Fundamental Computer Science I Final

Course Fundamental Computer Science, Dr. Holger Kenn
e-mail: h.kenn@iu-bremen.de, tel.:+49 421 200 3112

Name:

Matriculation Number:

INSTRUCTIONS: Read **all** the problems carefully before you start working. The number of points given for a problem are a rough indication of its difficulty or the time it takes to write them down. Start with the simple problems. Most problems can be answered in a few lines of text or equations. Don't get stuck with the algorithm writing tasks, first try to get an idea how the algorithm works and sketch it for yourself in plain text. Leave the detailed writing of the longer algorithms until the end.

1.) Algorithms

Given this algorithm in pseudocode that multiplies two $n \times n$ matrices A and B.

MATRIXMULTIPLY(A, B)

```
n ← rows[L]
let C be an n × n matrix.
for i ← 1 to n do
    for j ← 1 to n do
        c_ij ← 0
        for k ← 1 to n do
            c_ij ← c_ij + a_ik · b_kj
        end for
    end for
end for
return C
```

a) What is the running time of this algorithm in asymptotic notation? Give all bounds. (2P)

b) Modify the algorithm so that it multiplies a $m \times n$ with a $n \times p$ matrix. What is the runtime (in asymptotic notation) ? (3P)

2. Sorting

Given the array A and this quicksort algorithm.
A= (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1)

QUICKSORT (A, p, r)

```
if p < r then
    q ← MIDDLEPARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
end if
```

MIDDLEPARTITION (A, p, r)

```
i ← ⌊(p+r)/2⌋
exchange A[r] ↔ A[i]
x ← A[r]
i ← p − 1
for j ← p to r − 1 do
    if A[j] ≤ x then
        i ← i + 1
        exchange A[i] ↔ A[j]
    end if
end for
exchange A[i + 1] ↔ A[r]
return i + 1
```

a) Sort the array using this quicksort algorithm. Show the state of the array for each recursion level of QUICKSORT , starting with the deepest recursion level and walking your way up to the final sorted array. (5P)

b) Give an array of 5 numbers that reproduces the worst-case runtime of this quicksort algorithm.(5P)

3. Heaps

For reference, you will find (MAX) HEAPSORT, MAXHEAPIFY and BUILD-MAXHEAP on the next page.

Given the array $A = (3, 6, 9, 1, 8, 5, 11)$

a) Is this a min heap? Give all violations of the heap property.(4P)

b) Write the function MinHeapify and BuildMinHeap. (4P)

c) What is the content of A after the execution of BUILDMINHEAP(A) ? On what elements MinHeapify is called? Which elements are compared or exchanged? Give the sequence of compares and exchanges. (6P)

d) Write MinHeapsort using the MinHeap. Run it on the array, show the contents of the array after each iteration of the FOR loop. (6P)

MAXHEAPIFY (A,i)

```
l ← LEFT[i]
r ← RIGHT[i]
if l ≤ heap-size[A] and A[l] > A[i] then
    largest ← l
else
    largest ← i
end if
if r ≤ heap-size[A] and A[r] > A[largest] then
    largest ← r
end if
if largest ≠ i then
    exchange A[i] ↔ A[largest]
    MAXHEAPIFY(A, largest)
end if
```

BUILDMAXHEAP (A,i)

```
HEAP-SIZE[A] ← length[A]
for i ← ⌊length[A]/2⌋ downto 1 do
    MAXHEAPIFY(A, i)
end for
```

MAXHEAPSORT (A)

```
BUILDMAXHEAP(A)
for i ← length[A] downto 2 do
    exchange A[1] ↔ A[i]
    HEAP-SIZE[A] ← HEAP-SIZE[A] − 1
    MAXHEAPIFY(A, 1)
end for
```

4. Hashing

Given an Array of integers of length 16 i.e. with array indices 0 to 15 and a hash function $h(k, i) = ((k \mod 23) + i) \mod 16$.

Insert the following sequence of numbers into the array, show the contents of the array after each step. Use hashing with open adressing.

