

## Narcissus search engine

The code is produced as a collaboration between Phil Jones and Aharon Amir

```
define Search_Result {
  result;      # the search result itself
  status;     # what status is the Search_Result in? One of One, MinusOne, OneI or MinusOneI
  cycle_counter; # counts the the times the search results go through the shades.
               # If it is a new or unclicked search result, the status_counter of the result is
always set to 1.
  click_counter; # counts the number of times the Search_Result was clicked (or shadow
clicked)
  was_clicked;  # is set if the last searcher clicked on this search result
  was_printed; # is set if this result was shown to a previous searcher
}

define Term_List {

  art; # this is the term which, in this example, will not be harvested by the narcissus.
       # more and other terms can be placed here.
}

# this function calculates "shade status" of Search Results in the database.
function calculate_shade(Search_Results) {

  # Note that Search_Results are objects as defined above.
  # They are annotated with extra fields for tracking status and counting clicks etc.

  create a new, empty, printable_list;

  for each Search_Result in Search_Results {

    if Search_Result.status is One {
      if Search_Result.was_clicked {
        # changes the status to -1, which will then affect
        # where such a Search_Result will appear on the
        # search result page, i.e. at the lower end,
        # hence giving the Search Result a darker shade.
        change Search_Result.status to MinusOne;
        change Search_Result.click_counter to 0;
        # we want to count the number of clicks in the next state
      } else {
        # do nothing ... we leave status as One
      }
      include Search_Result at the top of the printable_list;
    }
    elsif Search_Result.status is MinusOne {
      r = the square root of Search_Result.cycle_counter;
      if Search_Result.was_clicked {
        increment Search_Result.click_counter
      }

      if Search_Result.click_counter is greater than r {
        #changes the status of a Search_Result to 1i.
      }
    }
  }
}
```

```

        # These Search Results will be temporarily removed
        # from search shown results, given as even darker shade.
        change Search_Result.status to One!;
        change Search_Result.click_counter to 0;
    }
    include Search_Result at the bottom of the printable_list;

} elseif Search_Result.status is One! {
    # we have the Search_Result, which means that
    # it already matches the search-term
    # Because we're in state One! we don't actually print it
    # but we assume that it has been "shadow clicked"
    # and so we advance towards state MinusOne!
    increment Search_Result.click_counter;

    r = the square root of Search_Result.cycle_counter;

    if the Search_Result.click_counter is greater than r {
        change Search_Result.status to One!;
        # the status of the search result changes to One! if the search result
        # was clicked more times than the number representing the square root
        # of amount of cycles the search result had through the system
        change Search_Result.click_counter to 0;
        # the click counter is set to 0 so that it could re-count.
    }

} elseif Search_Result.status is MinusOne! {
    # Again, because we're in the shadow-world,
    # where the Search_Result is not visible, we deduce the click
    # automatically if the search term relates to the search result
    increment Search_Result.click_counter;
    # but we now also increment the cycle counter
    increment Search_Result.cycle_counter;

    r = the square root of Search_Result.cycle_counter;

    if the Search_Result.click_counter is greater than r {
        change Search_Result.status to One;
        # the status of the search result changes to One if the search result
        # was clicked more times than the number representing the square root
        # of amount of cycles the search result had through the system
        change Search_Result.click_counter to 0;
        # the click counter is set to 0 so that it could re-count.
    }
}
}
#End of for loop
return the printable_list;
# brings the list of eligible search results to printed on the page
}
# End of calculate_shade() function

function search (Term_List) {
    get search_terms from user;

```

```

if the search_terms are utterly new one and the engine does not have any
files with related results {
    print "click here to add a relevant site to be harvested by Narcissus";
    get url from user;
    run_spider(url);
    # if the user puts a query that has NO relevant search results in the system
    # they are prompted to suggest a possible relevant site
}

