Saraswathi B

1BM19CS032

Skip list

#include<bits/stdc++.h>

#define Max\_level 3

const float P = 0.5;

using namespace std;

struct skipnode

{

int value;

skipnode \*\*for\_arr;

skipnode(int level, int &value)

{

for\_arr = new skipnode \* [level + 1];

memset(for\_arr, 0, sizeof(skipnode\*) \* (level + 1));

this->value = value;

}

~skipnode()

{

delete [] for\_arr;

}

};

struct skiplist

{

skipnode \*header;

int value,level;

skiplist()

{

header = new skipnode(Max\_level,value);

level = 0;

}

~skiplist()

{

delete header;

}

void displayNodes();

bool searchElement(int &);

void deleteElement(int &);

void insertElement(int &);

};

float frand()

{

return (float) rand() / RAND\_MAX;

}

int random\_level()

{

static bool first = true;

if (first)

{

srand((unsigned)time(NULL));

first = false;

}

int lvl = (int)(log(frand()) / log(1.-P));

return lvl < Max\_level ? lvl : Max\_level;

}

void skiplist::insertElement(int &val)

{

skipnode \*curr = header;

skipnode \*update[Max\_level+1];

memset(update,0,sizeof(skipnode\*)\*(Max\_level+1));

for(int i = level;i>=0;i--)

{

while(curr->for\_arr[i] != NULL && curr->for\_arr[i]->value < val)

{

curr = curr->for\_arr[i];

}

update[i]=curr;

}

curr = curr->for\_arr[0];

if (curr == NULL || curr->value != val)

{

int lvl = random\_level();

if (lvl > level)

{

for (int i = level + 1;i <= lvl;i++)

{

update[i] = header;

}

level = lvl;

}

curr = new skipnode(lvl, val);

for (int i = 0;i <= lvl;i++)

{

curr->for\_arr[i] = update[i]->for\_arr[i];

update[i]->for\_arr[i] = curr;

}

}

}

void skiplist::deleteElement(int &val)

{

skipnode \*x = header;

skipnode \*update[Max\_level + 1];

memset (update, 0, sizeof(skipnode\*)\*(Max\_level + 1));

for (int i = level;i >= 0;i--)

{

while (x->for\_arr[i] != NULL && x->for\_arr[i]->value < val)

{

x = x->for\_arr[i];

}

update[i] = x;

}

x = x->for\_arr[0];

if (x->value == val)

{

for (int i = 0;i <= level;i++)

{

if (update[i]->for\_arr[i] != x)

break;

update[i]->for\_arr[i] = x->for\_arr[i];

}

delete x;

while (level > 0 && header->for\_arr[level] == NULL)

{

level--;

}

}

}

bool skiplist::searchElement(int &s\_value)

{

skipnode \*x = header;

for (int i = level;i >= 0;i--)

{

while (x->for\_arr[i] != NULL && x->for\_arr[i]->value < s\_value)

{

x = x->for\_arr[i];

}

}

x = x->for\_arr[0];

return x != NULL && x->value == s\_value;

}

void skiplist::displayNodes()

{

for (int i = 0;i <= level; i++)

{

skipnode \*x = header->for\_arr[i];

cout<<"Level "<<i<<": ";

while (x != NULL) {

cout << x->value<<" ";

x = x->for\_arr[i];

}

cout<<endl;

}

}

int main()

{

skiplist ss;

int choice, n;

cout<<"1.Insert Element"<<endl;

cout<<"2.Delete Element"<<endl;

cout<<"3.Search Element"<<endl;

cout<<"4.Display List "<<endl;

cout<<"5.Exit "<<endl;

while (1)

{

cout<<"Enter your choice : ";

cin>>choice;

switch(choice)

{

case 1: cout<<"Enter the element to be inserted: ";

cin>>n;

ss.insertElement(n);

break;

case 2: cout<<"Enter the element to be deleted: ";

cin>>n;

if(!ss.searchElement(n))

{

cout<<"Element not found"<<endl;

break;

}

ss.deleteElement(n);

break;

case 3: cout<<"Enter the element to be searched: ";

cin>>n;

if(ss.searchElement(n))

cout<<"Element "<<n<<" is in the list"<<endl;

else

cout<<"Element not found"<<endl;

break;

case 4: cout<<"Elements in list are: "<<endl;

ss.displayNodes();

break;

case 5: exit(1);

break;

default:

cout<<" Invalid option"<<endl;

}

cout<<endl;

}

return 0;

}