

1. LỜI MỞ ĐẦU

Trong công nghệ thông tin, tìm kiếm là vấn đề luôn luôn được đặt lên hàng đầu vì tầm quan trọng quyết định trong quá trình cập nhật và sử dụng thông tin. Bài toán công nghệ luôn đòi hỏi phải tìm kiếm sao cho tốn ít thời gian nhất, chất lượng tìm kiếm chính xác nhất. Trong các thuật toán tìm kiếm, tìm kiếm trên cây nhị phân tỏ ra hiệu quả bởi dễ dàng trong cấp phát bộ nhớ và độ phức tạp thời gian tối ưu là $O(\log_2 n)$.

Để có thể tìm kiếm trên cây nhị phân tìm kiếm, trước hết chúng ta phải tiến hành xây dựng cây nhị phân tìm kiếm trong bộ nhớ RAM. Cây nhị phân tìm kiếm là một cây nhị phân mà tại mỗi nút của nó đều thỏa mãn điều kiện là giá trị tại nút đó lớn hơn giá trị tất cả các nút con, cháu, chắt... ở bên con trái và nhỏ hơn giá trị tất cả các nút con, cháu, chắt ... ở bên con phải. Như vậy khi tiến hành tìm trên cây chúng ta chỉ cần tiến theo độ cao của cây, mà độ cao cây chính là $\log_2 n$ khi n là số nút. Vì vậy đây là thuật toán có độ phức tạp về thời gian là rất tối ưu.

Trong quá trình giảng dạy, không như mảng hoặc danh sách liên kết, để biểu diễn, cây nhị phân tìm kiếm việc biểu diễn trên màn hình máy tính là quá trình gặp nhiều khó khăn. Vì vậy sinh viên với những kiến thức mới tiếp cận của mình khó mà hình dung được việc lưu trữ cây nhị phân trong bộ nhớ RAM của máy tính là như thế nào.

Trong đề tài này chúng tôi tiến hành thực hiện các giải thuật xây dựng cây nhị phân tìm kiếm, giải thuật tìm kiếm trên cây nhị phân tìm kiếm và giải thuật mô phỏng cây nhị phân tìm kiếm trên màn hình máy tính. Đặc biệt các giải thuật này được chuyển cụ thể thành hàm và tổ chức chương trình trên ngôn ngữ lập trình C^{++} .

2. MỤC TIÊU ĐỀ TÀI

Việc nghiên cứu đề tài này và xây dựng chương trình trên nền ngôn ngữ lập trình C^{++} nhằm hướng tới các mục tiêu sau:

- Nghiên cứu, lập hàm tạo cây nhị phân tìm kiếm.
- Nghiên cứu, lập hàm tìm một khóa X trên cây nhị phân tìm kiếm.
- Nghiên cứu lập hàm vẽ cây nhị phân tìm kiếm ra màn hình.
- Nghiên cứu lập hàm thêm một nút vào hàng đợi.

- Nghiên cứu lập hàm lấy ra một nút từ hàng đợi.

- Tạo dựng tài liệu cơ bản trực quan giúp cho các giáo viên, sinh viên và những người quan tâm nâng cao kiến thức khi tìm hiểu và học các môn học như : Kỹ thuật lập trình C⁺⁺, Cấu trúc dữ liệu và giải thuật...

3.NỘI DUNG CỦA ĐỀ TÀI

I. Xây dựng cây nhị phân tìm kiếm

Để xây dựng cây nhị phân tìm kiếm có n nút, ta thực hiện việc thêm từng lá có thông tin x vào cây, để cho cây vẫn là cây nhị phân tìm kiếm. Như vậy khi thực hiện n lần thêm một lá ta sẽ có cây nhị phân tìm kiếm có n nút.

