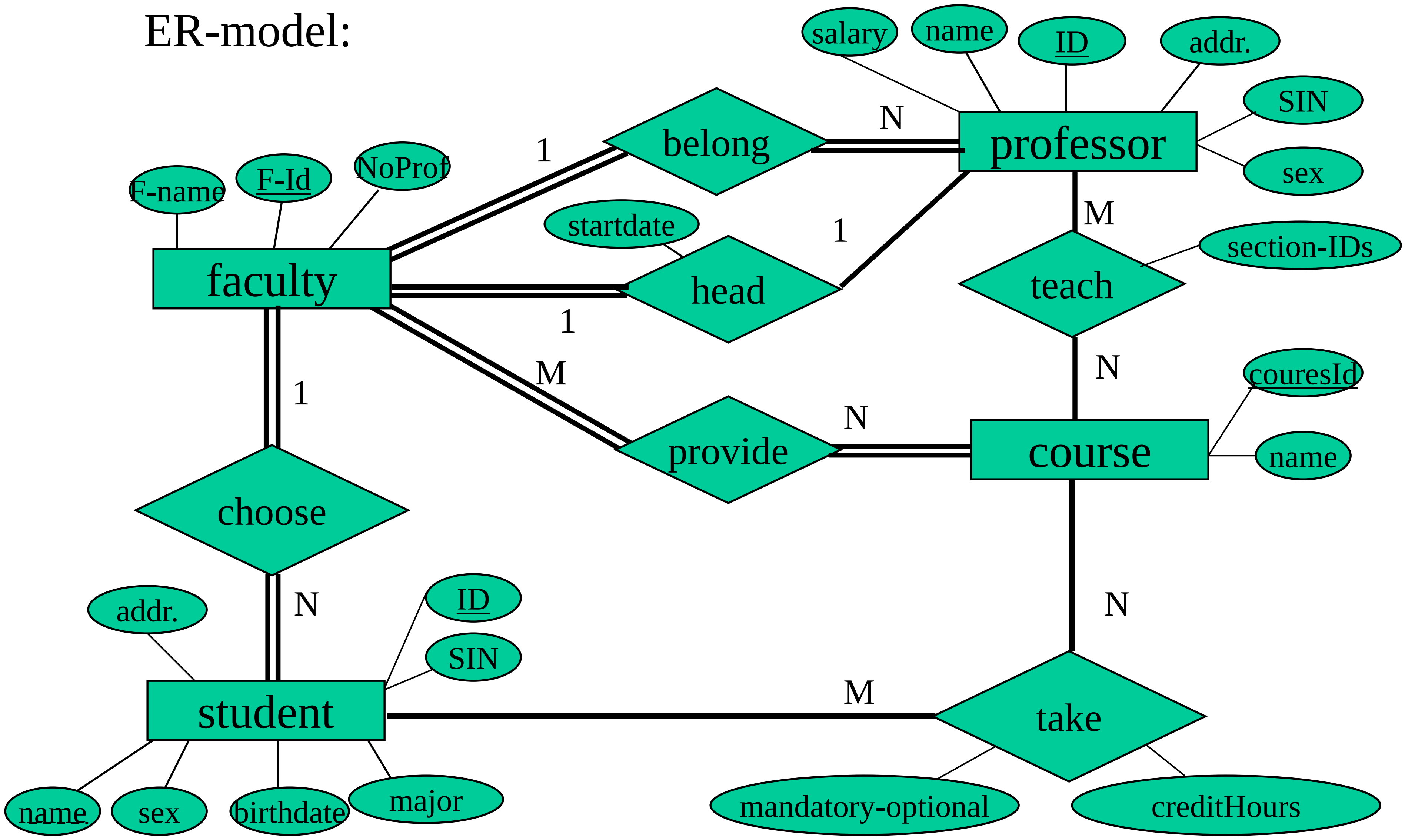
Analysis of Midterm-Examination

1.(10) Draw an ER-diagram to describe the following real world problem.

(a) A university is organized into faculties.
(b) Each faculty has a unique name, ID and number of professors and a specific professor is chosen as the faculty head.
(c) Each faculty provides a number of courses.
(d) Each course has a unique name and courseID.
(e) Each professor has a name, SIN, address, salary, sex and courses taught by him/her.
(f) Each professor belongs to a faculty and can teach several sections of a course.
(g) Each student has a name, ID, SIN, address, GPA, sex, and major.
(h) Each student can choose one faculty as his/her major faculty and take several courses with certain credit hours. Some of the courses are mandatory and some are optional.

Analysis of Midterm-Examination

ER-model:

