ITM 613 Algorithm

Mid Term Assignments

Due: 2020/05/06 12:00

Gradings:

- Give full points when correct, 1/n for solving each n subproblems. 0 for totally wrong or none, -1 for each errors.
- There may be partial points for proofs if the direction is correct.
- Copying other's work will be get zero points in the assigment.

1. [Soundex] (5 pts)

Soundex is a phonetic algorithm for indexing names by sound, as pronounced in English. It is used to encode similar words into a same representation. It was first developed in 1910s, even before computers were in use.

The Soundex code for a name consists of a letter followed by three numerical digits. The algorithm is as follows:

1. Save the first letter of the name and drop all other occurrences of `[a, e, i, o, u, y, h, w]`.
2. Replace consonants with digits as follows (after the first letter):
   - `b, f, p, v` → 1
   - `c, g, j, k, q, s, x, z` → 2
   - `d, t` → 3
   - `l` → 4
   - `m, n` → 5
   - `r` → 6
3. If two or more letters with the same number are adjacent in the original name (before step 1), only leave the first letter; also two letters with the same number separated by 'h' or 'w' are coded as a single number, whereas such letters separated by a vowel are coded twice. This rule also applies to the first letter.
4. If you have too few letters in your word that you can't assign three numbers, append with zeros(`0`) until there are three numbers. If you have four or more numbers, retain only the first three.

For example, applying "`algorithm`" with the algorithm, the result for each steps are as follows:

1. `algrtm`
2. `a42635`
3. `a42635`
4. `a426`

Hence we have `a426` for the word "`algorithm`".

Similarly, we can encode "`janghwan`" to `j525`.

(a) Apply the name "`alkhwarizmi`", a 8c Persian mathematician, with the algorithm. Show results for each steps as above.

(b) Apply your own name with the algorithm, showing results for each steps above.

2. [Asymptotic notations] (9 pts)

State whether the following statements are true or false, and explain your answer by proving or disproving it. (3 pts each)

(a) $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

(b) $6n^3 = \Theta(n^2)$

(c) $an^2 + bn + c = \Theta(n^2)$

3. [Search] (30 pts)

A *searching problem* looks up for an item in a sequence:

- **Input**: Sequence of numbers $A = \langle a_1 . a_2, \ldots, a_n \rangle$ of length n, and a value v
- **Output**: An index i such that $v = A[i]$, or NIL if it does not exist in A.

(a) Write pseudocode for `LINEAR-SEARCH(A, v)`, which iterates through A and look for v. *(5 pts)*

(b) Prove that your algorithm is correct by using a loop invariant argument. State the loop invariant and show that it fulfills the three necessary properties. *(10 pts)*

(c) Write pseudocode for linear search using the divide-and-conquer approach. (5 pts)

[binary search]

Here is a new observation: If sequence A is already sorted, we can check the midpoint value of sequence A with v, and ignore half of the sequence. This *binary search algorithm* may be repeated until v is found, or the sequence becomes emtpy. It halves the size of the remaining sequence each time of division.

(d) Write pseudocode for `BINARY-SEARCH(A, v)`. Argue that the worst-case running time of binary search is $\Theta(\lg n)$. The procedure may be either iterative or recursive. *(5 pts)*

(e) Show that the solution to the binary-search recurrence $T(n) = T(n/2) + \Theta(1)$ is $T(n) = \Theta(\lg n)$. Explain your answer. *(5 pts)*

4. [Maximum Subarray problem] (10 pts)

We have studied solving the maximum subarray problem using the divide-and-conquer approach, which improved the running time from $\Theta(n^2)$ of brute-force to $\Theta(n \lg n)$. When this solution was explained in a seminar at Carnegie Mellon University, Jay Kadane designed within a minute an $\Theta(n)$-time algorithm.

Here is the idea of Kadane: start from left end of the array, and then progress toward the right, while maintaining the maximum subarray we have computed so far.

If at iteration j we know a maximum subarray of $A[1..j]$, we know that a maximum subarray of next iteration $A[i..j+1]$ is either the same maximum subarray of $A[1..j]$, or a new subarray $A[i..j+1]$ where $1 \leq i \leq j+1$. We can apply this observation to iteratively find a maximum subarray.

Write pseudocode for `LINEAR-MAXIMUM-SUBARRAY(A)` procedure, which returns the tuple of `(max-left, max-right, max-sum)`. You may write in either bottom-up or top-down fashion. (hint: Use dynamic programming)

5. 2020 National Election (40 pts)

> This problem is intentionally written in Korean to help understanding the terms used in election.

