ใบงานที่ 25

เรื่อง ต้นไม้เพื่อการค้นหา

เสนอ

อาจารย์ ปิยพล ยืนยงสถาวร

จัดทำโดย

นาย สารินทร์ อินต๊ะรักษา รหัส 65543206082-1

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา โครงสร้างข้อมูลและขั้นตอนวิธี

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 1 ปีการศึกษา 2566

คำสั่ง/คำชี้แจง

- สร้างโค้ดโปรแกรมตามตัวอย่างในเอกสารประกอบการสอน
- แสดงโค้ดโปรแกรมเป็นส่วนๆ พร้อมอธิบาย
- แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน
- สรุปผลการทดลอง

ลำดับขั้นการทดลอง

```
int Data[MaxData];
int N,key,Times;
bool result;
struct Node //Declare structure of node of Tree
{
int info;
struct Node *lson,*rson;
};
struct Node *Start[MaxData],*Root,*p; // Declare pointer node
Node *Allocate() //Allocate 1 node from storage pool
{
struct Node *temp;
temp=(Node*)malloc(sizeof(Node)); //Allocate node by size declare
return(temp);
}
bool Duplicate(int i,int Data1) //Check Duplication Data
{
int j;
for(j=1;j<=i;j++)
if(Data1==Data[j])
return(true);
return(false);
}
```

- ประกาศอาร์เรย์ชื่อ Data ขนาด MaxData ซึ่งเป็นตัวแปรที่เก็บข้อมูลชนิดจำนวนเต็ม (int)
- ประกาศตัวแปร N, key, และ Times ที่ใช้เก็บค่าจำนวนเต็ม
- ประกาศตัวแปร result ซึ่งเป็นชนิดข้อมูลบูลีน (bool) และใช้เก็บค่าความจริง (true/false)
- ประกาศโครงสร้าง (structure) ที่ชื่อว่า Node ซึ่งมีสมาชิกข้อมูล int ชื่อ info และสมาชิกข้อมูลชนิด (pointer) ชื่อ lson และ rson เพื่อเก็บข้อมูล
- ประกาศตัวแปร Start เป็นอาร์เรย์ของโพยเตอร์ของโครงสร้าง Node, ตัวแปร Root เป็นโพยเตอร์ของโครงสร้าง Node, และตัวแปร p เป็นโพยเตอร์ของโครงสร้าง Node.
- Node *Allocate เป็นฟังก์ชันที่สร้างและคืนโครงสร้าง Node ใหม่จากหน่วยความจำ (memory) โดยใช้การจองหน่วยความจำด้วย malloc
- bool Duplicate ฟังก์ชันที่ใช้สำหรับตรวจสอบว่าค่า Data1 ซ้ำกับค่าในอาร์เรย์ Data หรือไม่ โดยคืนค่า true ถ้าซ้ำ และ false ถ้าไม่ซ้ำ

```
void PrepareRawKey(int N)
{
int i,temp;
srand(time(NULL)); //for difference random number in rand()
for (i=0;i<N;i++)
{
temp=(rand() % 89)+10; //random difference number 10..99
while(Duplicate(i-1,temp)) //Loop if Still Duplicate
temp=(rand() % 89)+10; //random again
Data[i]=temp; //Keep new Number
} //End for
} //End Fn.
void DispKey(int N)
{
int i;
for(i=0;i<N;i++)
printf("(%2d)",i); //Show Subscript i
printf("\n");
for(i=0;i<N;i++)
printf(" %2d ",Data[i]); //Show Data[]
printf("\n");
}
```