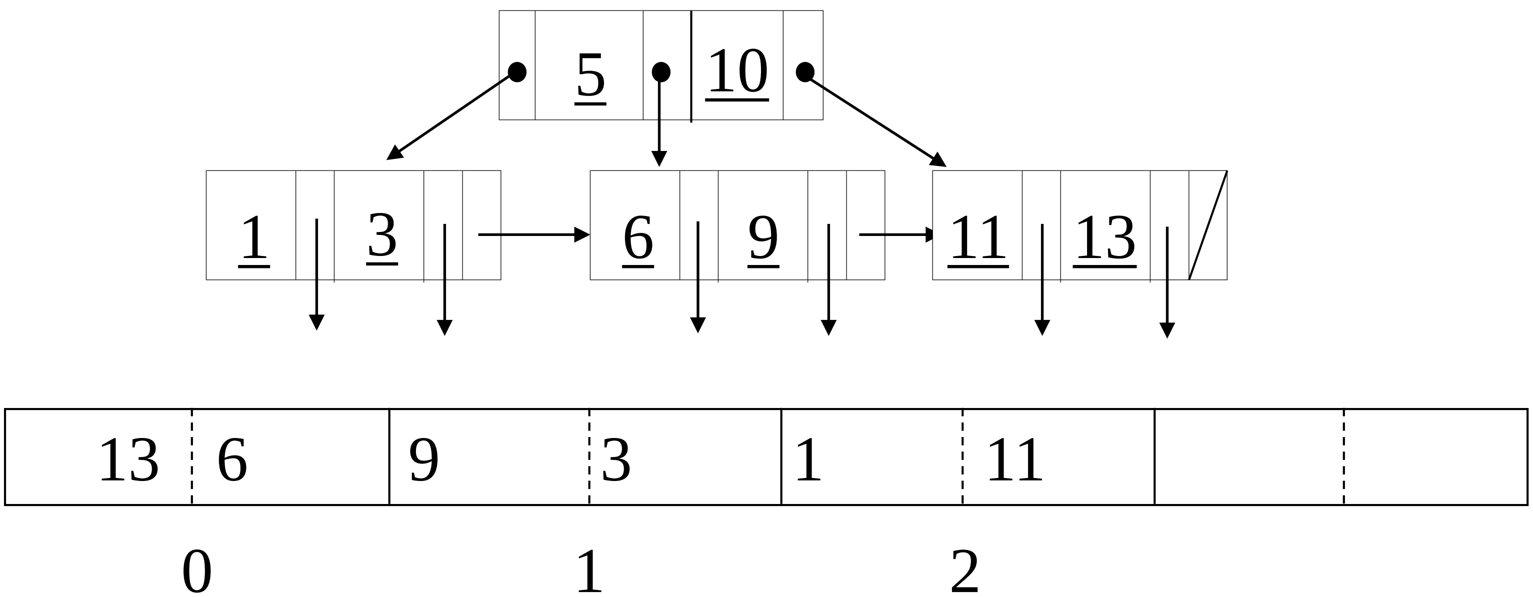


2. (20) (a) The following is the algorithm to search a tree in depth-first manner. Change it to an algorithm to store a B+-tree (a linked list stored in main memory) in a data file. (10)

push(root); (*push the root into stack.*)
while (stack is not empty) do
{ v := pop( );
print(v); (*or store v in a file.*)
let v_1, …, v_k be the children of v;
for (i = k to 1) {push(v_i)};
}

Analysis of Midterm-Examination

(b) Apply the algorithm to the tree shown in Fig. 1 and give the result (i.e., the data file storing the tree). (10)



Store a B+-tree on hard disk

data	address-of-parent	position

stack: S

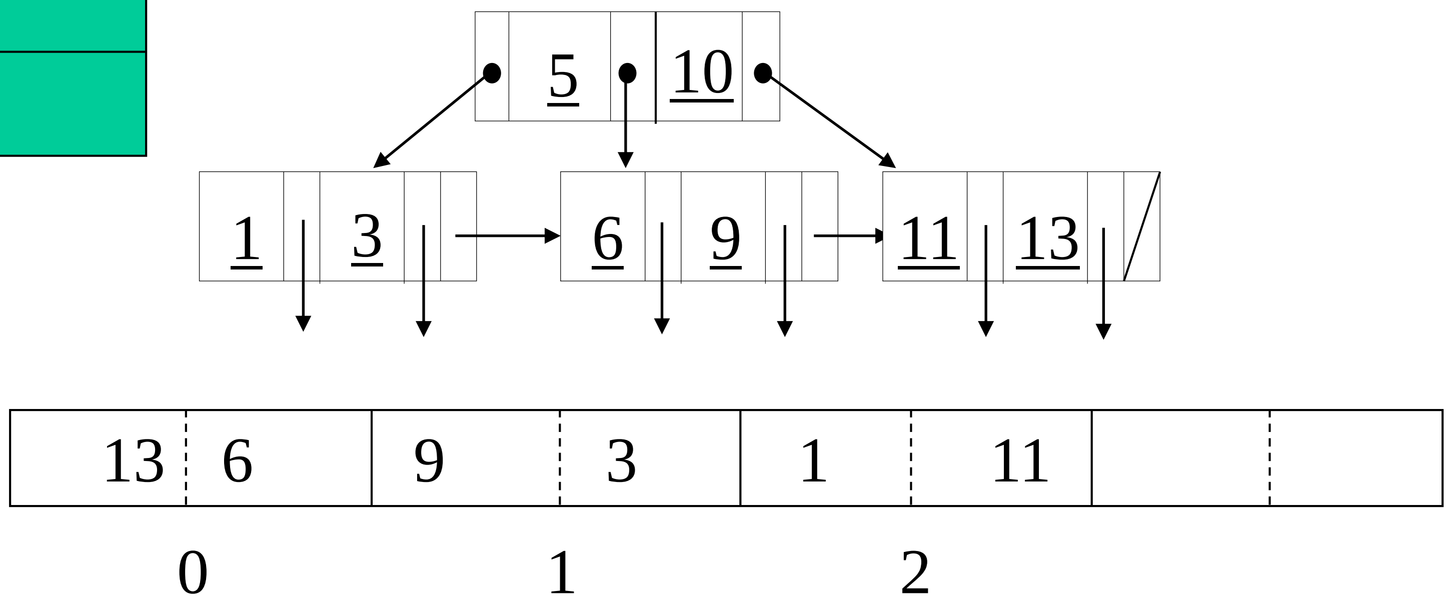
Algorithm:

push(root, -1, -1);
while (S is not empty) do
{   x := pop( );
    store x.data in file F;
    assume that the address of x in F is ad;
    if x.address-of-parent $\neq$ -1 then {
        y := x.address-of-parent;
        z := x.position;
        write ad in page y at position z in F;
    }
    let x_1, …, x_k be the children of v;
    for (i = k to 1) {push(x_i, ad, i)};
}

