Spring 2015, Test 1, Introduction to Algorithms

Name:
Section:
Email id:

3rd March, 2015

Answer all Six questions. You have 90 minutes to complete the exam.
You can use a calculator - You are allowed to use your text books and notes.

Sample Solution

1. **Recursion and Design and Analysis**

(a) You are given as input three **unsorted** arrays A, B, C of size n. Each of the array elements is an integer. Write an efficient algorithm (pseudo code) to test whether every element of A is larger than every element of B and every element of C is larger than every element of A. [output is either YES or NO]. What is the running time of your algorithm in O notation. [**5 points**] (A linear time algorithm gives you full credit. If it takes more than linear time, you will get partial credit.)

1. Find min and max of array A
2. Find max of array B
3. Find min of arrray C.
4. if (min of C > max of A) and (min of A > max of B)

return YES
else return NO

1, 2, 3 takes O(n) time
4 takes O(1) time
Overall complexity O(n)

(b) The following partial pseudo code computes 3^{3^n}, for integer $n \geq 0$. Complete the pseudo code (one loop as given is sufficient). [**5 points**]

```
Input n ( assumed to be an integer greater than or equal to 0)
Output: x

x = 3
i = 1
while (i <= n) {

x = // complete this statement
i = i+1
}

return x
```

x * x * x

idea: $3^{3^n} = 3^{3^{n-1} * 3^{n-1} * 3^{n-1}} = 3^{3^{n-1} * 3} = 3^{3^n}$

2. **Recurrence Relations, Master Theorem**

(a) Solve the recurrence equation. Write an explicit expression for $T(n)$. Assume n is a power of 2 ($n = 2^k$), and $T(1) = 0$

$$T(n) = T(n/2) + \log_2(n)$$

[**5 Points**]

$T(n) = T(\frac{n}{2}) + \log_2 n$

$n = 2^k$

$T(2^k) = T(2^{k-1}) + k$

Let $Q(k) = T(2^k)$

$Q(k) = Q(k-1) + k$

$Q(0) = T(2^0) = T(1) = 0$

$Q(k-1) = Q(k-2) + k-1$

$\vdots$

$Q(1) = Q(0) + 1$

$Q(0) = 0$

$Q(k) = \frac{k(k+1)}{2}$

$T(n) = O\left((\log_2 n)^2\right)$

(b) How many print statements (cout) will be execute by the following program when n = 2, n = 3 and n = 6. Write a recurrence relation and solve for the number of times print statements are executed for a general n (again assume n is a power of 2). [**5 Points**]

```
void csci2300(int n)
{
  if (n>1)
  {
    cout << ''Scientia et Instantia'' << endl;
    csci2300(n/2);
    csci2300(n/2);
    csic2300(n/2);
  }
return;
}
```

$n = 2$ 1

$n = 3$ 1

$n = 6$ $1 + 3*1 = 4$

$T(n) = 3T(\frac{n}{2}) + 1$

using masters theorem

$T(n) = O(n^{\log_2^3})$

$Q(k) = 3Q(k-1) + 1$

$Q(k-1) = 3Q(k-2) + 1$

$Q(k) = 3^2 Q(k-2) + 3 + 1$

$\vdots$

$Q(k) = 3^k Q(k-k) + 3^{k-1} + \cdots + 1$

$= \frac{3^k - 1}{2} = \frac{3^{\log_2^n} - 1}{2}$

3. **Divide and Conquer and Design of Algorithm**

Given an array, A, of **sorted even integers(not necessarily consecutive)** of size n. Describe an efficient algorithm to test whether there exists an index i such that A[i] = 2i. Array A could be [-12,2,4,6,10]
Output will be YES as there is an index 5 and A[5]=2*5 = 10
Input A[1],A[2],..., A[n]
Output YES if there is an i such that A[i] =2i , NO otherwise

(a) Provide a pseudo code description of an algorithm. [**5 points**] (Efficiency is important - linear time will get you half the points)

(b) What is the complexity of your algorithm. [**5 points**]

