Pattern matching with multyple string reduction to kmp

Authored by Nitzan Migdal

206763351

==Description==

This program takes an array of strings and searches for them inside a given string and returns the matches and there placement in the string given to us.

Program DATABASE:

Pm :

1. new\_state = how many states are in the pm. ///in my program not initialized.  
2. zero\_state = a pointer to the first pm\_state\_t.

pm\_state\_t contains:

1. \_transitions- A list of possible transitions from it (also states).  
2. Output:- if the state is an accepting state it has the value of the string that it represents.  
3. id – number of the state.  
4. depth – how long is the string is  
5. fail – the longest Sub String of the current string we are working on inside all the strings added.

pm\_labeled\_edge:

1.label – the value of the char that connects the the last state to the state in the labeled edge.  
2.state – pointer to the state.

Dbllist\_t :

1.(\_size\_t)size – the size of the list.  
2.(db\_node\_t\*)head – the head of the list.  
3.(db\_node\_t\*)tail – the tali of the list.

db\_node\_t:

1.(void\*)data – the data is used in a few different ways. In the transition portion it is used as a (pm\_labeled\_edge) that points to the state connected to the value in transitions of the last state and the label that says what is the connection between them.  
2.(db\_node\_t\*)next – gets the next value.  
3. (db\_node\_t\*)prev – gets the previous value.

Functions:  
  
**FILE**: **Slist**:  
  
 dbllist-init – gets malloced dbllist\_t\* and sets the values of the contents inside it to NULL and initializing the first node and sets it as the head of the list.  
  
 dbllist\_destroy – Gets a dbllist\_t and removes every node inside it and then frees the list it self , if the value dealloc == FREE it also frees the nodes inside because they are malloced.  
  
 dbllist\_append – given (void\*)data and (dbllist\_t\*)List it adds the data to the list.   
 If the size of the list is 0 we will just set the value of the data of the first node that was initialized in the dbllist\_init.  
 If not we will make a new node and set it as the next after the tail. And set the tail to be the new node.  
  
 dbllist\_prepend – the same as append but at the start of the list.  
   
 dbllist\_remove – gets (node\_t)Node (dbllist\_t\*)List and removes the node from the list.  
  
  
**FILE: Pattern\_matching.c**

pm\_state\_init – gets a malloced state\_t and sets the values of it. (mallocs the dbllist\_t in output and in \_transitions) and sets the values of id and depth to 0 until the will be reset to the real values inside of the set\_state.  
  
 pm\_init – gets a malloced pm and initialize it.  
  
 pm\_addstring – gets: pm to set the string to , string to add, length of the string.  
 For every character in the string we will check if the current state we are in have the value of the char in its \_transitions dbllist if it does we will go to the next char and set the cur\_state at the state we found if not we will make a new state and set it at the end of the \_transitions of the cur\_state and then sets the new state as the cur\_state.   
   
 pm\_goto\_get – gets a state and a symbol and searches for the symbol in the transitions of the state.  
  
 pm\_goto\_set – gets a state a symbol and another state and makes a new pm\_labeled\_edge and sets the symbol as the label of the new edge and then sends the value to append to the transitiions of the to\_state.  
  
 pm\_makeFSM – gets a pm and sets all the value of fail in every state inside the pm.   
  
 pm\_fsm\_search – gets a state, string,length of the string and serches for all the substrings inside the string that is in the pm.  
  
 state\_destroy – a recursive function that gets a state and destroies the state and the states that are connected to it inside the transtitions and output.   
  
 pm\_destroy – gets a pm and send the zerostate of the pm to the state\_destroy. Then frees the pm.

==Program Files==  
pattern\_matching.c- the file contains the functions that is writen above.   
pattern\_matching.h- an header file ,contain structs,declerations of functions.  
Slist.c- contains the functions listed above.  
Slist.h - an header file ,contain structs,declerations of functions.

==How to compile?==  
compile: gcc main.c pattern\_matching.c slist.c -o tester  
run: ./tester

==Input:==  
ano need to insert input from the user.

==Output:==

will output the setting of a new state in this way : allocating the state (number of state):  
from cur\_state -> with this label -> to\_state.  
Will output the setting of the fail state to a state that the id is not 0:

fail(cur\_state) = to\_state;