대한민국은 300인의 국회의원을 선출하는 21대 국회의원 총선거를 2020년 4월 15일에 실시했다. 총선을 통해 유권자는

- 각 지역을 대표하는 지역구 의원 253명과
- 각 정당을 대표하는 비례대표 의원 47명

을 선출한다.

투표자는 투표소에서 1) 자기 지역을 대표할 지역구 의원을 투표함과 동시에 2) 지지하는 정당을 투표하는데, 이 때 각 정당이 얻은 득표율을 기준으로 비례대표의 의석 수를 계산한다.

- 각 정당이 얻게 될 전체 의석 수에서 이미 지역구에서 얻은 의석 수는 제외하는데 (연동형),
- 다만 전체 비례대표 의석 수 모두를 이렇게 적용하지는 않고 연동률을 50%만 적용하여 47석 중 30석에만 적용하도록 맞춘다 (준연동형).
- 47석 중 나머지 17석은 정당의 득표율에 비례해서 단순하게 배분한다 (병립형).

결국 개표 후에 각 정당별로 지역구의 총 당선자 수와 비례대표 득표율을 안다면, 병립형 의석 수와 준연동형 의석 수를 각각 구할 수 있다.

중앙선거관리위원회는 연동형 비례대표제 의석배분을 계산하는 동영상("[연동형 비례대표제] 제21대 국선 비례대표 의석배분 어떻게 달라지나"; https://youtu.be/G4jYpy_meHs)을 게재하여 변경된 공직선거법 제 189조(http://www.law.go.kr/%EB%B2%95%EB%A0%B9/%EA%B3%B5%EC%A7%81%EC%84%A0%EA%B1%B0%EB%B2%95)의 내용을 설명하고 있다.

또한 참여연대에서는 국회 의석 수 계산기 사이트 (http://watch.peoplepower21.org/election/) 를 제공한다.

이를 참고하여, 정당별로 비례대표로 당선된 의석 수를 계산하는 함수를 구현하고자 한다.

```
COMPUTE-PROPORTIONAL-RATE-SEATS-COUNT(a, b)
```

- 인자 a 는 개표 결과 각 정당 별 지역구 의석(first-past-the-post) 수가 저장되어 있는 배열이다. 총합은 253이다. 예: $[100, 80, 40, 30, 3]$
- 인자 b 는 개표 결과 정당 별 비례대표 득표율이 저장되어 있는 배열이다. 예: $[0.40, 0.30, 0.10, 0.20, 0.0]$

a와 b는 같은 크기의 배열이고, 동일한 위치에는 동일한 당에 관한 정보가 들어 있다.

※ 본 문제의 모든 의사코드에서 각 배열은 인덱스가 1부터 시작한다.

병립형 의석 수를 구하는 `COMPUTE-MIXED-MEMBER-MAJORITARIAN-SEATS-COUNT` 함수와 준연동형 의석 수를 구하는 `COMPUTE-MIXED-MEMBER-PROPORTIONAL-SEATS-COUNT`를 구현하고자 한다.

비례대표 의석 수를 구하는 전체 함수 `COMPUTE-PROPORTIONAL-RATE-SEATS-COUNT(a, b)`:

```
COMPUTE-PROPORTIONAL-RATE-SEATS-COUNT(a, b)
1  // 전체 비례대표 의석 수 계산
2  n = b.length
3  let total-seats-count[1..n] be a new array
4  p-seats-count = COMPUTE-MIXED-MEMBER-PROPORTIONAL-SEATS-COUNT(a, b)
5  j-seats-count = COMPUTE-MIXED-MEMBER-MAJORITARIAN-SEATS-COUNT(a, b)
6  for i = 1 to n
7      total-seats-count[i] = p-seats-count[i] + j-seats-count[i]
8  return total-seats-count
```

다음의 상수는 전역변수로 주어진다:

$$
\begin{aligned}
&\mathrm{T} = \textsc{Total-Seats-Count} = 300 \\
&\mathrm{D} = \textsc{District-Seats-Count} = 253 \\
&\mathrm{P} = \textsc{Mixed-Member-Proportional-Total-Count} = 30 \\
&\mathrm{J} = \textsc{Mixed-Member-Majoritarian-Total-Count} = 17
\end{aligned}
$$

[병립형 비례대표 의석 수 구하기]

병립형(Mixed Member Majoritarian) 의석 수를 구하는 `COMPUTE-MIXED-MEMBER-MAJORITARIAN-SEATS-COUNT(a, b)` 의 의사코드는 다음과 같이 작성할 수 있다.

