Mã đề
CD 2011 - 02

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
BỘ MÔN KHOA HỌC MÁY TÍNH

***

Họ tên: …… **Lớp**: …… **SHSV**: ……	**ĐỀ THI MÔN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT** **Ngày thi**: ……/……/…. **Thời gian 90'** (**Sinh viên được sử dụng tài liệu**)	Hà nội, ….. /….. / …… ***Trưởng bộ môn***

Bài 1.

a) So sánh ưu nhược điểm khi lưu trữ cây nhị phân chiều cao h dùng: (1) mảng, (2) cấu trúc liên kết **(1 Điểm)**

```
struct BNode
{
    DATA_TYPE data; //là kiểu dữ liệu lưu trữ tại nút
    struct BNode * Lchild, *Rchild; //con trỏ tới cây con trái và con phải
}
```

Theo các tiêu chí:

- Bộ nhớ,
- thời gian truy cập một nút bất kỳ,
- tìm nút cha của một nút bất kỳ trên cây

Biểu diễn cây nhị phân dùng mảng: dùng mảng có số lượng phần tử bằng số lượng nút trên cây nhị phân đầy đủ chiều cao h (số lượng phần tử của mảng sẽ là $2^{h+1}-1$ phần tử).

Với nút thứ i ($i \geq 0$) thì nút con của nó sẽ là $2i+1$ và $2i+2$, nút cha của nó sẽ là $\left\lfloor \frac{i-1}{2} \right\rfloor$

Biểu diễn cây nhị phân dùng cấu trúc liên kết: mỗi nút có thêm hai con trỏ là con trỏ tới cây con trái và con trỏ tới cây con phải.

Tiêu chí	Biểu diễn dùng mảng	Biểu diễn dùng cấu trúc liên kết
Bộ nhớ	**Biểu diễn 1 phần tử**: không cần sử dụng thêm bộ nhớ phụ **Biểu diễn cho cả cây**: luôn cần bộ nhớ $2^{h+1}-1$ cho cây nhị pahan bất kỳ chiều cao h, nên sẽ rất lãng phí bộ nhớ nếu cây nhị phân đó không phải là cây gần hoàn chỉnh.	**Biểu diễn 1 phần tử**: Cần thêm bộ nhớ phụ lưu trữ con trỏ tới cây con trái và cây con phải **Biểu diễn cho cả cây**: Cây có bao nhiêu nút thì cấp phát động bấy nhiêu bộ nhớ. Tiết kiệm bộ nhớ hơn nếu dùng biểu diễn cho cây nhị phân bất kỳ
Thời gian truy cập 1 nút bất kỳ	$O(1)$ Vì mảng hỗ trợ truy cập ngẫu nhiên	$O(h)$ Cấu trúc liên kết không hỗ trợ truy cập ngẫu nhiên, ta phải truy cập thông qua các nút tổ tiên của nút đó
Tìm nút cha của một nút bất kỳ	Thời gian là $O(1)$	Thời gian trung bình cỡ $O(n)$, vì ta phải duyệt từ gốc để tìm tới nút cha của nút đó

b) Đánh giá thời gian thực hiện tồi nhất của hàm sau theo O-lớn **(1 điểm)**

```
double fastPower(double x, int n)
{
   double fract;
   if(n==0) return 1;
   fract = fastPower(x,n/2);

   if(n%2==0) return fract* fract;
   else return fract*fract*x;
}
```

Hàm trên được cài đặt đệ quy, lời gọi đệ quy là `fract = fastPower(x,n/2);`
Được gọi 1 lần trong hàm, ta có công thức đệ quy tổng quát là

$$T(n) = \begin{cases} 1 \text{ nếu } n = 0 \\ T(n/2) + 1 \text{ nếu } n > 0 \end{cases}$$

Ta có thể viết gọn lại là $T(n) = T\left(\frac{n}{2}\right) + 1$
Áp dụng định lý thợ với $a = 1, b = 2$ và $f(n) = 1$
$n^{\log_b a} = n^{\log_2 1} = 1$ → trường hợp 2 của định lý thợ
Vậy kết luận $T(n) = \theta(\log n)$

c) So sánh ưu nhược điểm của phương pháp tổ chức tìm kiếm dùng mảng và áp dụng thuật toán tìm kiếm nhị phân, cây nhị phân tìm kiếm và dùng bảng băm theo các tiêu chí sau **(1 Điểm)**