Idea: if $A[i] > 2i$ then every $j > i$, $A[j] > 2j$
if $A[i] < 2i$ then every $j < i$, $A[j] < 2j$

```
l = 1; h = n;
While h ≥ l do
  {
    mid = ⌊(h+l)/2⌋
      if A[mid] == 2*mid return yes
      if A[mid] < 2*mid then l = mid+1
      if A[mid] > 2*mid then h = mid-1
  }
return NO
```

Complexity is $O(\log_2 n)$
(binary search)

4. **Algorithms with Numbers**

(a) Using Extended Euclidean Algorithm, compute the (multiplicative) inverse of 17 mod 83. [**6 points**]

a	b	quotient	remainder
83	17	4	15
17	15	1	2
15	2	7	1
2	1	2	0
1	0		

$1 = 1 - 0$

$= 1 - (2 - 2*1)$

$= 3*1 - 2$

$= 3*(15 - 7*2) - 2$

$= 3*15 - 22*2$

$= 3*15 - 22*(17 - 1*15)$

$= 25*15 - 22*17 = 25*(83 - 4*17) - 22*17$

$= 25*83 - 122*17$

inverse is $= -122 = -39 = 44$

(b) what is the value of $7^{3^{2015}}$ *mod* 10 (Hint: 10 is a product of two prime numbers 2 and 5). You have to show your work (just writing the answer gives 2 points) and what theorems/lemmas you used. [4 **points**]

$7^4 = 1 \bmod 10$ (2×5) by Euler's Theorem $a^{(p-1)(q-1)} = 1 \bmod pq$

$3^{2015} \bmod 4 = -1 \bmod 4$

$7^{-1 \bmod 4} \bmod 10 = 7^3 \bmod 10$

$= 3$

5. **Recursive binary Trees**

A binary tree is a tree that has either 0 or 1 or 2 children at each vertex. Let B_n denote the number of binary trees with n vertices. Naturally B_1=1. and B_2=2.

(a) Draw Binary trees with 3, 4 vertices, determine the exact values of B_3, B_4 [**4 points**].

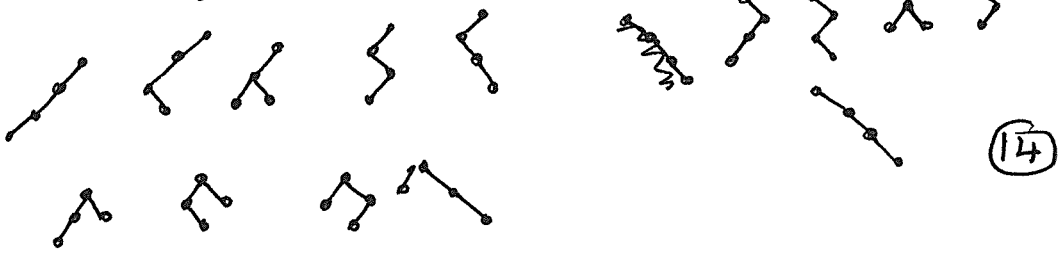
(b) Compute the number of Binary trees with 5 vertices (and **not draw**) - Show your work [**2 points**]

(c) For general n, derive a recurrence relation (and not solve) for B_n. [**4 points**].

B_3

(5)

B_4



(14)

$B_4 = B_3 B_0 + B_0 B_3 + B_2 B_1 + B_1 B_2$

$$B_n = \sum_{i=0}^{n-1} B_i \, B_{n-1-i}$$

$B_5 = B_4 B_0 + B_3 B_1 + B_2 B_2 + B_1 B_3 + B_0 B_4$

$= 14 + 5 + 4 + 5 + 14 = 42$

6. **Graphs and Graph Algorithms**

(a) A complete k-partite graph $K_{n_1,n_2,\cdots,n_k}$ has $N = n_1+n_2+\cdots+n_k$ vertices partitioned into parts of sizes $n_1, n_2, \cdots, n_k$ and it has edges between any two parts that don't belong to the same part. An example of $K_{2,3}$ will be with nodes {1,2,a,b,c} and edges {{1,a},{1,b},{1,c},{2,a},{2,b},{2,c}}. There are 5 nodes and 6 edges in this graph $K_{2,3}$.