- void PrepareRawKey เป็นฟังก์ชันที่ถูกใช้ในการเตรียมข้อมูลในอาร์เรย์ Data โดยรับค่า N เป็นพารามิเตอร์ ฟังก์ชันนี้จะสร้างข้อมูลสุ่ม N ค่าจำนวนเต็มในช่วง 10 ถึง 99 แล้วเก็บค่าเหล่านี้ในอาร์เรย์ Data. การสร้างค่าสุ่มที่ไม่ซ้ำกันใช้ฟังก์ชัน Duplicate เพื่อตรวจสอบความซ้ำกันกับข้อมูลที่มีอยู่แล้วในอาร์เรย์ Data
- void DispKey เป็นฟังก์ชันที่ใช้ในการแสดงค่าที่อยู่ในอาร์เรย์ Data และตัวเลขดัชนีของแต่ละค่า ฟังก์ชันนี้จะแสดงค่าของ N ดัชนีแรกและค่าที่เก็บในอาร์เรย์ Data

```
void CreateBST(int N)
{
int i;
bool Finish;
struct Node *T1,*p;
p=Allocate(); //Set Root Node
p->info=Data[0];
p->lson=NULL;
p->rson=NULL;
Root=p; //Set Root Node Pointer
for (i=1;i<N;i++) //Count by Number of Data
{
T1=Root; //Let T1 point at Root Node
p=Allocate();
p->info=Data[i];
p->lson=NULL;
p->rson=NULL;
Finish=false;
while(!Finish)
{
if(Data[i]<T1->info)
if(T1->lson==NULL)
{
//Add Left Node
T1->lson=p; //Let LSON Last Node point to New Node
Finish=true; //Done
}
else
T1=T1->lson; //Skip to Left Son
else
if(T1->rson==NULL)
{
//Add right Node
T1->rson=p; //Let RSON Last Node point to New Node
Finish=true; //Done
}
else
T1=T1->rson; //Skip to Right Son
} //End while
} //End for
} //End Fn.
```

- Allocate จะจองหน่วยความจำสำหรับโหนดแรกและกำหนดค่าข้อมูลให้กับโหนดนี้เท่ากับ Data[0] และกำหนดค่าโพยเตอร์ลูกซ้ายและลูกขวาให้เป็น NULL แล้วกำหนด Root ให้ชี้ไปที่โหนดแรก
- หลังจากนั้น, ฟังก์ชันจะวนลูปผ่านอาร์เรย์ Data ที่เหลือ (ข้อมูลที่เรียงตามลำดับจาก Data[1] ถึง Data[N-1]) เพื่อเพิ่มโหนดในต้นไม้ตามลำดับข้อมูล
- ในแต่ละรอบของลูป, ฟังก์ชันจะทำการเริ่มการเพิ่มโหนดในต้นไม้ที่ Root และใช้ตัวชี้ที่ชื่อ T1 ในการเดินตามต้นไม้เพื่อค้นหาตำแหน่งที่เหมาะสมในการเพิ่มโหนดใหม่
- ฟังก์ชันจะเริ่มที่ Root และเลื่อกผ่านต้นไม้โดยเปรียบเทียบค่าข้อมูล Data[i] กับค่าข้อมูลของโหนดปัจจุบัน T1->info หาก Data[i] น้อยกว่า T1->info และโหนดซ้าย (T1->lson) ยังไม่มีค่า (คือ NULL) โหนดใหม่จะถูกเพิ่มเป็นซ้ายของ T1 ถ้า Data[i] มากกว่าหรือเท่ากับ T1->info และโหนดขวา (T1->rson) ยังไม่มีค่า (คือ NULL) , โหนดใหม่จะถูกเพิ่มเป็นขวาของ T1
- ขั้นตอนนี้จะทำซ้ำจนกว่าโหนดใหม่จะถูกเพิ่มในต้นไม้ และที่สำคัญคือ โหนดใหม่จะถูกเพิ่มในต้นไม้ให้ถูกตำแหน่งตามลำดับของข้อมูลที่ถูกจัดเก็บในอาร์เรย์ Data

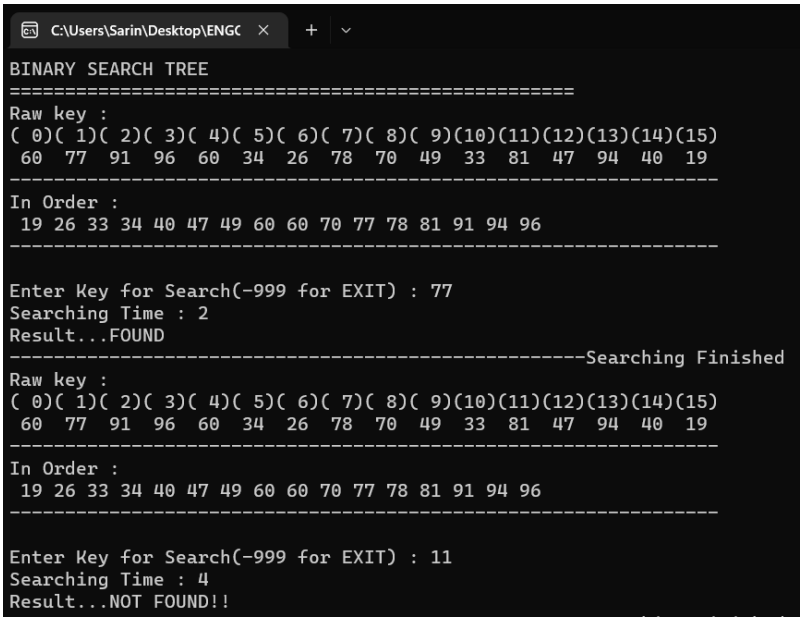
```
void InOrder(struct Node *i)
{
if (i != NULL) //if i NOT NULL
{
InOrder(i->lson); //Call left Son by InOrder
printf(" %2d",i->info); //Display INFO
InOrder(i->rson); //Call Right Son by InOrder
}
}
bool SearchBST(int key)
{
struct Node *T1;
Times=0;
T1=Root;
while(T1!=NULL)
{
Times++; //Count the search time
if(key==T1->info)
return(true); //Found
else
if(key<T1->info)
T1=T1->lson; //Skip T1 to Left
else
T1=T1->rson; //Skip T1 to Right
} //End While
return(false); //NOT Found
} //End Fn.
```