Tiêu chí	**Tìm kiếm nhị phân**	**Cây nhị phân tìm kiếm**	**Bảng băm**
Bộ nhớ dùng lưu trữ các phần tử	$O(n)$ Tỉ lệ với số phần tử, tuy nhiên mỗi phần tử không phải lưu trữ thêm dữ liệu thừa	$O(n)$ Tỉ lệ với số phần tử, tuy nhiên mỗi phần tử phải lưu trữ thêm dữ liệu thừa (2 con trỏ)	Số lượng ô nhớ xác định trước, là kích thước bảng băm (thường lớn hơn số lượng phần tử cần lưu nhiều lần)
Thời gian tìm kiếm	$O(\log n)$	$O(\log n)$	$O(1)$
Thêm phần tử	$O(n)$	$O(\log n)$	$O(1)$
Xoá phần tử	$O(n)$	$O(\log n)$	$O(1)$
In ra danh sách các phần tử hiện có	$O(n)$	$O(n)$	Không hỗ trợ thao tác này, nếu muốn in ta phải duyệt toàn bộ bảng băm

Bài 2.

a) Biểu thức dạng hậu tố là gì? Ưu điểm của biểu thức dạng hậu tố? **(1 Điểm)**

Biểu thức dạng hậu tố: Là cách biểu diễn biểu thức trong đó toàn tử đứng sau các toán hạng mà nó tác động
Ví dụ: 3 5 + a –
Ưu điểm của biểu thức dạng hậu tố là chỉ có một cách định giá (cách tính) duy nhất. Không như biểu thức dạng trung tố cần quy định thêm về độ ưu tiên của toán tử, và dấu ngoặc.
Biểu thức dạng hậu tố được dùng để biểu diễn biểu thức trong máy tính

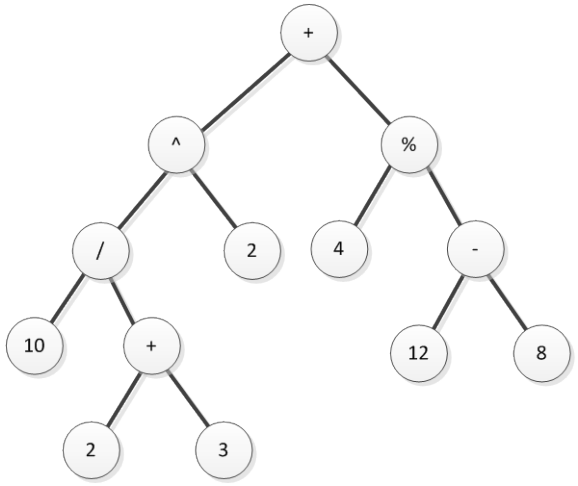
b) Định giá biểu thức dạng hậu tố sau (trình bày rõ các trạng thái trung gian của STACK **(1 Điểm)**

$10\ \ 2\ \ 3\ +\ /\ 2\ \hat{}\ 4\ \ 12\ \ 8\ -\ \%\ +$

Gặp	STACK	Ghi chú
10	10	Gặp toán hạng → đẩy vào stack
2	10, 2	Đỉnh stack ở bên phải nhất
3	10, 2, 3	
+	10, 5	Thực hiện 2+3
/	2	Thực hiện 10/5
2	2, 2	
^	4	Thực hiện 2^2 (2^2)
4	4, 4	
12	4, 4, 12	
8	4, 4, 12, 8	
-	4, 4, 4	Thực hiện 12-8
%	4, 0	Thực hiện 4 %4
+	4	Thực hiện 4 + 0

