Homework 4
SNU 4190.210 Fall 2012

Kwangkeun Yi

due: 10/12(Fri) 24:00

이번 숙제의 목적은

- 데이터의 속 구현을 프로그램 해 보기.
- 재귀함수가 항상 종료함을 확인해 보기.

Exercise 1 "가마 코드"

문장을 표현하는 코드를 고안한다고 합시다. 문장은 단어들의 유한한 나열이라고 합시다. 사용할 단어들은 "가마", "꼭", "꽃", "타고" 만 있다고 합시다. 이 단어들을 가지고 다양한 문장들이 만들어 질 수 있습니다. 예를 들어 "가마가마꼭가마꽃가마타고가마" 혹은 "꽃가마꼭타고가마가마꼭가마" 등.

위의 문장을 0과 1로 표현하는 방법을 고안합시다. 단어마다 0과 1로 구성된 코드를 지정하고, 문장들은 그 코드들을 일렬로 쓰면 되겠습니다. 두 가지 방식이 있습니다:

1. 크기고정 코드(fixed-length encoding): 단어가 4개이므로 각각 두 비트로 표현하면 됩니다: 00(가마), 01(꼭), 꽃(10), 타고(11). 그래서 "가마가마꼭가마꽃가마타고가마"는 0000010010001100 가 됩니다. 위의 코드를 복구(decode)하는 것은 쉽습니다. 두 비트씩 잘라서 해당 단어로 바꿔주면 됩니다. 전체가 16비트가 필요합니다.

2. 크기변동 코드(variable-length encoding): 단어의 사용빈도를 알고 자주 사용되는 단어일 수록 짧은 코드를 사용하면, 문장을 표현하는 코드는 더 짧아질 수 있습니다.

   단어들의 사용빈도로 보면, "가마"가 제일 자주 사용되고, 다음으로는 "타고", "꼭", "꽃" 순서라는 것을 알았다고 합시다. 그러면 다음과 같이 단

어마다 크기가 다른 코드를 지정하게 됩니다: 0(가마), 11(타고), 100(꼭), 101(꽃). 그러면, "가마가마꼭가마꽃가마타고가마"는 0010001010110 이 됩니다. 16비트가 아니고 13비트로 표현됩니다.

위의 코드를 복구(decode)하는 것도 쉽습니다. 코드를 읽어가면 반복해서 다시 살필 필요 없이 단어들이 결정됩니다: 첫번째 0을 읽으면 "가마"입니다. 다른것이 될 수 없어요. 두번째 0을 읽으면 또 "가마"이고요. 다음 1을 읽으면 그 다음 00까지 읽게됩니다. 그때까지 해당 코드가 없으니까요. 100까지 읽으면 그 때 "꼭"이 됩니다. 다른게 불가능합니다. 이런식으로 주욱 읽으면서 해당하는 유일한 문장이 쉽게 복구됩니다.

크기변동 코드를 써도 복구가 쉬운 이유는 뭘까요? 각 단어 코드의 앞부분이 또 다른 단어의 코드가 되지 않도록 했기 때문입니다. 단어 코드들을 보세요: 0, 11, 100, 101. 각 코드의 어떤 앞부분을 봐도 다른 코드가 되지 않습니다. 이런 성질을 "앞겹침 없는(prefix-free)"성질이라고 부릅니다. 이러한 코딩방식이 JPEG, MPEG, ZIP등에서 사용하는 방법입니다.

이번 숙제에서는 "앞겹침 없는"성질을 가지는 크기변동 코드를 만드는 프로그램 `vlencode`를 작성하는 것입니다. 단어와 빈도수 짝들의 리스트를 받아서 단어와 코드 짝들의 리스트를 돌려줍니다. 단어는 스트링, 빈도수는 정수, 코드는 0/1의 리스트입니다.

이것이 1951년, MIT의 학생이었던 David Huffman이 "Information Theory"강의의 프로젝트 리포트로 제출한 코딩방법입니다. 강의를 맡은 교수는 Robert Fano였고, 당시 최적의 코딩방법이 알려지지 않은 상태에서 동료인 Claude Shannon과 머리를 싸매고 고민하고 있었다고 합니다. Fano교수는 이 문제를 강의 프로젝트 문제로 낸 것이고, 수강생이었던 Huffman이 해결했던겁니다. 이 결과물로 Huffman은 교수를 능가했고, 강의에서 당연히 A를 받았고, 유명해졌습니다. 여러분도 Huffman정도는 될 수 있습니다. 아래 힌트면 충분할 겁니다. Google로 답을 찾지 말고 도전해 보세요.

- 위의 `vlencode`는 두갈래 나무구조(binary tree)를 이용해서 만들 수 있습니다. 나무구조의 잎새에 단어들이 매달려 있고, 각 나무구조의 가지마다 $\cdots$ *[중략]* $\cdots$. 이러면 "앞겹침 없는"성질의 코드가 단어마다 배정되겠지요. 잎새에만 단어들이 매달려 있기 때문입니다.