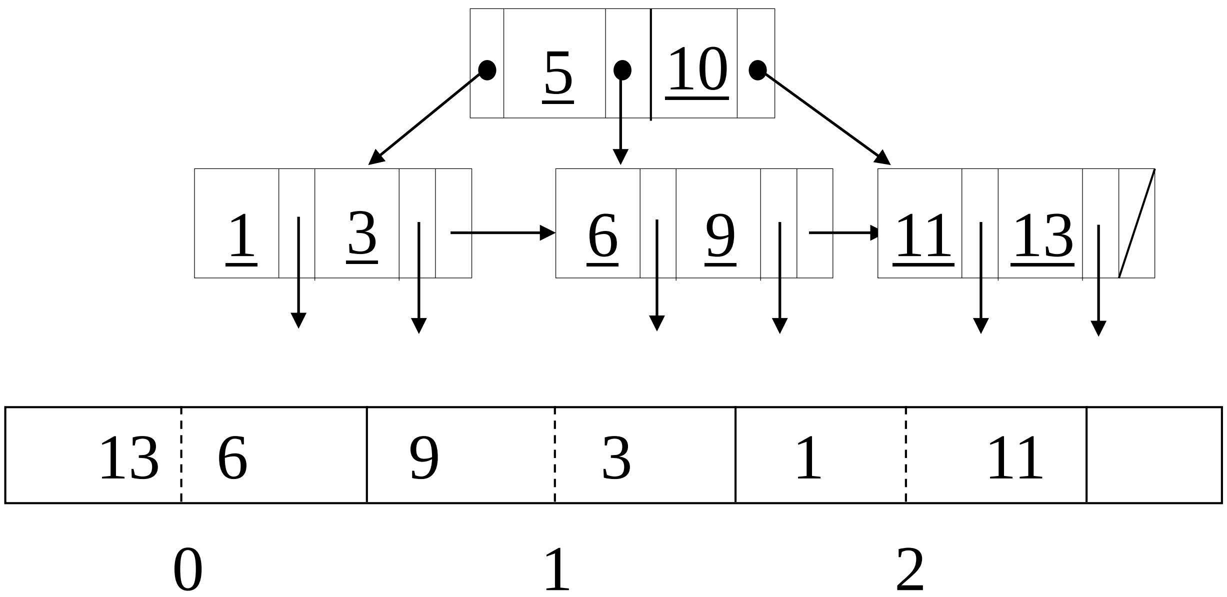
Analysis of Midterm-Examination

B+-tree stored in a file:

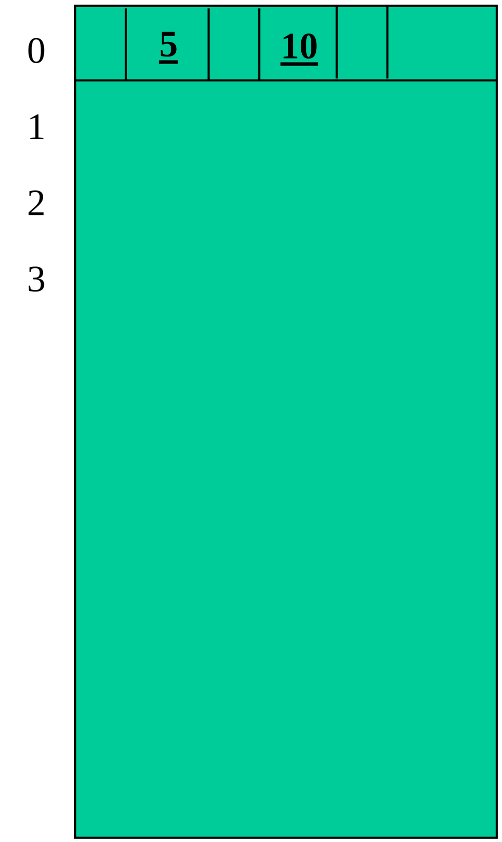
0	1	5	2	10	3
1	1	2	3	1	
2	6	0	9	1	
3	11	2	13	0	



Analysis of Midterm-Examination



B+-tree stored in a file:

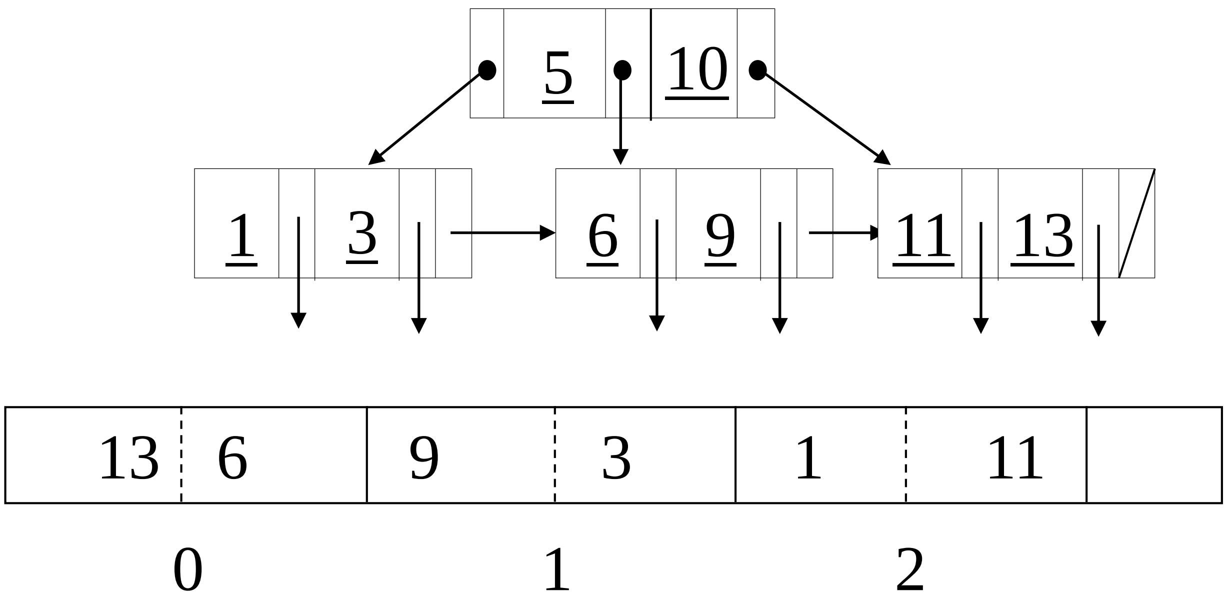


Stack:

5,10	-1	-1

1, 3	0	1
6, 9	0	2
11, 13	0	3

Analysis of Midterm-Examination



B+-tree stored in a file:

0	1	5	2	10	3
1	1	2	3	1	
2	6	0	9	1	
3	11	2	13	0	

Stack:

1, 3	0	1
6, 9	0	2
11, 13	0	3

→

6, 9	0	2
11, 13	0	3

3.(5) Given the relation schemas shown in Fig. 2, construct expressions (using SQL language) to evaluate the following query:
Find the names of employees who works on all the projects controlled by department ‘Applied Computer Science’.

