```cpp
vector<map<char,int>> edges;    // edges[i]  : the labeled edges from node i
vi link;                        // link[i]   : the parent of i
vi length;                      // length[i] : the length of the longest string in the ith class
int last;                       // the index of the equivalence class of the whole string
unordered set<int> terminals;

void build(string s) {
    // add the initial node
    edges.push back(map<char,int>());
    link.push back(-1);
    length.push back(0);
    last = 0;
    FOR0(i, sz(s)){
        // construct r
        edges.push back(map<char,int>());
        length.push back(i+1);
        link.push back(0);
        int r = edges.size() - 1;
        // add edges to r and find p with link to q
        int p = last;
        while(p >= 0 && edges[p].find(s[i]) == edges[p].end()) {
            edges[p][s[i]] = r;
            p = link[p];
        }
        if(p != -1) {
            int q = edges[p][s[i]];
            if(length[p] + 1 == length[q]) {
          // we do not have to split q, just set the correct suffix link
                link[r] = q;
            }
            else {
                // we have to split, add q'
                edges.push back(edges[q]); // copy edges of q
                length.push back(length[p] + 1);
                link.push back(link[q]); // copy parent of q
                int qq = edges.size()-1;
                // add qq as the new parent of q and r
                link[q] = qq;
                link[r] = qq;
                // move short classes pointing to q to point to q'
                while(p >= 0 && edges[p][s[i]] == q) {
                    edges[p][s[i]] = qq;
                    p = link[p];
                }
            }
        }
        last = r;
    }
    // finding terminals
    int p = last;
    while(p > 0) {
        terminals.insert(p);
        p = link[p];
    }
}

```