- InOrder เป็นฟังก์ชันที่ใช้ในการแสดงข้อมูลในต้นไม้ Binary Search Tree โดยใช้ลำดับ InOrder ซึ่งหมายถึงการแสดงข้อมูลในลำดับซ้าย - โหนดปัจจุบัน - ขวา และฟังก์ชัน SearchBST ใช้ในการค้นหาค่าในต้นไม้ BST โดยเริ่มที่รากของต้นไม้และค้นหาไปเรื่อยๆ จนกว่าจะพบค่าหรือจบการค้นหา และจะคืนค่า true หากค้นพบ และ false ถ้าไม่พบ

```c
int main()
{
printf("BINARY SEARCH TREE\n");
printf("==================================================\n");
N=16;
PrepareRawKey(N);
CreateBST(N);
while(key!=-999)
{
printf("Raw key :\n");
DispKey(N); //Raw key
printf("------------------------------------------------------------------\n");
printf("In Order :\n");
InOrder(Root);
printf("\n------------------------------------------------------------------\n");
printf("\nEnter Key for Search(-999 for EXIT) : ");
scanf("%d",&key); //Read key from KBD
if(key!=-999)
{
result=SearchBST(key);
printf("Searching Time : %d\n",Times);
printf("Result...");
if(result)
printf("FOUND\n"); //if found
else
{
Beep(600,600);
printf("NOT FOUND!!\n"); //if NOT found
}
printf("--------------------------------------------------Searching Finished\n");
} //End if
} //End While
return(0);
} //End Main
```

- แสดงข้อความ "BINARY SEARCH TREE" บนหน้าจอ
- กำหนดค่า N เป็น 16 นำมาสร้างในต้นไม้ BST
- เรียกใช้ฟังก์ชัน PrepareRawKey เพื่อเตรียมข้อมูลแบบสุ่มในอาร์เรย์
- เรียกใช้ฟังก์ชัน CreateBST เพื่อสร้างต้นไม้ BST จากข้อมูลที่อยู่ในอาร์เรย์ Data
- แสดงเมนูโหนดต้นของโปรแกรมในลูป while(key!=-999):
- แสดงข้อมูลต้นไม้ต้นฉบับที่เริ่มต้นด้วย DispKey(N)
- แสดงข้อมูลในต้นไม้ในลำดับ InOrder ด้วย InOrder(Root)
- สำหรับค่า key ที่ไม่ใช่ -999 ระบบจะทำการค้นหาค่า key ในต้นไม้ BST
- แสดงผลลัพธ์ค้นหา ("FOUND" หากพบ และ "NOT FOUND!!" หากไม่พบ)
- โปรแกรมจะวนลูปเริ่มต้นใหม่ถ้าผู้ใช้ไม่ป้อน -999 เพื่อสิ้นสุดโปรแกรม

ผลลัพธ์การทดลอง



สรุปผลการทดลอง

โปรแกรมนี้ใช้เพื่อสร้างต้นไม้ BST จากข้อมูลที่เตรียมไว้และให้ผู้ใช้ค้นหาค่าในต้นไม้และแสดงผลลัพธ์ของการค้นหาบนหน้าจอ

สื่อ / เอกสารอ้างอิง

ไฟล์ประกอบการสอนของ อาจารย์ ปิยพล ยืนยงสถาวร เรื่อง : ต้นไม้เพื่อการค้นหา