Gebze Institute of Technology
Department of Computer Engineering
CSE 321 Introduction to Algorithm Design
Fall 2014
Midterm Exam
November 13th 2014

Student ID and Name	Q1 (20)	Q2 (20)	Q3 (20)	Q4 (20)	Q5 (20)	Total

Read the instructions below carefully

- All cases of confirmed cheating will be reported for disciplinary action.
- You have 150 minutes.

Q1.

a) Analyze the worst case and average case complexities of binary search **(10 points)**

b) Analyze the average case complexity of insertion sort **(10 points)**

DONE IN CLASS!

Q2. Find the order of growth of the following expressions. Indicate the complexity class the function belongs to. Finally, give an ordering of the complexity classes you have found. Prove your statements. **(20 points)**

a) $f(n) = \lfloor \log_2(n) \rfloor$ $\Rightarrow$ HW#1, Q4 e $\Rightarrow \in \Theta(\log n)$

b) $f(n) = \sum_{i=1}^{n} \log(i^2)$ $\Rightarrow$ HW#1, Q6 b $\Rightarrow \in \Theta(n \log n)$

c) $f(n) = \sum_{i=1}^{n} (i+1)3^{i+1}$ $\Rightarrow$ HW#1, Q6 c $\Rightarrow \in \Theta(n\, 3^n)$

d) $f(n) = 9f(n/3)+n$; $f(1)=1$

e) $f(n) = f(n-2)+(3/n)$; $f(1)=1$

$f(0)=1$

d $\Rightarrow$ Master Theorem:

$T(n) = a\,T\left(\frac{n}{b}\right) + g(n)$

$a=9,\ b=3 \Rightarrow \log_b a = \log_3 9 = 2$

$1 < \log_b a \Rightarrow \log_b a > 1$

If $g(n) = O(n^c)$, $[c=1]$

then $T(n) = \Theta(n^2)$ since $\log_b a = 2$

✓

e

$\Theta(\log n) < \Theta(n \log n) < \Theta(n^2) < \Theta(n\, 3^n)$

$f(0)=1$

$f(1)=1$

$f(2) = f(0) + \frac{3}{2} = 1 + \frac{3}{2}$

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

$f(4) = f(2) + \frac{3}{4} = 1 + \frac{3}{2} + \frac{3}{4}$

$f(5) = f(3) + \frac{3}{5} = 1 + \frac{3}{3} + \frac{3}{5}$

$f(6) = f(4) + \frac{3}{6} = 1 + \frac{3}{2} + \frac{3}{4} + \frac{3}{6}$

$$f(n) = \begin{cases} 1 + 3\cdot\left(\frac{1}{2}+\frac{1}{4}+\dots+\frac{1}{n}\right) & \text{if } n \text{ is even} \\ 1 + 3\cdot\left(\frac{1}{3}+\frac{1}{5}+\dots+\frac{1}{n}\right) & \text{if } n \text{ is odd} \end{cases}$$

Recall that Harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ $\Rightarrow$ $1 + 3\cdot\left(\frac{1}{2}+\frac{1}{4}+\dots+\frac{1}{n}\right) = 1 + \frac{3}{2}\cdot\left(1+\frac{1}{2}+\frac{1}{3}+\dots+\frac{1}{n}\right)$

$= 1 + \frac{3}{2}\cdot H_{n/2}$

$\Rightarrow \in \Theta(\log n)$

if n is even

n is odd:

$1 + \frac{1}{3} + \frac{1}{5} + \dots + \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$

$- \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \dots + \frac{1}{n-1}\right)$

$= H_n - \frac{1}{2}\cdot\left(1 + \frac{1}{2} + \dots \frac{1}{\frac{n-1}{2}}\right) = H_n - \frac{1}{2}\cdot H_{\left(\frac{n-1}{2}\right)} \in \Theta(\log n)$ if n is odd

Q3.

a) Consider the following problem called *Subset Sum Problem:*

> *"Given a set of integers and an integer z, is there a non-empty subset whose sum is z?"*

For instance, given the set {6,-4, 3, 5,-2} and z=0, the answer is yes because the subset {6,-4,-2} has sum that equals 0.

Which of the problems we have seen in class is the Subset Sum Problem related to? How? What is the difference between them?

