|  |  |  |  |
| --- | --- | --- | --- |
| 서약서: 나는 본 과제를 수행함에 있어서 허가되지 않은 도움을 주거나 받지 않았음을 서약합니다. | | | |
| 분반:42 | 학번:2016312761 | 이름:여혁수 | (서명)여혁수 |
| 강의 시간 첨삭 자료로 활용함에 동의합니다. | | 동의함 | 동의하지 않음  o |

**분석 리포트 [과제명: 숫자에서 영어를! ]**

1. 분석(문제 분석, 자료 구조 및 알고리즘 분석) (숫자에서 영어를!)

문제 분석) 숫자로 된 문자열이 주어지는데, 순차적으로 숫자를 확인하면서 마주칠 수 있는 상황은 크게 어떤 숫자가 그 다음 숫자와 합쳐 한 알파벳을 이룰 수 있는 경우와 이룰 수 없어 두 숫자가 따로 알파벳을 이루는 경우로 나뉜다. 예외적으로 ‘0’은 자기 자신이 한 알파벳도 될 수 없으므로 무조건 그 전의 문자와 합쳐서 하나의 알파벳을 이루어야 한다. 전역 변수를 설정할 수 없고 in-place하게 문제를 해결해야 한다. 사용할 함수에 어떤 정보를 담은 인자도 전달할 수 없어서 재귀 함수를 이용하는 것도 불가능한 것 같다.

자료 구조) in-place하게 알고리즘을 짜서 추가적인 정보를 담는 변수만 몇 개 선언했음.

알고리즘 분석) 문자열의 처음부터 어느 지점까지를 변환했을 때 가능한 영단어의 수를 가진 값을 계속 저장하면서 반복문을 문자열의 길이만큼 도는 알고리즘입니다. 우선, 문자열의 맨 앞 숫자가 0이라면 그 0은 알파벳으로 바꿀 수 없으므로 예외로 처리하였습니다. 이 조건을 반복문 안에 넣으면 쓸데없이 조건을 확인하는 횟수가 늘어나기 때문에 반복문 밖으로 빼냅니다. 맨 앞 숫자는 아예 반복문에서 제외하려고 입력받은 문자열의 길이가 1이면 단어는 1개 나오게 되니까 바로 1을 리턴하도록 했습니다. 이제 반복문에서 두 번째 숫자부터 확인합니다. 일단 현재 숫자가 0이 아니라면 이 숫자 자체로 어떤 알파벳을 이룰 수 있으므로 이 숫자를 빼고 이때까지 탐색했던 부분 문자열이 이룰 수 있는 영단어의 수가 전체 가능한 영단어의 수와 같다고 할 수 있습니다. 추가적으로 그 전 숫자와 이 숫자가 합쳐서 한 알파벳을 이룰 수 있다면 이 두 숫자를 빼고 이때까지 탐색했던 부분 문자열이 이룰 수 있는 영단어의 수가 전체 가능한 영단어의 수에 더해진다고 할 수 있습니다. 앞서 말한 두 조건을 점화식으로 나타냈을 때, 현재까지 탐색한 문자열의 길이가 n이라고 하면, NumOfCases[n] = NumOfCases[n-1] 이것이 첫 번째 조건만 만족했을 때이고, 두 번째 조건도 만족한다면, NumOfCases[n] = NumOfCases[n-1] + NumOfCases[n-2]가 되는 것입니다. 두 조건 모두 확인했다면 다음 문자로 진행하는데 이 때, 남은 문자열의 길이가 1 줄어든 것이므로 우리가 계산한 NumOfCases[n], NumOfCases[n-1]은 다음 iteration에서 각각 NumOfCases[n-1], NumOfCases[n-2]가 됩니다. 그래서 이 두 값만 저장해서 다음 숫자로 넘어간다면 계속해서 점화식을 적용시킬 수 있고, 문자열 끝까지 진행하면서 전체 가능한 영단어의 수를 계산할 수 있게 됩니다. 모든 숫자는 자신이 반복문 안에서 선택된 인덱스의 값일 때와 그 전 인덱스의 값이 될 때 총 두 번 확인하게 되므로 시간 복잡도는 O(n)이 됩니다.

2. 생각해보기 (숫자에서 영어를\_v2!)

숫자에서 영어를\_v2’는 인풋 스트링의 문자에 ‘**?**’가 추가됩니다. ‘**?**’는 0~9까지의 어떤 숫자로든 바뀔 수 있습니다.

알고리즘 분석) 제가 설명했던 알고리즘과 동일하게 반복문에서 ?가 인덱스로 선택되면 0~9까지 모두 올 수 있으므로 ?가 0일 때부터 9일 때 까지 경우의 수를 모두 세야합니다. 우선 ‘?’ 자체로 알파벳을 이룰려면 ‘?’는 1~9중 하나이어야 합니다. 즉 ‘?’는 하나의 알파벳으로 정해진 것이 아니고 9개의 모양이 있는 것이므로 인덱스0~n까지 문자열에서 나올 수 있는 단어의 수를 나타내는 점화식은 기존값에서 9를 곱한 것인 NumOfCases[n] = NumOfCases[n-1] \* 9 가 됩니다. ‘?’값이 변하는 것이 나올 수 있는 단어의 경우들 전체에 영향을 주기 때문에 곱하는 것입니다. 또한 그 전 문자가 1이나 2나 ‘?’라면 같이 이루는 숫자가 알파벳이 될 수 있습니다. 이 조건까지 만족하는 경우 점화식은 NumOfCases[n] = NumOfCases[n-1]\*9 + NumOfCases[n-2]\*(두 숫자가 이룰 수 있는 알파벳 수)가 됩니다. 시간복잡도는 기존의 점화식에 constant만 변경되었으므로 여전히 O(n)이 될 것입니다.

3. 비고 (교수나 조교에 하고 싶은 말)

DP 문제인 것 같긴 한데 저는 DP 문제를 항상 추가적인 배열을 만들어 풀어왔기 때문에 추가적인 배열을 선언할 수 없어서 못 푸는 문제인 줄 알았는데 일단 점화식을 생각해서 식을 세워보니까 추가적으로 두 개의 변수만 선언해서 저장하고 전달하면 전체 경우의 수를 계산할 수 있겠다는 결론에 이르렀습니다. 이게 될까? 하면서 테스트 했는데 전부 정답이었습니다.. 저는 제 알고리즘이 고려하지 못한 예외 케이스가 있을 것 같습니다. 그리고, ‘?’가 세 개 이상 연속될 때도 점화식이 n-1과 n-2만으로 구성될 수 있는지 살짝 헷갈리는데, 일반 숫자와 다를 것 없이 독립적인 문제인 것 같습니다.

Ex) 1232???834?20

123까지 단어 3개(ABC, LB, AW) , 1232까지 단어 3개. (ABCB, LCB, AWB)

1232? 까지: NumOfCases[n-1]->3 \* 9(1~9까지) + NumOfCases[n-2]->3 \* 10(20,21…29) = 57;

1232?? 까지: NumOfCases[n-1]->57 \* 9(1~9) + NumOfCases[n-2]->3 \* 17(10,11…26) = 564;

1232??? 까지: NumOfCases[n-1]->564 \* 9(1~9) + NumOfCases[n-2]->57 \* 17(10,11…26)(?)