나무구조를 위해 아래의 함수들을 정의해서 사용하세요:

$$\begin{aligned}
&\texttt{leaf} : \mathit{string} \times \mathit{value} \to \mathit{tree}\\
&\texttt{node} : \mathit{tree} \times \mathit{value} \times \mathit{tree} \to \mathit{tree}\\
&\texttt{isleaf?} : \mathit{tree} \to \mathit{bool}\\
&\texttt{leftsub} : \mathit{tree} \to \mathit{tree}\\
&\texttt{rightsub} : \mathit{tree} \to \mathit{tree}\\
&\texttt{leafval} : \mathit{tree} \to \mathit{value}\\
&\texttt{leafstr} : \mathit{tree} \to \mathit{string}\\
&\texttt{rootval} : \mathit{tree} \to \mathit{value}
\end{aligned}$$

- 나무구조를 단어의 빈도 순서에 따라서 잘 만들어야 가장 최적의 앞겹침 없는 코드를 단어에 배정할 수 있습니다. 빈도수가 작은 단어부터 시작해서 나무구조를 만드는데 $\cdots$ *[중략]* $\cdots$ 방식으로 조합해가면 되겠지요.
- 재귀적으로 정의한 함수들은 항상 끝나는 지를 수업시간에 익힌 방식으로 확인하고 함수소스옆에 주석으로 포함해서 제출하세요.

□

Exercise 2 "SKI 용액 반응기"

"SKI" 용액이라는 것을 상상합시다. SKI용액을 만들어 상온에 놔두면 반응이 일어나기 시작하고 용액의 성분이 변하게 됩니다. 그 용액의 초기 상태에 따라서, 일정시간이 지나면 반응이 멈추고 굳어 버리는 경우도 있고, 어떤 경우는 반응이 끝 없이 일어나면서 용액이 무한히 변해가기도 합니다.

SKI용액 E를 만드는 방법은 다음의 5가지 입니다.

$$\begin{array}{rcll}
E & \to & \texttt{S} \mid \texttt{K} \mid \texttt{I} & \\
& \mid & x & \text{variables}\\
& \mid & \texttt{(}E\ E\texttt{)} &
\end{array}$$

SKI용액의 예들은

$$\texttt{K},\ \texttt{(I}\ x\texttt{)},\ \texttt{(S ((K}\ x\texttt{)}\ y\texttt{))},\ \texttt{(((S K) K)}\ x\texttt{)}$$

등이 되겠지요.

SKI용액의 반응 규칙은 아래와 같습니다. 용액중에 아래 반응 규칙의 왼편에 맞는 것이 있으면 그 것이 오른편의 것으로 바뀌게 됩니다.

$$\begin{aligned}
\texttt{(I}\ E\texttt{)} &\to E\\
\texttt{((K}\ E\texttt{)}\ E'\texttt{)} &\to E\\
\texttt{(((S}\ E\texttt{)}\ E'\texttt{)}\ E''\texttt{)} &\to \texttt{((}E\ E''\texttt{) (}E'\ E''\texttt{))}
\end{aligned}$$

예를 들어, SKI 용액 (((S K) I) x)는 다음과 같이 반응합니다:

$$(((\texttt{S K})\ \texttt{I})\ x) \to ((\texttt{K}\ x)\ (\texttt{I}\ x)) \to x$$

SKI용액을 받아서 반응과정을 거쳐 최종적으로 굳어버린 용액의 모습을 출력하는 함수 react

$$\texttt{react} : \textit{용액} \to \textit{void}$$

를 작성하세요. 참고로, 어느 한 순간에 위의 반응이 일어날 수 있는 부분이 여러군데 있을 수 있습니다. 예를 들어 ((K x) (I y))은 두 개의 반응과정이 가능합니다:

$$((\texttt{K}\ x)\ (\texttt{I}\ x)) \to x$$

혹은

$$((\texttt{K}\ x)\ (\texttt{I}\ x)) \to ((\texttt{K}\ x)\ x) \to x.$$

여러분의 react는 복수의 반응이 가능한 경우 어느 하나를 선택하면 됩니다.

SKI용액을 만들고 사용하는 아래의 함수들은 구현해서 사용하세요:

- S : *용액*
- K : *용액*
- I : *용액*
- v : *string* → *용액* (* 변수이름으로 용액만들기 *)
- a : *용액* × *용액* → *용액* (* 용액 두 개 괄호에 넣기 *)
- isS? : *용액* → *bool*
- isK? : *용액* → *bool*
- isI? : *용액* → *bool*
- isv? : *용액* → *bool*
- isa? : *용액* → *bool*
- var : *용액* → *string* (* 변수용액의 변수이름 *)
- al : *용액* → *용액* (* 괄호안 두 개 용액중에 왼쪽 용액 *)
- ar : *용액* → *용액* (* 괄호안 두 개 용액중에 오른쪽 용액 *)
- pprint : *용액* → *void* (* 이쁘게 프린트하기 *)

위에서 pprint는 TA가 지정하는 방식으로 프린트되야 합니다. □