$$
\begin{aligned}
&\textsc{Compute-Mixed-Member-Majoritarian-Seats-Count}(a, b) \\
&\quad \text{// 병립형 배분 의석 수 계산} \\
&\quad \text{let } j1[1..n], j2[1..n], j3[1..n], j4[1..n] \text{ be new arrays} \\
&\quad n = b.length \\
&\quad \text{// Compute based on proportion} \\
&\quad \textbf{for } i = 1 \textbf{ to } n \\
&\qquad j1[i] = b[i] \times J \\
&\qquad j2[i] = \textsc{floor}(j1[i]) \qquad \text{// 정수} \\
&\qquad j3[i] = j1[i] - j2[i] \qquad \text{// 소수점} \\
&\quad \text{// Adjust based on decimal points} \\
&\quad \textit{seats-count-so-far} = 0 \\
&\quad \textbf{for } i = 1 \textbf{ to } n \\
&\qquad j4[i] = j2[i] \\
&\qquad \textit{seats-count-so-far} = \textit{seats-count-so-far} + j2[i] \\
&\quad \textit{seats-to-allocate} = \mathrm{J} \text{ - } \textit{seats-count-so-far} \\
&\quad \textbf{for } i = 1 \textbf{ to } \textit{seats-to-allocate} \\
&\qquad \textit{allocate-index} = \textsc{k-th-Rank-Index}(j3, i) \\
&\qquad j4[\textit{allocate-index}] = j4[\textit{allocate-index}] + 1 \\
&\quad \textbf{return } j4
\end{aligned}
$$

※ 여기서 `FLOOR(x)` 은 `x` 의 소수점을 버리고 정수 부분을 반환하는 함수이다. 예: `FLOOR(3.14) == 3`

위의 예제 데이터 a, b 를 입력해보면 각 단계에서 값은 다음과 같다:

	1당	2당	3당	4당	5당	총합
a	100	80	40	30	3	**253**
b	0.40	0.30	0.10	0.20	0.0	**1.0**
$j1$	6.8	5.1	1.7	3.4	0.0	**17.0**
$j2$	6	5	1	3	0	**15**
$j3$	0.8	0.1	0.7	0.4	0.0	**2.0**
$j4$	7	5	2	3	0	**17**

[연동형 비례대표 의석 수 구하기]

다음의 순서에 따라 연동형 비례대표 의석 수를 구하는 함수를 완성하시오.

(a) 동영상에서 설명하는 의석 할당 정당 여부를 판단하는 함수를 구현하고자 한다. 한 정당의 지역구 의원 수와 득표율을 인자로 받아 의석 할당 정당 여부에 따라 `TRUE`, `FALSE`를 반환하는 함수 `PARTY-ABOVE-ELECTION-THRESHOLD(districts-count, proportional-rate)`의 의사코드(pseudocode)를 구현하시오. `districts-count`는 a의 원소이고 `proprtional-rate`는 b의 원소이다. *(5 pts)*

예를 들어, 위 예제를 대입했을 때 제 5당만 `FALSE`가 나온다.

한편, 연동형(Mixed Member Proportional) 비례대표 의석 수를 구하는 전체 코드는 다음과 같이 주어진다.

```
COMPUTE-MIXED-MEMBER-PROPORTIONAL-SEATS-COUNT(a, b)
 1  n = a.length
 2  let p-counts-rounded[1..n] be a new array
 3  // 최초 연동 배분 의석 수 계산
 4  p-counts1 = COMPUTE-MIXED-MEMBER-PROPORTIONAL-SEATS-COUNT1(a, b)
 5  // 소수 첫째 자리에서 반올림
 6  for i = 0 to n
 7      p-counts-rounded[i] = ROUND0(p-counts1[i])
 8  // 정족 수 검사
 9  p-seats-so-far = 0
10  for i = 1 to n
11      p-seats-so-far = p-seats-so-far + p-counts-rounded[i]
12  if p-seats-so-far < P
13      // 정족 수 미만
14      p-counts = ADJUST-PROPORTIONAL-SEATS-COUNT-BELOW-TOTAL(a, b, p-counts-rounded)
15  elseif p-seats-so-far > P
16      // 정족 수 초과
17      p-counts = ADJUST-PROPORTIONAL-SEATS-COUNT-ABOVE-TOTAL(a, b, p-counts-rounded)
18  else
19      p-counts = p-counts-rounded
20  return p-counts
```

※ 여기에서 `ROUND0` 함수는 주어진 숫자를 소수점 첫째 자리에서 반올림하는 함수이다. 예: `ROUND0(3.42) == 3`

(b) 최초 연동 배분 의석 수를 구하는 `COMPUTE-MIXED-MEMBER-PROPORTIONAL-SEATS-COUNT1(a, b)` 함수의 의사코드를 구현하시오. 반환하는 값은 인자 a, b 와 같은 크기를 갖는 배열이며, 각 원소의 최소값은 0이어야 한다. *(10 pts)*

또 위 예제 데이터를 입력했을 때 결과표를 완성하시오.

