Course Code : MCS-021
Course Title : Data and File Structures
Assignment Number : BCA(3)/021/Assignment/16-17
Maximum Marks : 100 Weightage : 25%
Last Dates for Submission : 15th October, 2016 (For July 2016 Session)
15th April, 2017 (For January 2017 Session)

Course Code : MCS-021 Course Title : Data and File

1. Write an algorithm for the implementation of a Tree. ***(20 Marks)***

Ans:

```
struct btree
{
int data;
struct btree *left;
struct btree *right;
};
void add(struct btree **q,int n)
{
struct btree *t,*temp;
t=malloc(sizeof(struct btree));
t->data=n;
t->left=NULL;
t->right=NULL;
if(*q==NULL)
*q=t;
else
{
temp=*q;
while(temp->left!=NULL && temp->right!=NULL)
{
if(temp->data<=n)
temp=temp->right;
else
temp=temp->left;
}
if(temp->data<=n)
temp->right=t;
else
temp->left=t;
}
}
inorder(struct btree *s)
{
if(s!=NULL)
```

```
{
inorder(s->left);
printf(" >%d",s->data);
inorder(s->right);
}
return;
}
preorder(struct btree *s)
{
if(s!=NULL)
{
printf(" >%d",s->data);
inorder(s->left);
inorder(s->right);
}
return;
}
postorder(struct btree *s)
{
if(s!=NULL)
{
inorder(s->left);
inorder(s->right);
printf(" >%d",s->data);
}
return;
}
main()
{
struct btree *p;
p=NULL;
clrscr();
add (&p,10);
add(&p,5);
add(&p,60);
add(&p,70);
printf("\n\nin-order=\n");
inorder(p);
printf("\n\npre-order=\n");
preorder(p);
printf("\n\npost-order=\n");
postorder(p);
```

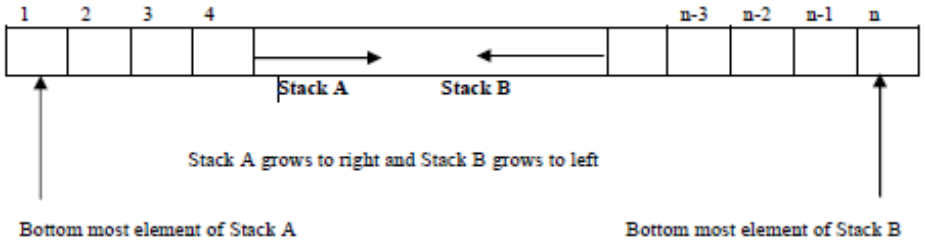
```
getch();
}
```

2. Implement multiple stacks in a single dimensional array. Write algorithms for various stack operations for them. ***(20 Marks)***

Ans:

Suppose A and B are two stacks. We can define an array stack A with n_1 elements and an array stack B with n_2 elements. Overflow may occur when either stacks A contains more than n_1 elements or stack B contains more than n_2 elements.

Suppose, instead of that, we define a single array stack with $n = n_1 + n_2$ elements for stack A and B together. Let the stack A "grow" to the right, and stack B "grow" to the left. In this case, overflow will occur only when A and B together have more than $n = n_1 + n_2$ elements. It does not matter how many elements individually are there in each stack.




```
#include<stdio.h>
#include <conio.h>
#define max 10                          // size of the Stack
int top1,top2,stack[max];

void push(int x)
{
  int x,ch;
  if(top1==top2-1)                      // Condition for checking If Stack is Full
  {
   printf("stack overflow\n");
   return;
  }
  printf("\npress 1 to push the element in 1st stack or press 2 for stack 2:");
  scanf("%d",&ch);
  if(ch==1)
     stack[++top1]=x;                   //increment the top and inserting element in stack1
  else
      stack[--top2]=x;                  //decrements the top of stack 2
  printf("%d successfully pushed\n",x);
```

```
  return;
}
void pop()
{
  int y,ch;
   printf("\npress 1 to pop the element from 1st stack or press 2 for from stack 2");
  scanf("%d",&ch);
  if(ch==1)
  {
     if(top1==-1)                          // Condition for checking If Stack 1 is Empty
     {
    printf("stack underflow\n");
    return;
     }
     y=stack[top1];
     stack[top1]=0;      //insert 0 at place of removing element and decrement the top
          top1--;
}
  else
  {
     if(top2==max)                  // Condition for checking If Stack 2 is Empty
      {
    printf("stack underflow\n");
    return;
      }
      y=stack[top2];
      stack[top2]=0;
          top2++;
  }

  printf("deleted \n");
  return;
}
void display (void)
{
  int i;
  if(top1==-1)
  {
  printf("stack 1 is empty\n");
  }
  else
  {
```

```
    printf("elements of Stack 1 are :\n");
    for(i=0;i<=top1;i++)
    {
   printf("%d\n",a[i]);
     }
 }
if(top2==max)
{
 printf("stack 2 is empty\n");
 return;
}
printf("Elements of stack 2 are \n");
for(i=max-1;i>=top2;i--)
{
 printf("%d\n",a[i]);
}
return ;
}
```

