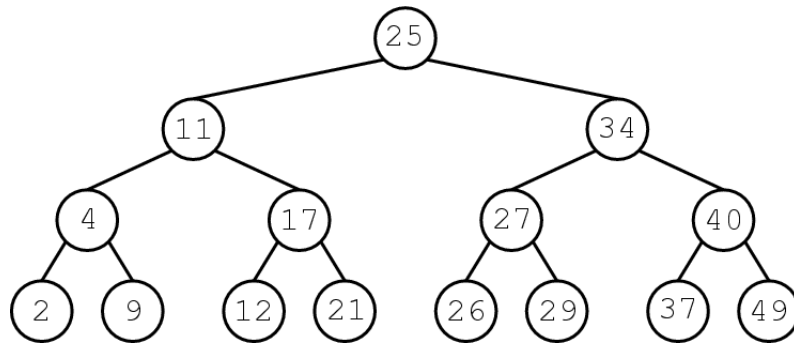


Midtsemestertest 2
TDT4120 Algoritmer og Datastrukturer
8. oktober 2003
LØSNINGSFORSLAG

1. Sett inn disse nøklene på riktig sted i det balanserte binære søketreet under:

2 4 9 11 12 17 21 25 26 27 29 34 37 40 49

Svar:



Figur 1: Binærtre

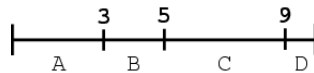
2. Anta at du har et binært søketre med n noder. Du har fått i oppgave å skrive en algoritme som skal hente ut et sammenhengende intervall bestående av totalt k nøkler fra dette søketreet. Algoritmen skal ta inn hvilken nøkkel den skal starte på og antall nøkler den skal hente ut (k). Hva blir kjøretiden til denne algoritmen uttrykt ved k og n ?

Svar:

$O(\log(n) + k)$, siden man kan søke frem et element i et binærtre i $O(\log(n))$ tid og hente ut de neste k elementene med Inorder tree walk.

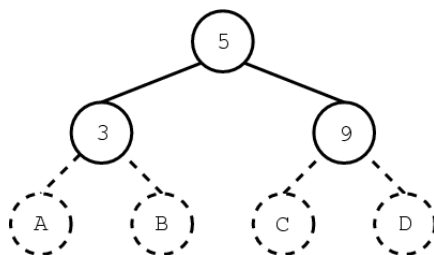
3. Binære søketrær brukes normalt til å organisere poster som er identifisert med én søke-nøkkel.

Gitt at vi har nøklene 3, 5 og 9 vil et 1-dimensjonalt intervall deles opp sånn:



Figur 2: 1-dimensjonalt intervall

Ved innsetning av nye noder vil punkter i verdiområdene A, B, C og D settes inn på sine respektive grener i det binære søketreet:



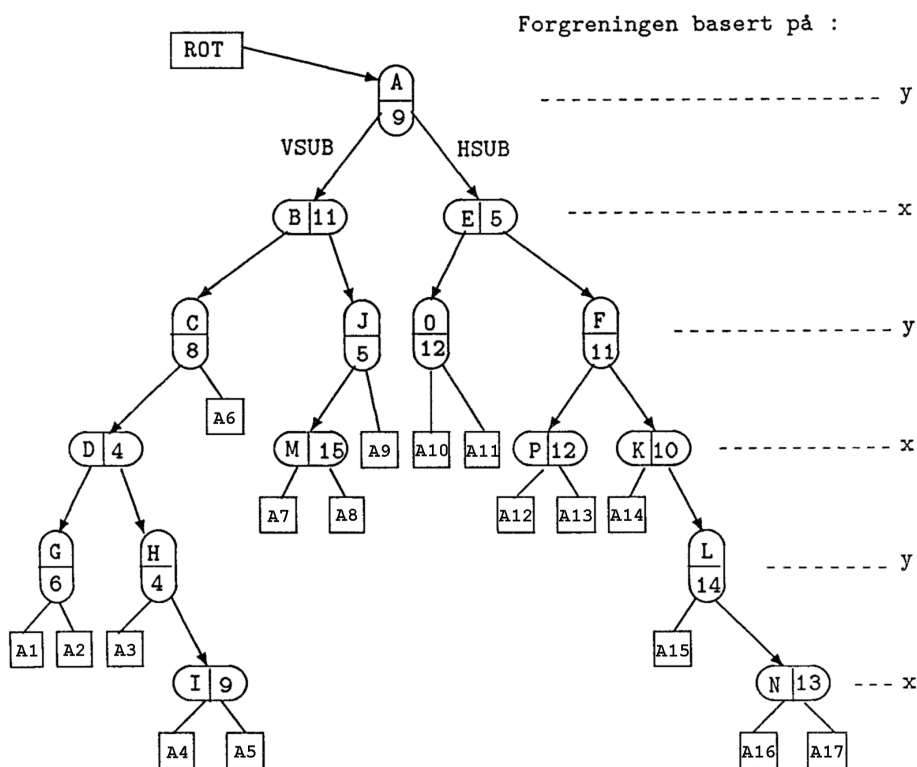
Figur 3: 1-dimensjonalt søketre

Idéen med binære søketrær kan generaliseres til poster som er identifisert med mer enn en nøkkel. Vi skal her tenke oss at en post er entydig identifisert med to heltallskoordinater i (x, y) -planet.