	1	**2**	**3**	**4**	**5**	**합계**
a	100	80	40	30	3	***253***
b	0.40	0.30	0.10	0.20	0.0	***1.0***
`p_counts1`						

다음과 같은 도움 함수들이 주어진다.

주어진 배열에서 큰 순서로 k번째 순위의 원소를 반환하는 함수 `K-TH-RANK(a, k)`가 있다.

```
// 1번째로 큰 원소를 구하시오
>>> K-TH-RANK([50, 10, 30], 1)
50

// 3번째로 큰 원소를 구하시오
>>> K-TH-RANK([50, 10, 30], 3)
10
```

반대로 배열에서 k번째 큰 순서의 원소의 인덱스를 반환하는 함수 `K-TH-RANK-INDEX(a, k)`가 있다.

```
// 1번째로 큰 원소의 인덱스를 구하시오
>>> K-TH-RANK-INDEX([30, 50, 10], 1)
2

// 2번째로 큰 원소의 인덱스를 구하시오
>>> K-TH-RANK-INDEX([30, 50, 10], 2)
1

// 3번째로 큰 원소의 인덱스를 구하시오
>>> K-TH-RANK-INDEX([30, 50, 10], 3)
3
```

위 도움 함수를 이용하여, 위에서 계산한 최초 연동 배분 의석 수가 정족 수(30석) 미만일 때 이를 조정하여 잔여 의석을 배분하는 함수 `ADJUST-PROPORTIONAL-SEATS-COUNT-BELOW-TOTAL(a, b, p_counts)` 의 의사코드는 다음과 같이 작성할 수 있다:

$\textsc{Adjust-Proportional-Seats-Count-Below-Total}(a, b, p\text{-}counts)$

$n = b.length$
let $p1[1..n], p[1..n]$ be new arrays
// 잔여 배분 의석 수
$p\text{-}seats\text{-}so\text{-}far = 0$
for $i = 1$ **to** n
    $p\text{-}seats\text{-}so\text{-}far = p\text{-}seats\text{-}so\text{-}far + p\text{-}counts[i]$
$seats\text{-}to\text{-}allocate = P - p\text{-}seats\text{-}so\text{-}far$
// 잔여 배분 의석 수 비율
for $i = 1$ **to** n
    $p1[i] = seats\text{-}to\text{-}allocate \times b[i]$
// 정수의 의석 먼저 배정
for $i = 1$ **to** n
    $p[i] = p\text{-}counts[i]$
// 소수점 이하 수가 큰 순으로 각 의석할당정당에 1석 씩 배분
for $i = 1$ **to** $seats\text{-}to\text{-}allocate$
    $allocate\text{-}index = \textsc{K-th-Rank-Index}(p1, i + 1)$
    $p[allocate\text{-}index] = p[allocate\text{-}index] + 1$
return p

※ 단, 의석 배분 과정에서 수가 같아서 해당 정당 사이의 추첨을 하는 경우는 없다고 가정한다. 만약 계산 시에 필요하다면, 배열에서 순서가 앞서는 정당이 가져가기로 한다.

예제 데이터를 입력했을 때 결과는 다음과 같다.

	1	2	3	4	5	*합계*
b	0.40	0.30	0.10	0.20	0.0	***1.0***
`p_counts`	9	5	0	15	0	***29***
`p`	10	5	0	15	0	***30***

(c) 이를 참고하여, 최초 연동 배분 의석 수가 정족 수를 초과했을 때, 이를 조정하는 함수 `ADJUST-PROPORTIONAL-SEATS-COUNT-ABOVE-TOTAL(a, b, p_counts)`의 의사코드를 작성하시오. *(10 pts)* 또 연동 배분 의석 수가 (`p_counts` 인자) 다음과 같을 때 결과는 얼마인지 구하시오:

- `[13, 12, 8, 10, 0]` (총 43석)

다음은 21대 총선에서 후보를 낸 총 41개 정당 중 지역구 당선자가 있거나 비례 득표율이 3% 이상인 정당의 지역구 당선자 수와 비례득표율이다.