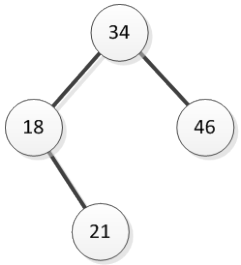
Kết quả cuối cùng là 4

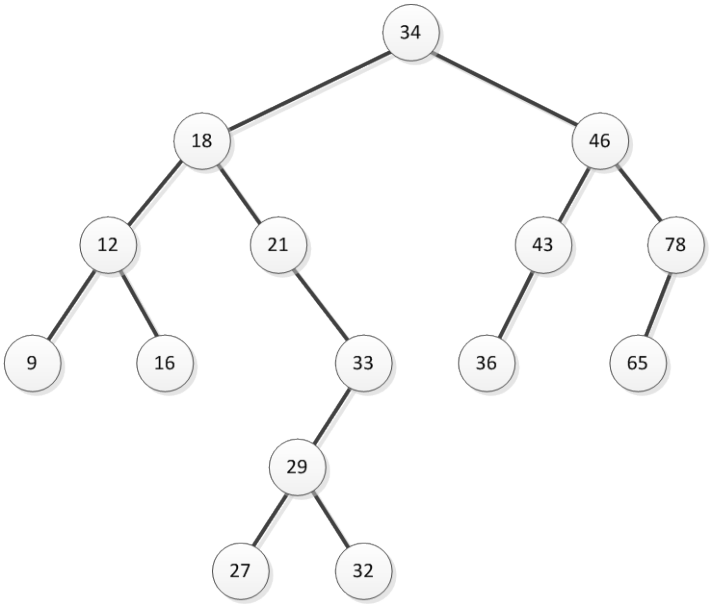
c) Vẽ cây biểu thức biểu diễn cho biểu thức ở phần b (không cần phải trình bày các bước trung gian) **(1 Điểm)**



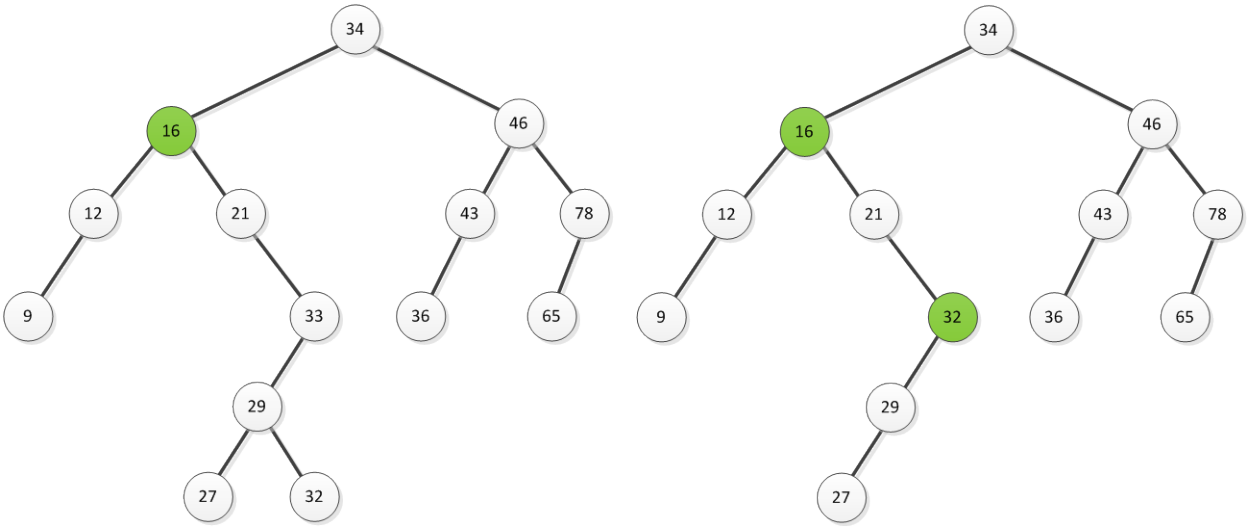
Bài 3.

a) Cho cây nhị phân tìm kiếm ban đầu như hình
thêm lần lượt dãy khóa 33, 43, 12, 36, 78, 29, 16, 9, 65, 27, 32. Hãy vẽ cây nhị phân kết quả thu được cuối cùng (không cần trình bày các bước trung gian).
(1 Điểm)





b) Với cây nhị phân tìm kiếm thu được ở phần a, thực hiện xóa lần lượt khóa 18 và 33.
Hãy vẽ cây kết quả thu được sau mỗi lần xóa
Chú ý: chọn nút thay thế là nút phải nhất trên cây con trái
(1 Điểm)