- Compute the number of edges in a complete 3-partite graph $K_{3,4,5}$ **[2 points]**
- Compute the number of edges in a complete k-partite graph $K_{n_1,n_2,\cdots,n_k}$ **[4 points]**

$K_{3,4,5}$

# of nodes = 12

# of edges = 12 + 20 + 15 = 47

$K_{n_1,n_2,n_3\cdots n_K}$

# of nodes = $n_1+n_2+\cdots+n_K = N$

# of edges = $\binom{N}{2} - \binom{n_1}{2} - \binom{n_2}{2} \cdots - \binom{n_K}{2}$

# of edges $= \sum_{\substack{i=1 \\ j>i}}^{k-1} n_i n_j$

$= n_1n_2+\cdots n_1n_K$

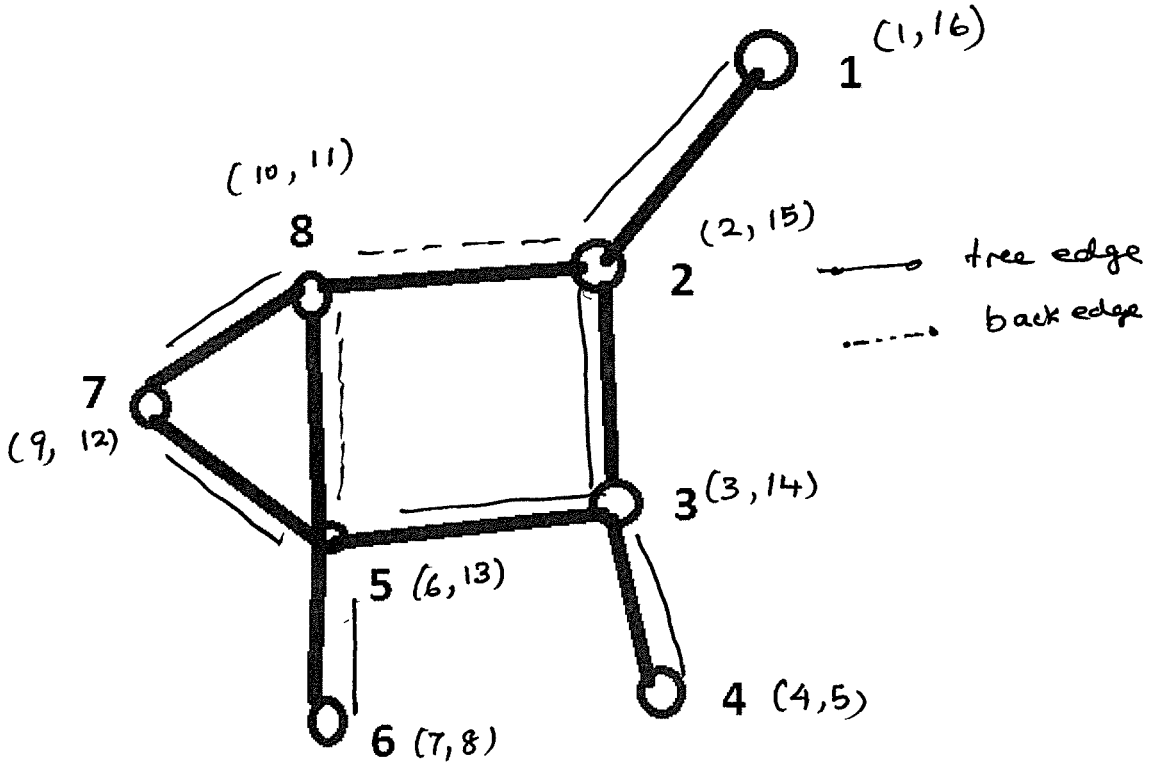
$+ n_2n_3+\cdots+n_2n_K$

$+ n_3n_4+\cdots n_3n_K$

$\vdots$

$n_{K-1}n_K$

(b) For the following graph starting at node 1, compute the pre and post numbers while doing the DFS on the graph. Initially all the nodes are unvisited. From any given visit the adjacent nodes in increasing order of the labels. [**4 points**]



i (Pre number, post number)

Vertex number