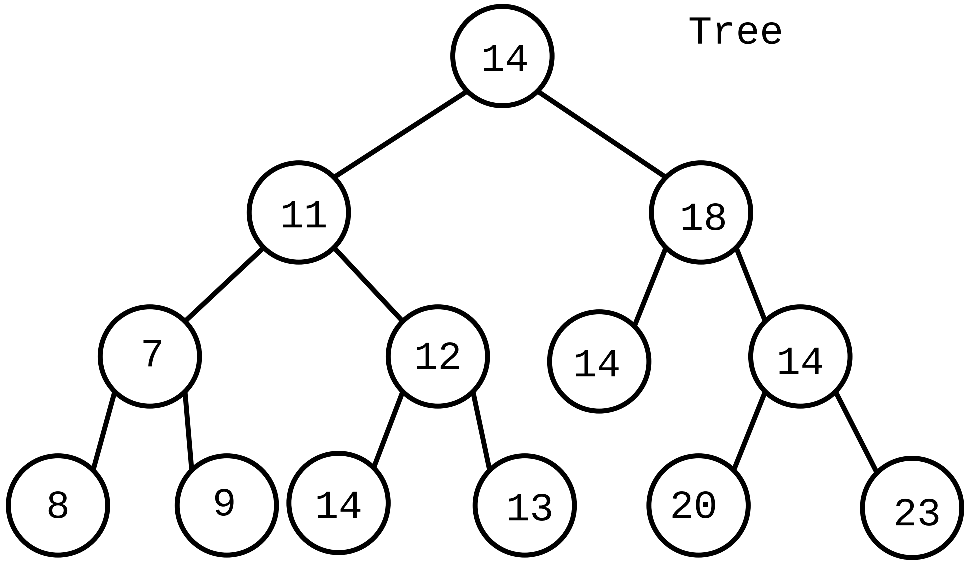
272 273 325 498 15 137 1033

(10P)

5. Binary Trees

For reference, you will find TREEINSERT on the next page.

Given the following binary tree:



a) Is it a binary search tree? List all violations of the binary search tree property. (4P)

b) Give a recursive algorithm for insertion into a binary search tree. (6P)

Given an empty binary search tree.

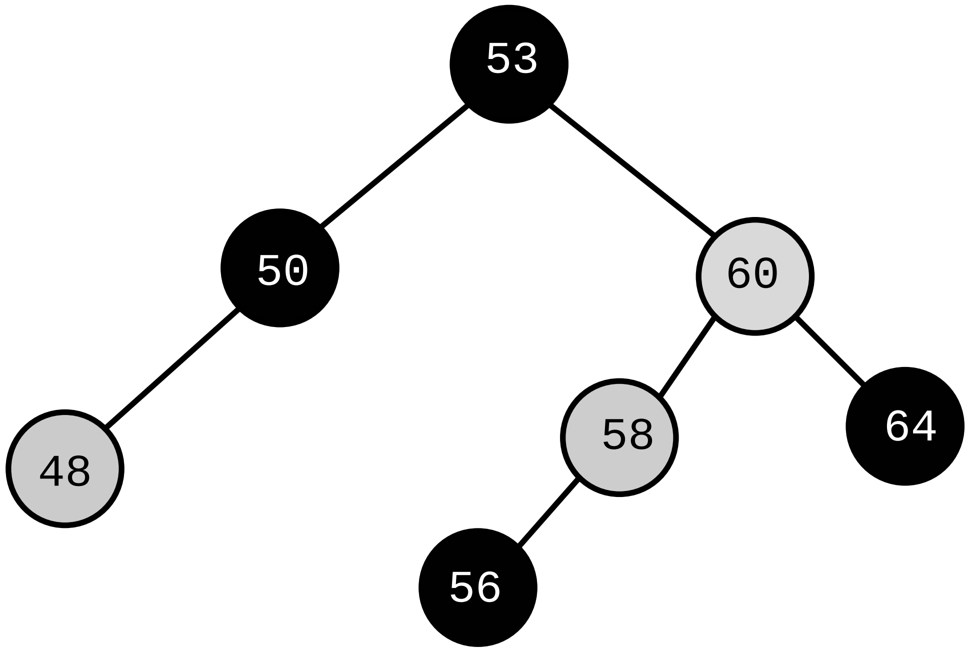
c) Insert the values 12, 15, 3, 7, 15, 12 into the binary search tree. Draw the tree after each insertion. (5P)

TREEINSERT(T, z)

```
y ← NIL
x ← root[T]
while x ≠ NIL do
    y ← x
    if key[z] < key[x] then
        x ← left[x]
    else
        x ← right[x]
    end if
end while
p[z] ← y
if y = NIL then
    root[T] ← z
else
    if key[z] < key[y] then
        left[y] ← z
    else
        right[y] ← z
    end if
end if
```