Vì vậy bài toán ở đây sẽ là thêm một nút có thông tin X vào cây nhị phân tìm kiếm T. Giải thuật được viết bằng hàm trong C⁺⁺ thể hiện như sau:

```
void ThemLa(kpt *&T, int x)
{
    kpt *q = new kpt; q->info = x; q->lptra=NULL; q->rptra=NULL;
    if(T==NULL)
        T=q;
    else
    {
        kpt *p=T;
        while (p!=NULL)
        {
            if(p->info==x)
                break;
            if(x<p->info)
            {
                if(p->lptra==NULL)
                {
                    p->lptra=q; break; }
                else
                    p=p->lptra;
            }
            else
            {
                if(p->rptra==NULL)
                {
                    p->rptra=q; break; }
                else
                    p=p->rptra;
            }
        }
    }
}
```

II. Biểu diễn cây nhị phân tìm kiếm trên màn hình

Cây nhị phân tìm kiếm là một cấu trúc móc nối trong RAM. Vì vậy để biểu diễn nó trên màn hình máy tính sao cho vẫn giữ nguyên tầng mức là không dễ dàng. Ở đây chúng tôi dùng cây trúc duyệt hàng đợi, với những nút có giá trị

rỗng (bằng NULL) thì không vẽ, các nút khác NULL thì vẽ. Như vậy sẽ tạo ra hình dáng cây nhị phân tìm kiếm sinh động trên màn hình.

Giải thuật được viết bằng hàm trong C⁺⁺ thể hiện như sau:

```
void vecay(kpt *T)
{ kpt *node; char string[25];
//-----
int gdriver = DETECT, gmode, errorcode;
int xmax, ymax;

/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();
/* an error occurred */
if (errorcode != grOk)
{
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(0);
}
//-----
if(T != NULL)
{ int tang = 0, spt ,dempt = 1; int dau = 1;
  int deltax =int(getmaxx()/(2)), deltax ;
  int x = deltax, y = 20;
  int huyen,delx,dely;

//Dem so nut de tinh do cao cay
  spt = 0;
  PUSH(T);
  setttextjustify(CENTER_TEXT, CENTER_TEXT);
  while((TI != -1)&&(BO != -1))
  { POP(node); spt++;
    if(node->lptr != NULL)
      PUSH(node->lptr);
    if(node->rptr != NULL)
      PUSH(node->rptr);
  }
  if(spt>1)
    deltax = int((getmaxx()-40)/(int(log(spt)/log(2))));
//-----
  PUSH(T); spt = 1;
  setttextjustify(CENTER_TEXT, CENTER_TEXT);
  setttextstyle(2, 0, 10);
  while((TI != -1)&&(BO != -1))
  { POP(node);
    if(node->lptr != NULL)
      PUSH(node->lptr);
    if(node->rptr != NULL)
      PUSH(node->rptr);
    // rectangle(xchu, 30, xchu + 30, 60);
```

```

// printf("%d; ",node->info);
itoa(node->info, string, 10);
circle(x, y, 15);
outtextxy(x, y, string);
//outtextxy(xchu + 15, 45, string);
//xchu = xchu + 30;
if(tang>0)
{
    huyen = int(sqrt(double(deltax)*deltax +
double(deltay)*deltay));
    if(huyen>0)
    {
        delx = int(deltax * 15/huyen);
        dely = int(deltay * 15/huyen);
    }
    line(x+dau*delx,y-dely,x+dau*(deltax-delx), y -
(deltay-dely) );
}
dempt++;
// Ha tang
if(dempt>spt)
{
    tang++; spt = pow(2,tang); dempt = 1 ;
    deltax =int(getmaxx()/(2*spt)); x = deltax;
    y = y + deltax; dau = 1;}
else
{
    x = x + 2*deltax; dau = - dau; }
}

}
getch(); closegraph();
}

```

III. Giải thuật đếm mức trên cây

Để biểu diễn cây trên một màn hình máy tính, điều không thể thiếu là chúng ta phải biết số nút trên cây hoặc số nút có mức cao nhất trên cây. Vì vậy trong khi vẽ chúng tôi có sử dụng hàm đếm mức trên cây.