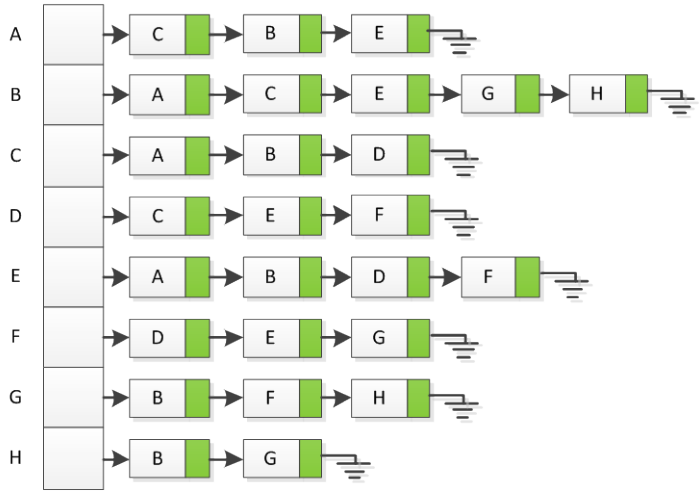


Bài 4. Cho một ***đơn đồ thị vô hướng*** $G(V, E)$ như sau

$V = \{A, B, C, D, E, F, G, H\}$

$E = \{(A, B), (A, C), (A, E), (B, E), (B, G), (C, D), (C, B), (D, E), (F, D), (F, E), (F, G), (H, B), (H, G)\}$

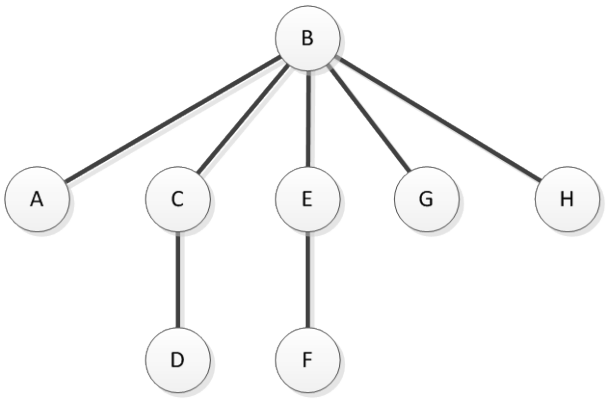
a) Hãy biểu diễn đồ thị trên dùng danh sách kề **(1 Điểm)**



b) Thực hiện BFS từ đỉnh B, hãy đưa ra thứ tự các đỉnh được thăm. **(1 Điểm)**

(Chỉ cần vẽ được hình trạng hàng đợi hoặc cây khung BFS là được đủ điểm)

STT	QUEUE	Ghi chú
0	B	
1	A, C, E, G, H	Lấy B ra, đưa các đỉnh kề với B mà chưa thăm vào QUEUE
2	C, E, G, H	
3	E, G, H, D	
4	G, H, D, F	
5	H, D, F	
6	D, F	
7	F	
8	Ø	



Cây khung BFS từ B

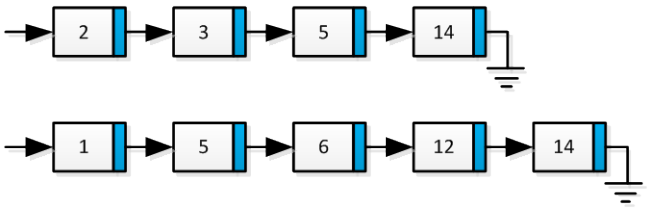
c) Hãy đưa ra các loại cạnh thu được khi BFS tại đỉnh B (BackEdge, CrossEdge, TreeEdge và ForwardEdge).
Lưu ý: Các đỉnh trên đồ thị được thăm theo thứ tự ABC **(1 Điểm)**

Phân loại cạnh

Cạnh cây Tree-Edge	BA,BV,BE,BG,BH,CD,EF
Cạnh ngược Back-edge	
Cạnh tới Forward-Edge	
Cạnh vòng Cross-Edge	AC, AE, ED, GF, GH, DF

Bài 5. Để biểu diễn các tập hợp số nguyên ta dùng danh sách liên kết đơn với cấu trúc một phần tử được khai báo như sau:

```
typedef struct Node
{
     int data;
     struct node *pNext;
} NODE;
```



a) Hãy xây dựng hàm tìm và trả về giá trị phần tử chẵn lớn nhất trong tập hợp trong trường hợp biết các phần tử của tập hợp được sắp xếp theo thứ tự tăng dần về giá trị. **(1 Điểm)**

```
int FindMax (NODE *pHead)
{

}
```

b) Hãy đánh giá thời gian thực hiện trong trường hợp tồi nhất của hàm bạn viết theo O-lớn **(0.5 điểm)**

```
int FindMax (NODE *pHead)
{
    NODE *ptr = pHead;
    int i= 0; //biến để đếm vị trí các phần tử
    int pos=-1;
    while(ptr!=NULL)
    {
           if(ptr->data%2==0)
           {
                  pos = i;
           }
           ptr = ptr->pNext;
           i++;
    }
    return pos;
}
```

Dễ thấy thời gian thực hiện của thuật toán trong trường hợp tồi nhất cỡ $O(n)$

Tổng điểm 12.5 Điểm