6. Red-Black-Trees

Given the following tree:

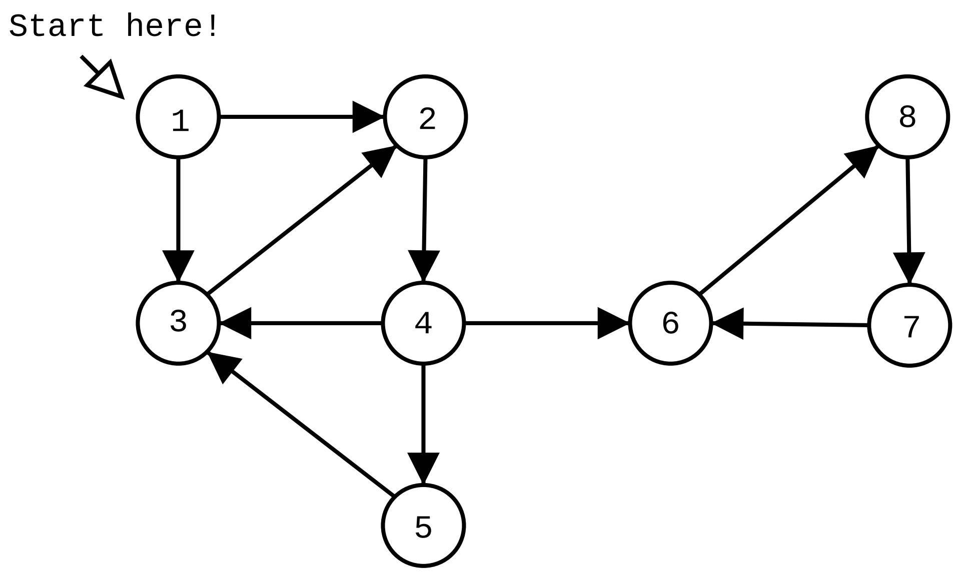


a) draw the tree after a RotateRight operation on node 60 (4P)

b) The resulting tree violates a red-black-property. Which one? What is necessary to remove the violation? (Hint: one rotation and two color changes) (6P)

7. Graphs

Given this directed graph.

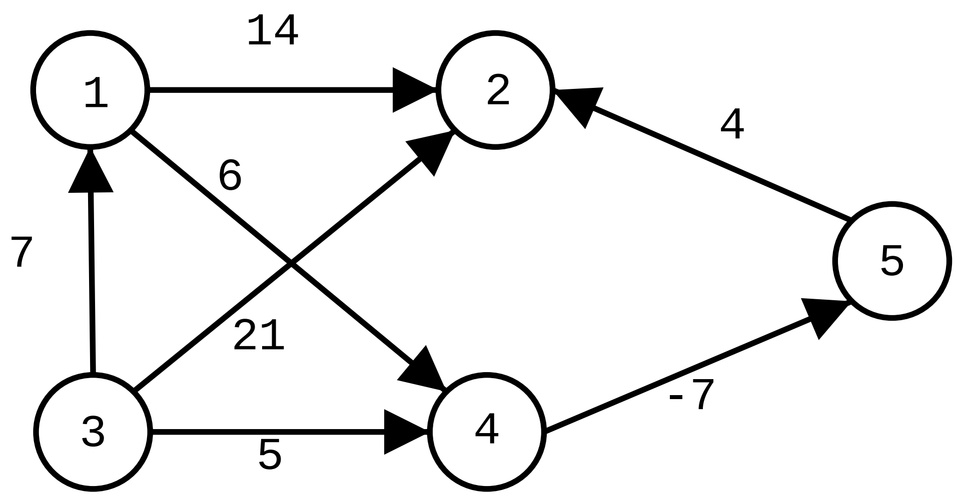


a) Give an algorithm that counts and lists all back edges in the graph. (7P)

b) Demonstrate your algorithm on the given graph. Start with node 1 and analyzes outgoing edges in clockwise order starting from the incoming edge used by the algorithm. (8P)

8. Paths

Given the following graph:



a) Give the adjacency (weight) matrix representation of the graph. (5P)

b) Determine all-pairs shortes path by demonstrating the Floyd-Warshall-Algorithm on this graph. Give all the intermediate $D^{(i)}$ matrices. Draw a complete graph with the shortest distances as edge weights. (10P)