EMPLOYEE

fname, minit, lname, <u>ssn</u>, bdate, address, sex, salary, superssn, dno

DEPARTMENT

Dname, <u>dnumber</u>, mgrssn, mgrstartdate

Fig. 2

PROJECT

Pname, <u>pnumber</u>, plocation, dnum

WORKS_ON

<u>Essn pno</u>, hours

Analysis of Midterm-Examination

```
SELECT E.FNAME,E.MINIT,E.LNAME
FROM EMPLOYEE E
WHERE
NOT EXISTS
    (SELECT P.pname FROM PROJECT AS P, DEPARTMENT D
    WHERE P.dnum = D.dnumber AND
        D.DNAME = "Applied Computer Science"))
        NOT EXISTS
            (SELECT * FROM WORKS_ON AS W,
            WHERE P.pnumber = W.pno AND
            W.essn=E.ssn ))
```

There is no project that the employee does not work on.

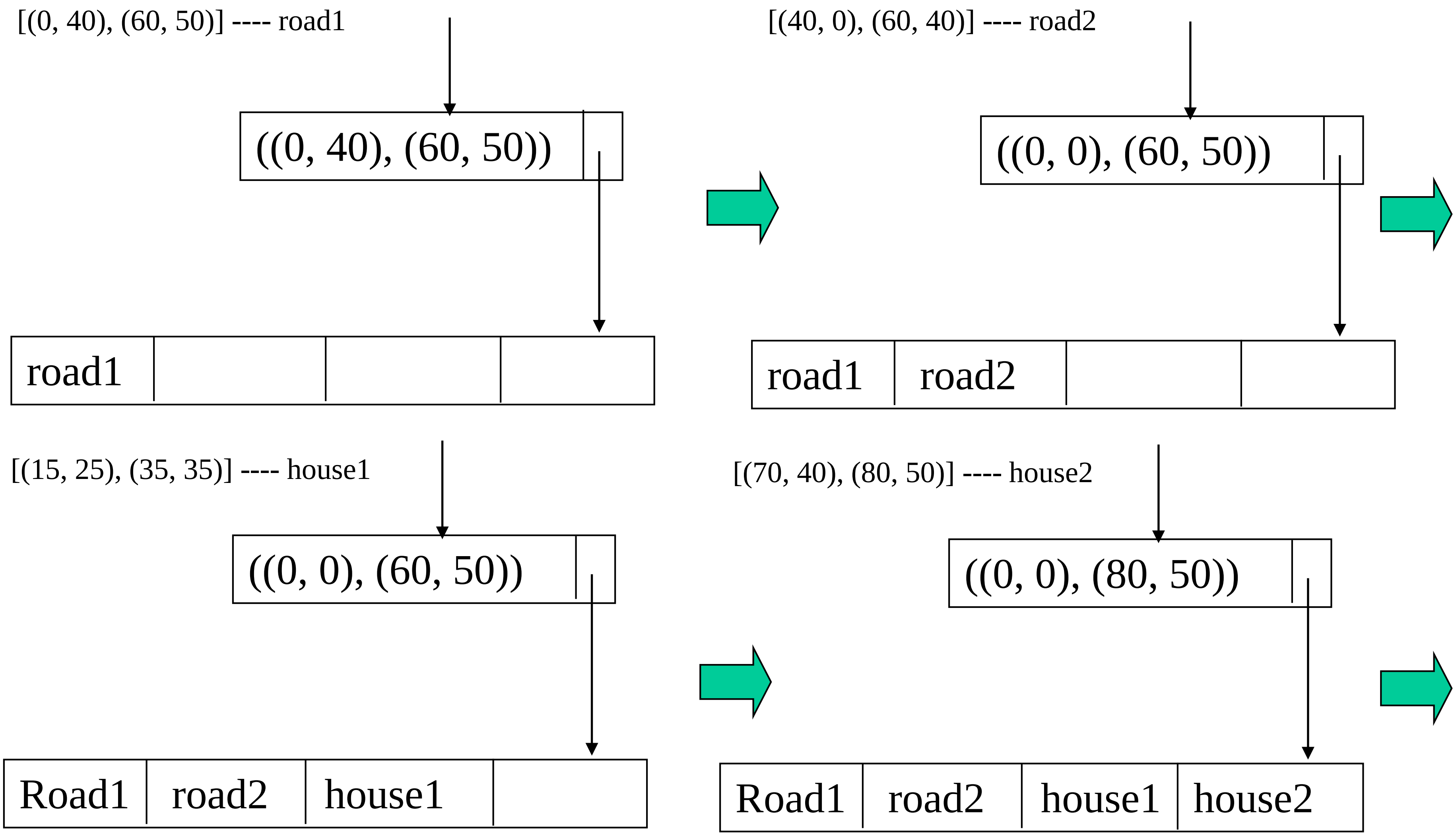
controlled by Department of "Applied Computer Science"

4. (20) Construct an R-tree over a set of records for geographical objects with the following coordinates [(x1, y1), (x2, y2)]:

[(0, 40), (60, 50)] ---- road1
[(40, 0), (60, 40)] ---- road2
[(15, 25), (35, 35)] ---- house1
[(70, 40), (80, 50)] ---- house2
[(70, 5), (80, 15)] ---- house3
[(35, 25), (80, 35)] ---- pipeline