3. Write a note of not more than 5 pages summarizing the latest research in the area of "Searching Techniques". Refer to various journals and other online resources. Indicate them in your assignment. ***(20 Marks)***

Ans:

The logic behind database searches

Searching in a database is matching your search words with existing words in the database. You enter one or several search words and the database will tell you whether these words can be found in the database. Most search tools automatically imply AND between each word entered. If you enter two words you will only get the results that include both words together in the same document.

There are search services that automatically search for related terms and synonyms. That kind of searches will yield a greater amount of hits.

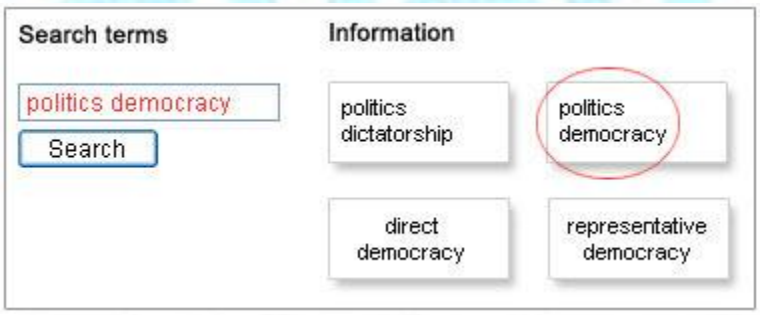


Illustration of matching using two terms

What happens when you execute the search in the example is that you ask the database to return documents where the words *politics* AND *democracy* are included - no other

documents. Had the search been executed using only the word *politics*, the upper left document had been returned by the database.

In many databases you can also use OR between the search words. Applying this to the example below, we would obtain documents that either include bribes OR corruption. The operator OR is used when you want to increase the number of hits, and it is very useful if you want to search for all kinds of synonyms.

Illustration of search with the OR operator

Include different word endings with truncation

Besides choosing which words to search for, you also have to think about in which form you write them. If you, for instance, want to find publications about democratisation studies, you might also be interested in publications wherein such words as *democratization* with American English spelling, or any ending of the word, like *democracies* (plural form), or *democratic* (as in democratic parties) occur. Instead of doing several separate searches, you can use a so-called truncation. Enter the stem of the word and a truncation character, like *democra**, and you will retrieve all publications wherein the truncated word stem occurs, irrespective of which ending it has.

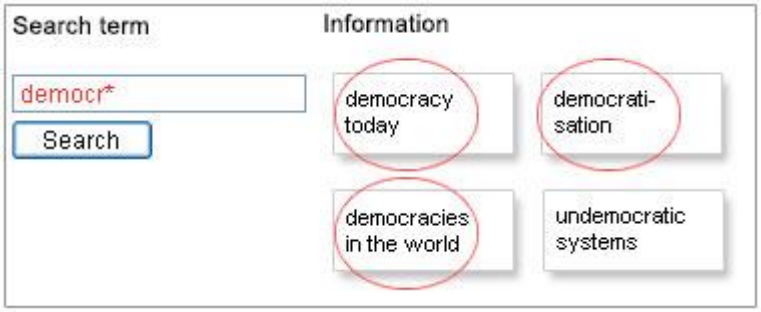


Illustration of search with truncation.