정당명 (이름순)	국민의 당	더불어민주 당	더불어시민 당	미래통합 당	미래한국 당	열린민주 당	정의 당	무소 속	기타	계
지역구 당선자 (명)	0	163	0	84	0	0	1	5	0	**253**
비례득표율 (%)	6.79	0	33.35	0	33.84	5.42	9.67	0	10.93	**100.0**

편의상 정당명을 이름 순으로 하여 각각을 a, b 라고 한다. 위 데이터 중 지역구 당선자가 있거나 비례 득표율이 3% 이상인 데이터만 취하면 다음과 같다.

- a: $[0, 163, 0, 84, 0, 0, 1, 5]$
- b: $[0.0679, 0.0, 0.3335, 0.0, 0.3384, 0.0542, 0.0967, 0.0]$

(무소속은 별도 정당이 아니기 때문에 비례대표를 가질 수 없다.)

(d) 위에서 구현한 의사코드에 대입하여, 아래의 표를 완성하여 각 정당별 비례 의석 수를 완성하시오. *(5 pts)*

party	1	2	3	4	5	6	7	8	sum
`a`	0	163	0	84	0	0	1	5	**253**
`b`	0.0679	0	0.3335	0	0.3384	0.0542	0.0967	0	**0.8907**
`compute_mixed_member_majoritarian_seats_count`									
`j1`									
`j2`									
`j3`									
`j4`									
`compute_mixed_member_proportional_seats_counts`									
`p_counts1`									
`p_counts_rounded`									
`p_counts`									
total									
`p_seats_count`									
`j_seats_count`									
`proportional_rate_seats_count`									

소수 정당의 국회 진출을 도와준다는 취지의 준연동형 비례대표제는 본 취지와 달리 거대 양당이 각자의 위성정당을 출범시키는 계기가 되었다. 더불어민주당의 위성정당이 더불어시민당, 미래통합당의 위성정당을 미래한국당으로 보기로 한다.

(e) 만약에 미래한국당과 더불어시민당이 나오지 않았고, 두 당에 투표한 득표율이 각각 미래통합당과 더불어민주당에 들어갔다면 결과가 어찌 되었을지 계산하시오. *(5 pts)*

아래 데이터는 (d) 의 데이터에서 2번째와 3번째 원소를 합치고 4번째와 5번째 원소를 합친 배열들이다.

- $a2$: $[0, 163, 84, 0, 1, 5]$
- $b2$: $[0.0679, 0.3335, 0.3384, 0.0542, 0.0967, 0.0]$

다음의 표를 완성하고, 다음의 질문에도 답하시오.

party	**1**	**2**	**3**	**4**	**5**	**6**	**sum**
`a`	0	163	84	0	1	5	**253**
`b`	0.0679	0.3335	0.3384	0.0542	0.0967	0	**0.8907**
`compute_mixed_member_majoritarian_seats_count`							
`j1`							
`j2`							
`j3`							
`j4`							
`compute_mixed_member_proportional_seats_counts`							
`p_counts1`							
`p_counts_rounded`							
`p_counts`							
total							
`p_seats_count`							
`j_seats_count`							
`proportional_rate_seats_count`							

위 (d)의 결과에서 2번째 3번째 원소를 합친 값과 이 결과의 2번째 원소의 크기를 비교하고, (d)에서 4번째 5번째를 합친 결과와 이 결과의 3번째 원소의 크기를 비교하시오.

(f) 만약에 병립형이 없고, 오로지 100% 연동형으로만 했을 경우 (e)의 입력을 넣어서 표를 완성하고, 결과가 어떻게 바뀌었을지 비교하시오. *(5 pts)*

100% 연동형이 되면 다음 사항들이 변경된다.

- 연동형에 배분하는 의석이 기존 30석에서 47석으로 늘어난다. 병립형 비례대표는 적용하지 않는다.
- 연동형 배분 의석 수를 계산할 때 사용한 50% 반영률은 사라지고, 100% 반영한다.

party	1	2	3	4	5	6	sum
a	0	163	84	0	1	5	**253**
b	0.0679	0.3335	0.3384	0.0542	0.0967	0	**0.8907**
compute_mixed_member_majoritarian_seats_count							
j1							
j2							
j3							
j4							
compute_mixed_member_proportional_seats_counts							
p_counts1							
p_counts_rounded							
p_counts							
total							
p_seats_count							
j_seats_count							
proportional_rate_seats_count							