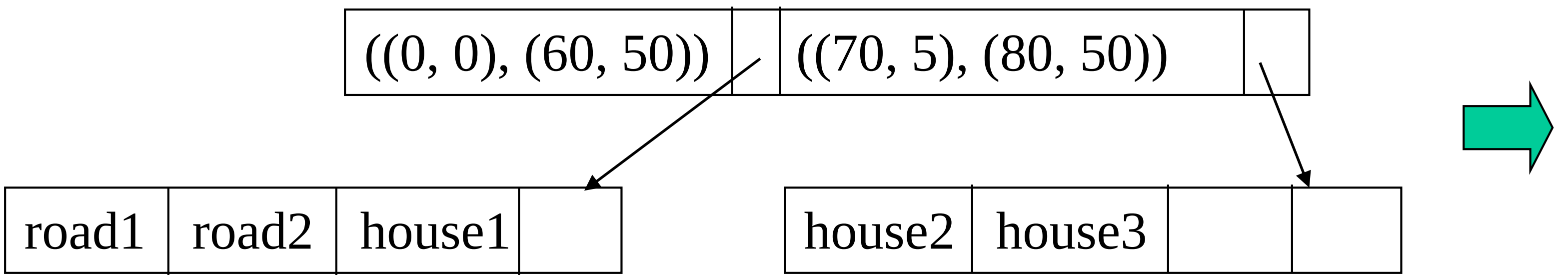
Assume that each leaf node can have at most 4 pointers and at least two pointers; and each internal node at most 2 pointers and at least 1 pointer.
Please give the computation process.

Analysis of Midterm-Examination



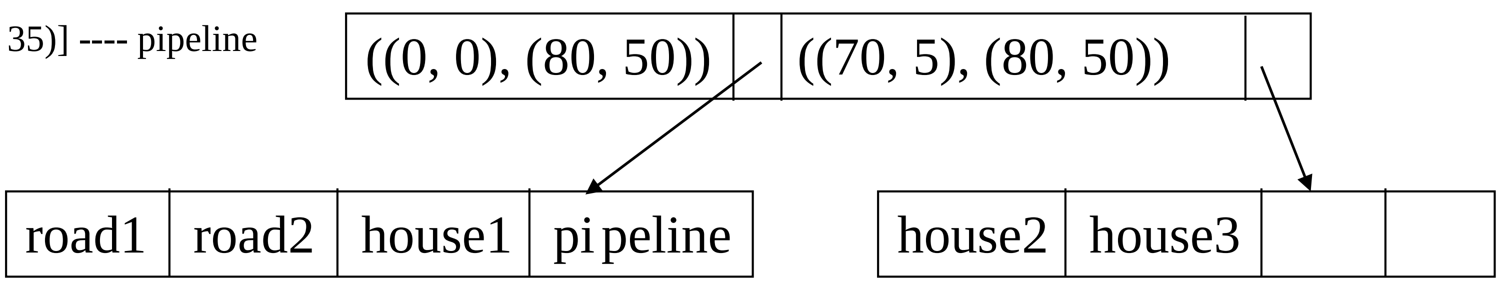
Analysis of Midterm-Examination

[(70, 5), (80, 15)] ---- house3

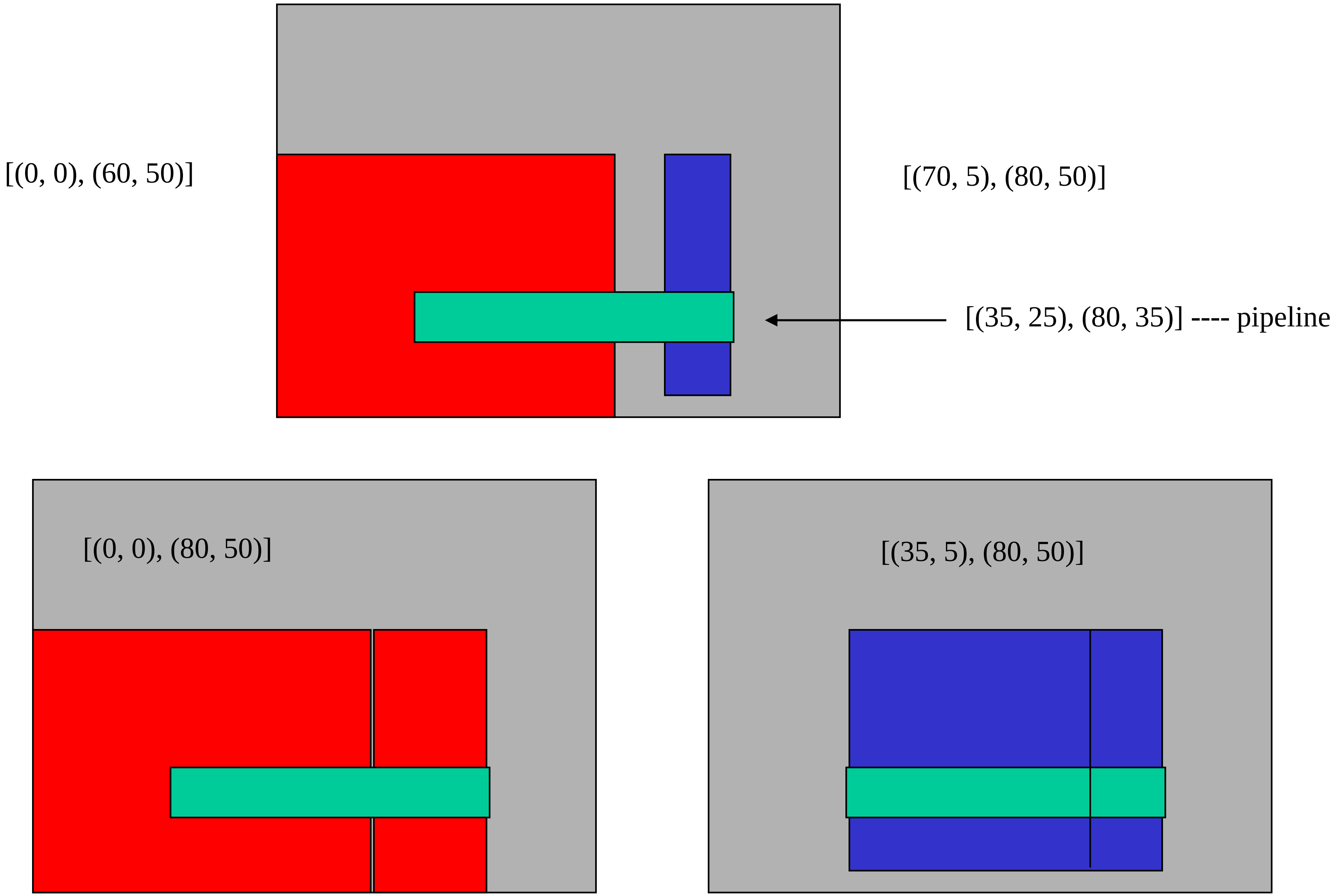


- If we expand the first subregion in the internal node, then we add 1000 square units to the region.
- If we extend the other subregion in the internal, then we add 1575 square units.