In many databases the truncation character can also be used at the beginning or in the middle of a word. When it is used in the middle of the word it is usually called a *wildcard*.

The most common truncation character is the asterisk (*), but other characters might occur, such as the question mark (?). Sometimes, different characters are used depending on whether it is used inside or at the ending of the word. Read the search help text of the database you are currently searching in.

It might be good to know that Google uses a form of automatic truncation. A search might retrieve hits on other variants of the word used in your search. However, in the library's article databases, for instance, you usually have to truncate words in order to retrieve all variants.

Free text search and metadata search

Which search words you choose is crucial, but you should also be aware of what information you are searching in. Are you searching in the whole document text or only among titles, authors, or other metadata that describes the documents?

Searching in the whole text of documents is usually called free text search, or full text search. A typical example of this is Google. When you are searching in the whole text, you can use specified words that occur in the text.

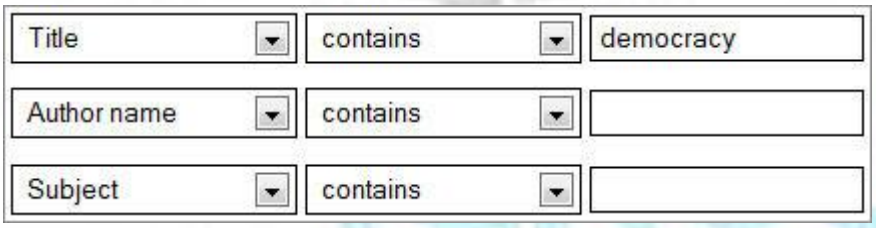
However, if you are searching, using only the metadata that describes the document, you cannot be as specific.

Phrase search

By putting two or more words withing quotation marks, you will only retrieve documents wherein the those words are in exactly that order. This technique works in most databases and search engines. Example: "third world".

Field search

In most databases you will find advanced search options where you can specify in which field you want to limit your search to. In the example to the right, you will only retrieve documents where the words economic development occurs in titles.



Delimiters

In some cases you need to limit your search result, usually when the search has yielded lots of hits with a great variety of relevance. This is often the case when the database is of a general and multidisciplinary character and covers many different subject areas and publication types.

In contrast to metadata searching, this is something you do after the initial search. In the interfaces of today's databases you will be usually be offered a set of delimiters that you can use to limit your search.

Publication type

For example, searching only for journals or dissertations.

Publication date

For example, limiting your search to find only material within the latest three years.

Subject area

In some cases you can choose among a number of subject areas. The example below is from the database Science Direct.

Managing search lists

Pay attention to how search results are presented. Relevancy means that the database automatically weight which documents are relevant based on your search qriteria. In

scientific databases the default setting is usually by date, but this option is one that can be easily changed to relevancy instead.

4. What is a B-Tree? Write a short note on its applications. ***(20 Marks)***

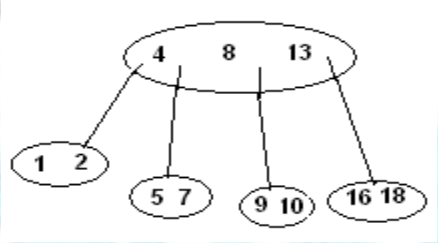
Ans:

A **B-tree** is a self-balancing **tree** data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time. The **B-tree** is a generalization of a binary search **tree** in that a node can have more than two children.

A B-tree is a balanced multi – way tree. A node of the b-tree contains more than one key. It is also called balanced sort tree. Remember B-tree is not a binary tree.

Attributes of B-tree having order N:-

1. Each node has maximum of N children(branches) and a minimum m/2
2. Each node has maximum of N-1 keys.
3. Keys are arranged in an order.
4. When key is inserted into full node, it will be broken.



Application

A database is a collection of data organized in a fashion that facilitates updation, retrieval and management of the data. Searching an unindexed database containing n keys will have a worst case running time of O (n). If the same data is indexed with a b-tree, then the same search operation will run in O(log n) time. Indexing large amounts of data can significantly improve search performance.

- Manipulate hierarchical data.
- Make information easy to search (see tree traversal).
- Manipulate sorted lists of data.
- As a workflow for compositing digital images for visual effects.
- Router algorithms
-