get tempA and tempB, we can divide aPStart + aPLen and bPStart + bPLen into k-1 sections and then Intersect their tempA and tempB separately.

2 Q2

2.1

- According to the logarithmic merge function, if there have two sub-indexes of g_i, then merge them to form a single new sub-index of generation g_{i+1}. So, $g_{i+1} = 2g_i$.
  Assume the smallest sub-index is g_0. And the largest sub-index contains 2^n smallest sub-index g_0 where $n = \lfloor log_2 t \rfloor$. So all sub-indexes are in range [g_0, $\mathrm{g}_{\lfloor log_2 t \rfloor}$], and, $\lfloor log_2 t \rfloor + 1 = \lceil log_2 t \rceil$. So, it will result in at most $\lceil log_2 t \rceil$ sub-indexes;

2.2

- The whole index size is tM.

- let h = $\lfloor log_2 t \rfloor$. So after t rounds. There is only one generation g_h in disk.
  The progress is:
  - one time merge two generation h-1
  - two times merge two generation h-2
  ...
  - 2^{h-1} times merge two generation 0

  So, the total cost is: $\sum_{i=0}^{h-1} = 2^i \cdot (2 * 2^{h-i-1} + 2^{h-i}) \cdot M = h * 2^{h+1} * M$

  Because: $h = \lfloor log_2 t \rfloor$. So, $h * 2^{h+1} * M = \lfloor log_2 t \rfloor * tM$.

  As a result, the I/O cost of the logarithmic merge is $O(t \cdot M \cdot log_2 t)$.

3 Q3

3.1

- $Precision = \frac{6}{20} = 0.3$

3.2

- $Recall = \frac{6}{8} = 0.75$ So, $F_1 = \frac{2PR}{P+R} = \frac{3}{7} = 0.43$

3.3

- $8 * 0.25 = 2$, so the uninterpolated precision could be $1, \frac{2}{3}, \frac{2}{4}, \frac{2}{5}, \frac{1}{3}, \frac{2}{7}, \frac{1}{4}$

3.4

- Because the highest precision that larger than 33% is $\frac{4}{11} = 0.364$, hence the interpolated precision at 33% recall is $\frac{4}{11} = 0.364$

3.5

- $MAP = (1 + 1 + \frac{3}{9} + \frac{4}{11} + \frac{5}{15} + \frac{6}{20})\ /\ 8 = 0.4163$

3.6

- $MAP_{largest} = (1 + 1 + \frac{3}{9} + \frac{4}{11} + \frac{5}{15} + \frac{6}{20} + \frac{7}{21} + \frac{8}{22})\ /\ 8 = 0.5034$

3.7

- $MAP_{smallest} = (1 + 1 + \frac{3}{9} + \frac{4}{11} + \frac{5}{15} + \frac{6}{20} + \frac{7}{9999} + \frac{8}{10000})\ /\ 8 = 0.4165$

3.8

- (6) - (5) = 0.5034 - 0.4163 = 0.0871

4 Q4

4.1

- $P(Q|d1) = \frac{2}{10} \times \frac{3}{10} \times \frac{1}{10} \times \frac{2}{10} \times \frac{2}{10} \times 0 = 0$
- $P(Q|d2) = \frac{7}{10} \times \frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} \times 0 \times 0 = 0$
- d1 and d2 ranked same

4.2

- $p(w_1|d_1) = 0.8 \times \frac{2}{10} + (1 - 0.8) \times 0.8 = 0.32$

  $p(w_2|d_1) = 0.8 \times \frac{3}{10} + (1 - 0.8) \times 0.1 = 0.26$

  $p(w_3|d_1) = 0.8 \times \frac{1}{10} + (1 - 0.8) \times 0.025 = 0.085$

  $p(w_4|d_1) = 0.8 \times \frac{2}{10} + (1 - 0.8) \times 0.025 = 0.165$

  $p(w_5|d_1) = 0.8 \times \frac{2}{10} + (1 - 0.8) \times 0.025 = 0.165$

  $p(w_6|d_1) = 0.8 \times \frac{0}{10} + (1 - 0.8) \times 0.025 = 0.005$

  so, $p(Q|d_1) = 0.32 \times 0.26 \times 0.085 \times 0.165 \times 0.165 \times 0.005 = 9.6 \times 10^{-7}$

- $p(w_1|d_1) = 0.8 \times \frac{7}{10} + (1 - 0.8) \times 0.8 = 0.72$

  $p(w_2|d_1) = 0.8 \times \frac{1}{10} + (1 - 0.8) \times 0.1 = 0.1$

  $p(w_3|d_1) = 0.8 \times \frac{1}{10} + (1 - 0.8) \times 0.025 = 0.085$

  $p(w_4|d_1) = 0.8 \times \frac{1}{10} + (1 - 0.8) \times 0.025 = 0.085$

  $p(w_5|d_1) = 0.8 \times \frac{0}{10} + (1 - 0.8) \times 0.025 = 0.005$

  $p(w_6|d_1) = 0.8 \times \frac{0}{10} + (1 - 0.8) \times 0.025 = 0.005$

  so, $p(Q|d_2) = 0.72 \times 0.1 \times 0.085 \times 0.085 \times 0.005 \times 0.005 = 1.3 \times 10^{-8}$

- d1 would be ranked higher