[(35, 25), (80, 35)] ---- pipeline

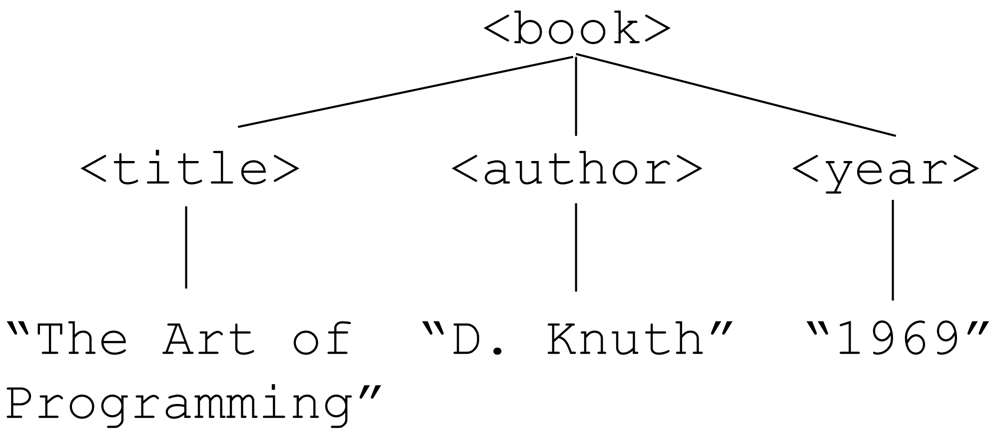


Analysis of Midterm-Examination



5. (15) Given the algorithm for transforming any XML document to a tree structure, apply the algorithm to the following document and trace the computation process.

<book>
<title>
“The Art of Programming”
</title>
<author>
“D. Knuth”
</author>
<year>
“1969”
</year>
</book>



Transform an XML document to a tree

Read a file into a character array `A`:

<	b	o	o	k	>	<	t	i	t	l	e	>	“	T	h	e		A	r	t	…

stack S:

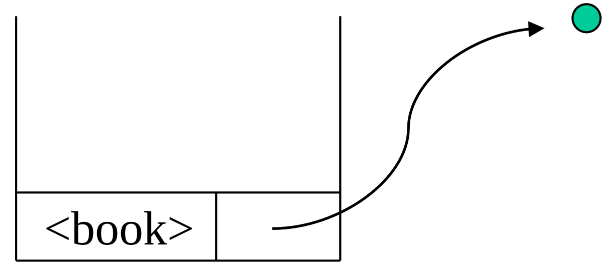
node_value	Pointer_to_node

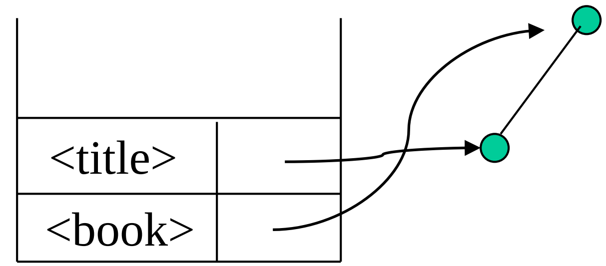
Transform an XML document to a tree

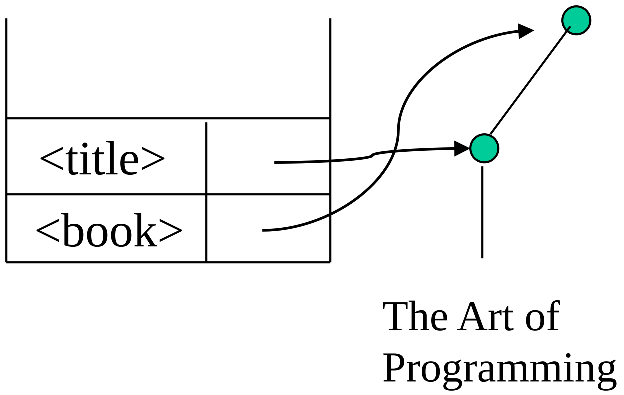
```
Algorithm:

Scan array A;
If A[i] is '<' and A[i+1] is a character then {
      generate a node x for A[i..j],
      where A[j] is '>' directly after A[i];
      let y = S.top().pointer_to_node;
      make x be a child of y; S.push(A[i..j], x);
If A[i] is ' " ', then {
      genearte a node x for A[i..j],
      where A[j] is ' " ' directly after A[i];
      let y = S.top().pointer_to_node;
      make x be a child of y;
If A[i] is '<' and A[i+1] is '/', then S.pop();
```

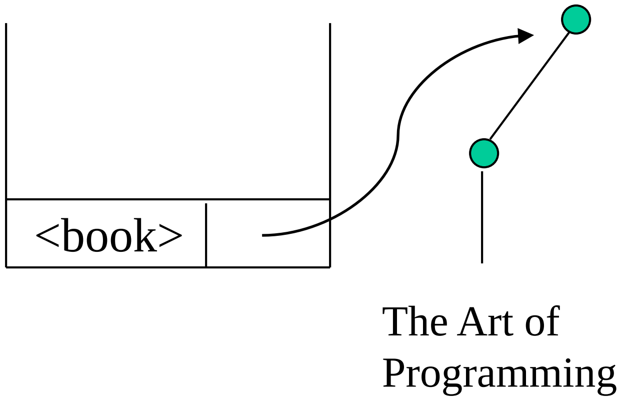
Analysis of Midterm-Examination

```
<book>
        <title>
                "The Art of Programming"
        </title>
        <author>
                "D. Knuth"
        </author>
        <year>
                "1969"
        </year>
</book>
```