Giải thuật xác định mức cao nhất trên cây được viết bằng hàm trong C⁺⁺ thể hiện như sau:

```

//-----
void DemMuc(kpt *p,int sm)
{
    if(p!=NULL)
    {
        sm = sm+1;
        if(sm>muc)
            muc=sm;
        DemMuc(p->lptr, sm);
        DemMuc(p->rptr, sm);
    }
}

```

4. KẾT QUẢ ĐẠT ĐƯỢC CỦA CHƯƠNG TRÌNH

Trong quá trình nghiên cứu, chúng tôi đã xây dựng được một chương trình trên ngôn ngữ lập trình C++ bao gồm các hàm như sau:

- * Tạo một cây nhị phân tìm kiếm trong bộ nhớ RAM.
- * Vẽ một cây nhị phân tìm kiếm ra màn hình
- * Tìm một khóa có giá trị X trên cây nhị phân tìm kiếm.
- * Thêm một phần tử vào hàng đợi.
- * Lấy ra một phần tử từ hàng đợi.

Để các đọc giả có thể cùng nghiên cứu và trao đổi, chúng tôi giới thiệu toàn bộ chương trình đã viết được trên ngôn ngữ lập trình C++ như sau:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<stdlib.h>
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<complex.h>
struct kpt
{
    int info; kpt *lptr, *rptr; };
kpt *Q[50];
int TI = -1;
int BO = -1;
int muc=0,m=0;
//-----
void POP(kpt *&x)
{
    if(BO<0)
        { printf("Hang doi rong!"); getch(); }
    else
        { x = Q[BO];
          if(BO==TI)
              { BO = -1; TI = -1; }
          else
              { BO = BO + 1;
                if(BO==50)
                    BO = 0;
              }
        }
}
//-----
void PUSH(kpt *x)
{
    if(((TI==49)&&(BO==0))||(TI==(BO-1)))
        {printf("Hang doi day !"); getch();}
    else
        { if(TI<0)
```

```

        {TI = 0; BO = 0; }
    else
        TI = TI +1;
    if(TI==50)
        TI = 0;
    Q[TI] = x;
}

}

//-----
void ThemLa(kpt *&T, int x)
{
    kpt *q = new kpt; q->info = x; q->lptr=NULL; q->rpتر=NULL;
    if(T==NULL)
        T=q;
    else
    {
        kpt *p=T;
        while(p!=NULL)
        {
            if(p->info==x)
                break;
            if(x<p->info)
                if(p->lpتر==NULL)
                {
                    p->lpتر=q; break; }
                else
                    p=p->lpتر;
            else
                if(p->rpتر==NULL)
                {
                    p->rpتر=q; break; }
                else
                    p=p->rpتر;
        }
    }
}

//-----
void vecay(kpt *T)
{
    kpt *node; char string[25];
    //-----
    int gdriver = DETECT, gmode, errorcode;
    int xmax, ymax;

    /* initialize graphics and local variables */
    initgraph(&gdriver, &gmode, "");

    /* read result of initialization */
    errorcode = graphresult();
    /* an error occurred */
    if (errorcode != grOk)
    {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(0);
    }
    //-----
    if(T != NULL)
    {
        int tang = 0, spt ,dempt = 1; int dau = 1;
        int deltax =int(getmaxx()/(2)), deltay ;
        int x = deltax, y = 20;
        int huyen,delx,dely;
    }
}

```