Propose a polynomial-time heuristic algorithm for this problem. ~~Analyze the complexity of your algorithm.~~ Justify that your algorithm runs in polynomial time

b) Consider the following problem called *Change Making Problem:*

> *"How can a given amount of money be made with the least number of coins given denominations?"*

Which of the problems we have seen in class is the ~~Subset Sum~~ Change Making Problem related to? How? What is the difference between them?

Propose a polynomial-time exact algorithm for the Turkish coin system. Analyze the complexity of your algorithm. Would your algorithm work in the old British coin system which includes coins such as 1, 3, 6, 12, 24, 30? Why or why not?

(a) Subset Sum Problem is the special case of knapsack problem where profits and weights are identical for each item.

Hence, the problem boils down to determine whether the upper limit for total weight can be strictly satisfied.

↳(which equals z in this case)

the items are the numbers & selecting the items to be put into the knapsack corresponds to selecting the non-empty subset of numbers

A possible heuristic algorithm:

```
Algorithm HeuristicSubsetSum(L[1..n], z)
    S ← L[1]
    for i = 1 to N do
        T ← $x_i + \bigcup_{y \in S} y$
        U ← T ∪ S
        Sort U
```

→ continue on Q5

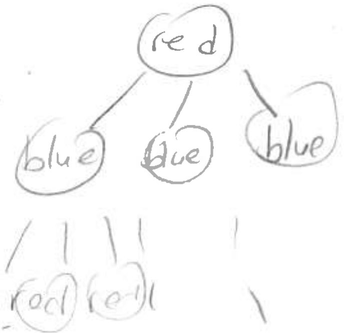
Q4. Design a BFS-based algorithm to test whether a given graph is bipartite. Analyze the complexity of your algorithm. **(20 points)**

→ If the graph is not connected, handle each component separately.

→ Pick any node $s \in V$ and color it red

→ Color all the neighbors of s with blue

⇒ All neighbors of these nodes must be "coloed red; their neighbors blue and so on, until the whole graph is coloed.



→ At the end, either we have a valid red/blue coloring of G, in which every edge has ends of opposite colors or there is some edge with ends of the same color.

If we have a valid red/blue coloring, the graph is bipartite

Otherwise, the graph is not bipartite.

(*) We can implement this algorithm on top of BFS, by simply taking the implementation of BFS and adding an extra array Color over the nodes.

Whenever we get to a step in BFS where we are adding a node v to a list $L[i+1]$, we assign

Color[v] = red if $i+1$ is an even number

Color[v] = blue if $i+1$ is an odd number

(*) At the end, simply scan all edges and determine whether ∃ any edge for which both ends received the same color.

Hence, As in BFS, total running time is $O(|V|+|E|)$.

Q5. Design an exact algorithm for the following task. For any even n, mark n cells on an infinite sheet of graph paper so that each marked cell has an odd number of marked neighbors. Two cells are considered neighbors if they are next to each other either horizontally or vertically but not diagonally. The marked cells must form a contiguous region, i.e., a region in which there is a path between any pair of marked cells that goes through a sequence of marked neighbors.
Prove that it is impossible to mark an odd number of cells, each with an odd number of marked neighbors. **(20 points)**

Q3 in Exercise 4.1 in your book!

SOLUTION IS ON THE INTERNET!

cont'd $\boxed{Q3}$

```
S ← ∅
y ← Smallest element of U
y ← y ∪ S
for each z ∈ U in increasing order do
        if y + cs/N < z < s
            z ← y
            S ← S ∪ z
end for
If S has a number between (1-c)s and s, output yes, o.w. output no
```

The lists S, T and U have size polynomial in N and 1/c
& all operation on them can be done in poly-time.
The size of the list is kept polynomial
since we only include a number z into S
if it is greater than the previous one by cs/N
& not greater than s.

Change Making Problem

It is a special case of the knapsack problem where each item has profit $p_j = -1$ and strict equality is imposed in the constraint about upper limit for total weights.

Poly-time Exact Algo for Turkish Coin System: Pick the largest denomination of coin which is not greater than the remaining amount.

135 TL ⇒ $\underbrace{100 + 20 + 10 + 5}_{4\ \text{coins}}$

In coins with
1, 3, 6, 12, 24, 30:

To have 48,
the above algorithm gives: $\overbrace{30 + 12 + 6}^{3\ \text{coins}}$
However, the optimum solution is: $\overbrace{24 + 24}^{2\ \text{coin}}$

Complexity: In each iteration, finding the largest one ≤ remaining [illegible] polynomial