Ved innsetting/ettersøking av poster basert på en (x, y) -nøkkel, kan vi la y bestemme forgreningen på 1. nivå i søketreet, x bestemme forgreningen på 2. nivå i søketreet, og deretter la y og x alternere som forgreningsverdier nedover i treet.

Dersom vi eksempelvis har gitt et tre der postene (A, B, \dots, P) har følgende (x, y) -nøkler:

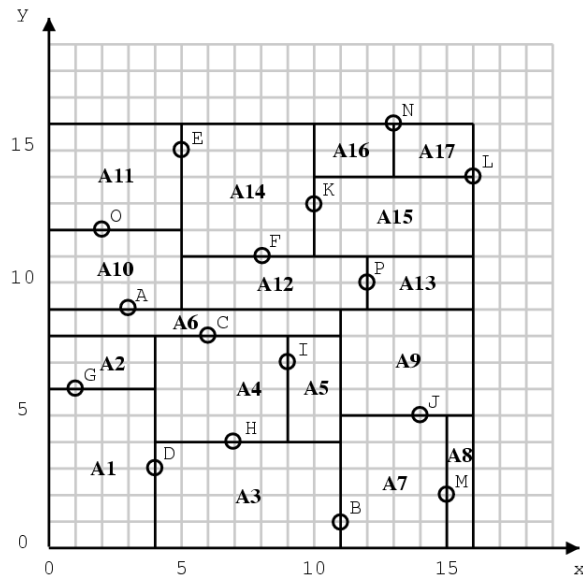
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
x :	3	11	6	4	5	8	1	7	9	14	10	16	15	13	2	12
y :	9	1	8	3	15	11	6	4	7	5	13	14	2	16	12	10



Figur 4: 2-dimensjonalt søketre

Figur 2 viste hvordan et vanlig binærtree deler opp et 1-dimensjonalt intervall. Illustrér i figuren nedenfor den rektangulære oppdelingen av (x, y) -planet som svarer til det 2-dimensjonale binærtreet gitt ovenfor. Plasseringen av punktene A, B, ..., P er allerede avmerket. Marker også i hvilke av de avgrensede områdene punktene A1, A2, ..., A17 vil finne seg.

Svar:



4. Node.java:

```
public class Node {

    public int xValue;
    public int yValue;
    public boolean splits_x_direction;
    public Node left;
    public Node right;

    public Node(int x, int y, boolean splits_x) {
        xValue = x;
        yValue = y;
        splits_x_direction = splits_x;
    }
}
```

Gitt denne javakoden, skriv pseudokode eller javakode for å sette inn et nytt punkt i treet. Anta at du lager en funksjon med rot-node og punkt som parameter. I Java noe tilsvarende `sett_inn(Node rot, int xVerdi, int yVerdi)`.

(a) med en rekursiv løsning

Svar:

```
//To insert a new point into the tree, one calls the method with the
//root of the tree as the first parameter
public static void recursive(Node node, int xValue, int yValue){
    //The values to be compared when deciding to go
    //left or right in the tree (either x og y-values):
    int nodeValue, newValue;

    if (node.splits_x_direction){
        nodeValue = node.xValue;
        newValue = xValue;
    } else {
        nodeValue = node.yValue;
        newValue = yValue;
    }

    if (newValue < nodeValue){
        //Go to left / make new node to the left
        if (node.left != null){
            recursive(node.left, xValue, yValue);
        } else {
            node.left = new Node(xValue, yValue, !node.splits_x_direction);
        }
    }

    } else {
        //Go to right/ make new node to the right
        if (node.right != null){
            recursive(node.right, xValue, yValue);
        } else {
            node.right = new Node(xValue, yValue, !node.splits_x_direction);
        }
    }
}
```

(b) med en iterativ løsning

Svar:

```
public static void iterative(Node node, int xValue, int yValue){
    //The values to be compared when deciding to go
    //left or right in the tree (either x og y-values):
    int nodeValue, newValue;
    boolean splits_x = false;

    //while(true) means that the loop will go on until a
    //break-statement is reached
    while (true){
        if (node.splits_x_direction){
            nodeValue = node.xValue;
            newValue = xValue;
        } else {
            nodeValue = node.yValue;
            newValue = yValue;
        }

        if (newValue < nodeValue){
            //Go to left / make new node to the left
            if (node.left != null){
                node = node.left;
            } else {
                node.left = new Node(xValue, yValue, !node.splits_x_direction);
                break;
            }
        } else {
            //Go to right / make new node to the right
            if (node.right != null){
                node = node.right;
            } else {
                node.right = new Node(xValue, yValue, !node.splits_x_direction);
                break;
            }
        }
    }
}
```

5. Vi har sett på søketre i to dimensjoner. Hvordan vil du lage søketre for k dimensjoner?
Skriv kort og overordnet idé:

Etter samme prinsipp som et søketre for to dimensjoner. På første nivå lar man den første dimensjonen bestemme forgreningen, på andre nivå lar man den andre dimensjonen bestemme forgreningen, osv til den k-te dimensjonen. Starter så igjen på den første dimensjonen for det (k+1)-te nivået.