Compact balanced parentheses. Recall that in lecture we introduced the balanced-parentheses representation of a binary trie. A common query when using this representation is \texttt{match}(i), which returns the index of the close parenthesis matching the open parenthesis at index \textit{i}. In this problem we will develop a compact data structure to answer these queries (though a succinct data structure is known).

1. Let \textit{S} be a string of balanced parenthesis. Divide \textit{S} into blocks of size \textit{B}. We will call an open parenthesis \textit{far} if its matching parenthesis is in a different block. A far parenthesis is a \textit{pioneer} if its matching parenthesis is in a different block than the matching parenthesis of the previous far parenthesis in \textit{S}. Prove that there are \textit{O}(|\textit{S}|/\textit{B}) pioneer parenthesis.

2. Develop and analyze a static data structure that, given a string of balanced parenthesis \textit{S}, answers \texttt{match} queries in \textit{O}(1) time and uses \textit{O}(|\textit{S}|) bits of space.