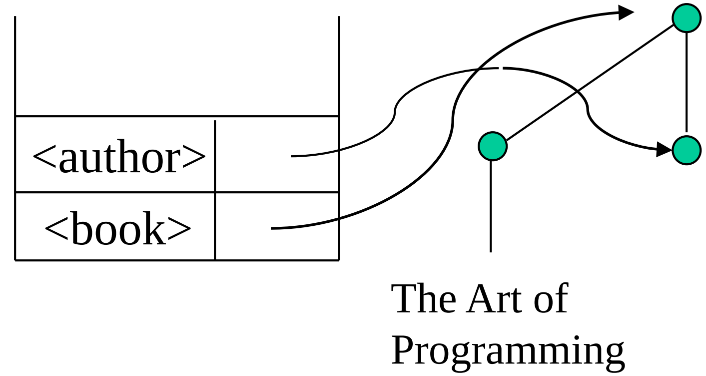


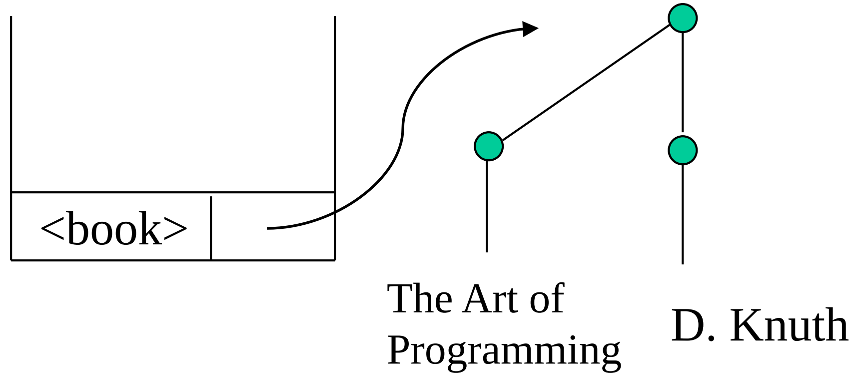


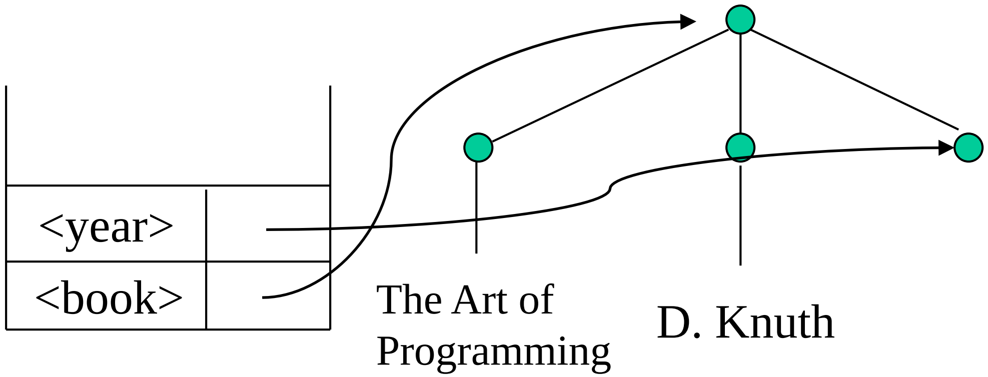


Analysis of Midterm-Examination

```
<book>
    <title>
        "The Art of Programming"
    </title>
    <author>
        "D. Knuth"
    </author>
    <year>
        "1969"
    </year>
</book>
```