```

//Dem so nut de tinh do cao cay
spt = 0;
PUSH(T);
settextjustify(CENTER_TEXT, CENTER_TEXT);
while((TI != -1)&&(BO != -1))
{ POP(node); spt++;
  if(node->lptr != NULL)
    PUSH(node->lptr);
  if(node->rptr != NULL)
    PUSH(node->rptr);
}
if(spt>1)
  deltax = int((getmaxy()-40)/(int(log(spt)/log(2))));
//-----
PUSH(T); spt = 1;
settextjustify(CENTER_TEXT, CENTER_TEXT);
settextstyle(2, 0, 10);
while((TI != -1)&&(BO != -1))
{ POP(node);
  if(node->lptr != NULL)
    PUSH(node->lptr);
  if(node->rptr != NULL)
    PUSH(node->rptr);
  // rectangle(xchu, 30, xchu + 30, 60);
  // printf("%d; ",node->info);
  itoa(node->info, string, 10);
  circle(x, y, 15);
  outtextxy(x, y, string);
  //outtextxy(xchu + 15, 45, string);
  //xchu = xchu + 30;
  if(tang>0)
  { huyen = int(sqrt(double(deltax)*deltax +
double(deltay)*deltay));
    if(huyen>0)
    { delx = int(deltax * 15/huyen);
      dely = int(deltay * 15/huyen);
    }
    line(x+dau*delx,y-dely,x+dau*(deltax-delx), y -
(deltay-dely) );
  }
  dempt++;
  // Ha tang
  if(dempt>spt)
  { tang++; spt = pow(2,tang); dempt = 1 ;
    deltax =int(getmaxx()/(2*spt)); x = deltax;
    y = y + deltax; dau = 1;}
  else
  {x = x + 2*deltax; dau = - dau; }
}

}
getch(); closegraph();
}

//-----
void DemMuc(kpt *p,int sm)
{
  if(p!=NULL)

```

```

        {
            sm = sm+1;
            if(sm>muc)
                muc=sm;
            DemMuc(p->lptr, sm);
            DemMuc(p->rptr, sm);
        }
    }
//-----

void main()
{ int tam; char chon; kpt *T=NULL;
while (1)
{ clrscr();
printf( "\n\n\n THUC DON CHUONG TRINH CAY NHI TIM KIEM" );
printf( "\n\n 1- Them la cho cay " ) ;
printf( "\n\n 2- Ve cay len man hinh " ) ;
printf( "\n\n 3- Dem muc cay " ) ;
cout<< "\n\n 0- thoat" ;
cout<< "\n\n Moi chon phim tuong ung voi thuc don :";
chon = getch();
switch(chon)
{ case '1':
    clrscr(); printf("Nhap gia tri truong info :");
    scanf("%d", &tam); ThemLa(T,tam); break;
    case '2': vecay(T); break;
    case '3': DemMuc(T,m); clrscr();cout<<"Cay co muc "<<muc;
getch(); break;
    case '0' : break ;
    };
    if (chon == '0')
        break;
    }
}

```

5. KẾT LUẬN

+ Đề tài đã tiến hành phân tích và cài đặt thành công giải xây dựng cây nhị phân tìm kiếm trong bộ nhớ RAM của máy tính.

+ Đề tài đã tiến hành phân tích và cài đặt thành công giải thuật biểu diễn cây nhị phân tìm kiếm ra màn hình máy tính.

+ Với sự thành công của chương trình này, sẽ là một công cụ trực quan sinh động giúp người học dễ dàng nắm bắt hơn về cây nhị phân tìm kiếm

+ Trong quá trình nghiên cứu và thực nghiệm, giải pháp chúng tôi đưa ra không tránh khỏi những sơ xuất với những ứng dụng cụ thể. Rất mong nhận được sự đóng góp của các đồng nghiệp và những độc giả gần xa quan tâm đến lĩnh vực này.

6.TÀI LIỆU THAM KHẢO

1. <https://quantrimang.com/hoc/ham-print-trong-python-163990>
2. Đỗ Xuân Lôi, Đại học Quốc gia Hà Nội, Cấu trúc dữ liệu và giải thuật.
- 3.<https://laptrinhcanban.com/python/nhap-mon-lap-trinh-python>.

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỎ ĐỊA CHẤT**

BÁO CÁO HỌC THUẬT 2025

Tên đề tài :

Nghiên cứu, lập trình
xây dựng chương trình cài đặt
cây nhị phân tìm kiếm trong bộ nhớ RAM
và biểu diễn trên màn hình bằng ngôn ngữ lập trình C⁺⁺.

Tác giả đề tài : *Phạm Đức Hậu, bộ môn HTTT&TT*

Hà Nội 12/2025