```

```

if the search_terms contain the word in Term_List {
    # art is an example of a search term that is being excluded
    # "art" is defined in Term_List
    print "no results possible, I have no idea what art is.";
    quit;
}

```

```

Search_Results = get the documents that match the search_terms;

```

```

# calls the calculate_shade function on the Search_Results and uses it to fill
# the printable_list of results
get the printable_list from calculate_shade(Search_Results);

```

```

if everything in printable_list has status One1 or MinusOne1 {
    print "All results to this search have seen too much light.";
    # informing user that there are results, however, they are shaded
}

```

```

}
# End of search() function

```

```

function run_spider (URL,Term_List) {
    # this is how the search engine harvests websites for relevant results.
    # it assumes people submit URLs they think might be relevant to the narcissus engine.

```

```

# operators of the narcissus engine can replace the string "art", or add other terms
# they might wish to be excluded on their narcissus operations on Term_List.

```

```

if URL is not on list of sites not to be revisited;
# make sure URL is not one the spider has already discarded for containing the string "art".
and if

```

```

    URL is not already on database;
    # make sure the URL is not on the database.

```

```

then
    fill the requested URL list;
    select the first URL;
    repeat;
    download the selected URL;
    save the page; metadata
    if this is an HTML containing page
    then
    for each page
    extract metadata;
    if any metadata includes string containing Term_List
        discard entry and place URL in sites not to be re-visited;
    # the above operation checks for the term we want to exclude from the entire search
    # engine, and discards sites that contain the string, for example, "art", in its metadata.

```

```

else
extract tags;
if any tag includes string containing Term_List
    discard entry and place URL in sites not to be re-visited;
# the above operation checks for the term we want to exclude from the entire search
# engine, and discards sites that contain the string, for example, "art", in its tags.
else
extract text;
if any text includes string containing Term_List
    discard entry and place URL in sites not to be re-visited;
# the above operation checks for the term we want to exclude from the entire search
# engine, and discards sites that contain the string, for example, "art", in its text.
else if URL was not discarded and put on list of sites not to be revisited;
then
    extract links;
    follow links and download pages;
    if any link points to a site containing string Term_List
        discard entry and place URL in sites not to be re-visited;
# the above operation checks for the term we want to exclude from the entire search
# engine, and discards sites that contain the string, for example, "art", in its any sites
# LINKED to the original site.
    else
if site was NOT put on list of sites not to be revisited and was NOT discarded;
save site URL;
save site tags;
save site metadata;
then
select the next URL;
until no more URLs;
} # end of spider

```

##### DISTRIBUTED SEARCH MODE #####

### this is a code for a possible use of Narcissus in multiple installations on various servers  
### and controlled by different people/organisations. This mode is using the same search  
### function we used earlier, with the addition of communicating with other servers  
### running their own Narcissus installations

```

function distributed_search (Term_List) {
    get search_terms from user;

    call other computers running narcissus connected to this one;
    get Term_List from other narcissus installations;

    if the search_terms are utterly new one and the engine does not have any
    files with related results {
        print "click here to add a relevant site to be harvested by Narcissus";
        get url from user;
        run_spider(url);
        # if the user puts a query that has NO relevant search results in the system
        # they are prompted to suggest a possible relevant site
    }

    if the search_terms contain the word Term_List {
        # art is an example of a search term that is being excluded;
    }
}

```

```
print "no results possible, I have no idea what art is.";
quit;
}

call Search_Results on other computers running narcissus connected to this one;
# calls other computers that run a connected installation of narcissus

get Search_Results;
# get the relevant search results from all connected computers
# this operation will affect the shade status of the search results on all
# connected computers

Search_Results = get the documents that match the search_terms;

# get the calculate_shade function on the Search_Results and uses it to fill
# the printable_list of results
get the printable_list from calculate_shade(Search_Results);

if everything in printable_list has status One1 or MinusOne1 {
    print "All results to this search have seen too much light.";
    # informing user that there are results, however, they are shaded
}

}
# End of distributed_search() function
```