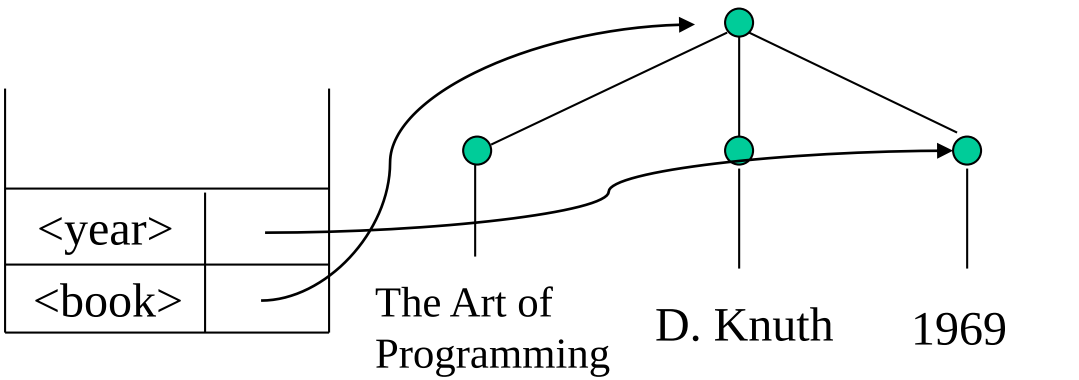


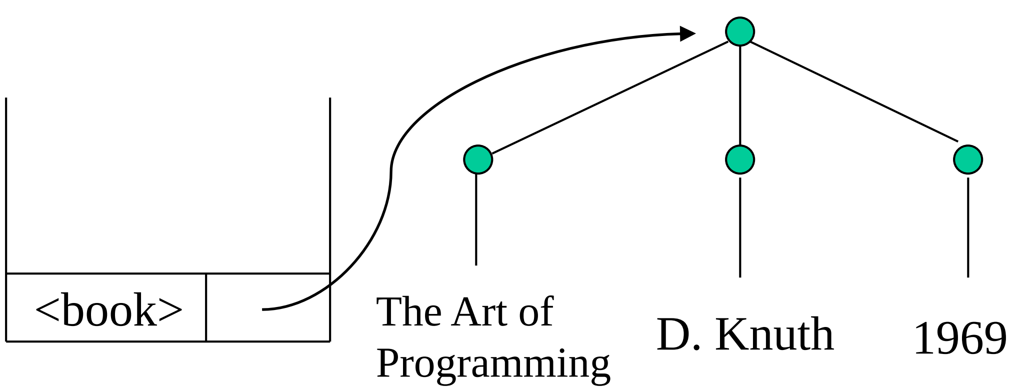


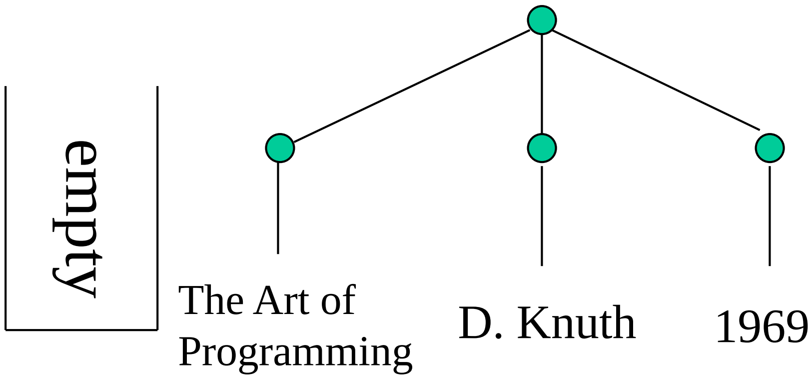


Analysis of Midterm-Examination

```
<book>
    <title>
        "The Art of Programming"
    </title>
    <author>
        "D. Knuth"
    </author>
    <year>
        "1969"
    </year>
</book>
```







6. (10) Fig. 3 is a DTD for a set of XML documents on movie and stars. Please produce a FLWR expression to find all those stars, who live at 123 Maple St., Malibu.

```
<!DOCTYPE Stars [
    <!ELEMENT Stars (Star*)>
    <!ELEMENT Star (Name, Address+, Movies)>
    <!ELEMENT Name (#PCDATA)>
    <!ELEMENT Address (Street, City)>
    <!ELEMENT Street (#PCDATA)>
    <!ELEMENT City (#PCDATA)>
    <!ELEMENT Movies (Movie*)>
    <!ELEMENT Movie (Title, Year)>
    <!ELEMENT Title (#PCDATA)>
    <!ELEMENT Year (#PCDATA)>
]>
```

Fig. 3

Assume that all the documents are stored in a file “stars.xml”.

```
let     $stars := doc(“stars.xml”)
for     $s in $stars/Stars/Star, $s1 in $s/Address
where   $s1/Street = “123 Maple St.” and
        $s1//City = “Malibu”
return $s/Name
```

```
let     $stars := doc(“stars.xml”)
for     $s in $stars/Stars/Star/Address
where   $s/Street = “123 Maple St.” and
        $s//City = “Malibu”
return $s/Name
```

```
let    $stars := doc(“stars.xml”)
for    $s in $stars/Stars/Star
where $s/Address/Street = “123 Maple St.”
       and $s/Address/City = “Malibu”
return $s/Name
```

← This is wrong!

Query: find all the stars that live at 123 Maple St., Malibu.

The following FLWR seems correct. But it does not work.

```
let      $stars := doc("stars.xml")
for      $s in $stars/Stars/Star
where $s/Address/Street = "123 Maple St."
         and $s/Address/City = "Malibu"
return $s/Name
```

Correct query:

```
let      $stars := doc("stars.xml")
for      $s in $stars/Stars/Star,
         $s1 in $s/Address
where $s1/Street = "123 Maple St." and
         $s1//City = "Malibu"
return $s/Name
```

```
<? Xml version = "1.0" encoding = "utf-8" … ?>
<Stars>
    <Star>
        <Name>Fay Wray</Name>
        <Address>
            <Street>123 Maples St.</street>
            <City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ave.</Street>
            <City>Mallibu</City>
        </Address>
    </Star>
            … more stars
</Stars>
```

7. (10) The following is an XML schema, please define a DTD which is equivalent to it.

```
<? Xml version = “1.0” encoding = “utf-8” ?>
<xs: schema xmlns: xs = “http://www.w3.org/2001/XMLSchema”>

<xs: complexType name = “movieType”>
        <xs: attribute name = “title” type = “xs: string” use = “required” />
        <xs: attribute name = “year” type = “xs: integer” use = “required” />
</xs: complexType>

<xs: element name = “Movies”>
        <xs: complexTyp>
                <xs: sequence>
                        <xs: element name = “Movie” type = “movieType”
                                minOccurs = “0” maxOcurs = “unbouned” />
                        <xs: element name = “Star” type = “xs:string”
                                minOccurs = “0” maxOcurs = “unbouned” />
                </xs: sequence>
        </xs: complexTyp>
</xs: element>
</xs: schema>
```

Answer:

```
<!DOCTYPE Movies [
        <!ELEMENT Movies (Movie*, Star*) >
        <!ELEMENT Movie EMPTY >
                <!ATTLIST  Movie
                        Title  CDATA #REQUIRED
                        Year   CDATA #REQUIRED
                >
        <!ELEMENT Star   #PCDATA >
]>
```

8. (10) Concerning the linear hash, answer the following questions:

(a) What is a phase?
(b) When to split a bucket?
(c) How to split a bucket?
(d) What bucket will be chosen to split next?
(e) How do we find a record inserted into a linear hashing file?

Answer:

a) In the linear hash process, a series of hash functions: h_0, h_1, h_2, … are used. In each phase i, h_i and h_{i+1} will be used. When the size of the primary area is doubled, phase i is completed.

$$h_i(\text{key}) = \text{key mod } (2^i * M),$$

Answer:

b) When an overflow occurs or when the load factor becomes larger than a certain threshold, we choose a bucket to split.
c) In phase i, we use h_{i+1} to split the records in the corresponding bucket.
d) A pointer variable n is maintained to indicate what bucked is chosen to split. Initially, n is set to 0. After each splitting n is increased by 1.
e) When we want to find a certain key value k in a linear hash file, we will try $h_j(k)$, and $h_{j+1}(k)$, where j is the number of phases made during the hash file's construction.