Midterm Exam II of CPS 230, Fall 2002

11/13/2002

- This midterm is closed-book. There are five problems. Total number of points is 100.
- Do all your own work. You may refer to algorithms or results that we covered in class.
- Please be sure to write down your name on every answer sheet you will hand in at the end of the exam.
- Do not spend too much time on any problem. Show your work, as partial credit will be given. Please be neat.
- Think clearly, stay calm. Good Luck!

Problem	Points	Score
1	20	
2	20	
3	15	
4	15	
5	30	
Total	100	

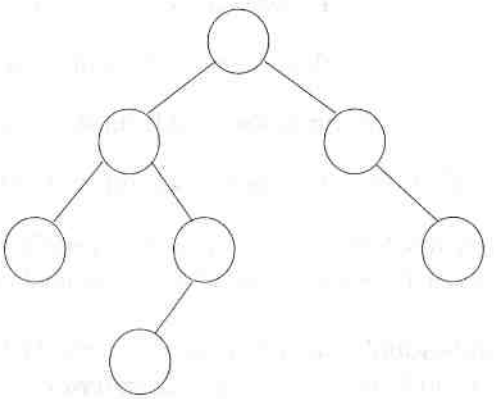
I certify that all work on this exam is solely the product of my own work.

YOUR NAME (print):_____

Signature:_____

1 Red-Black Trees [20 points]

a. [6 points] Label the following binary tree with numbers from the set $\{6, 22, 9, 14, 13, 1, 8\}$ so that it is a legal binary search tree.



b. [6 points] Label each node in the figure above with r or b denoting the colors RED and BLACK, respectively, so that the tree is a legal red-black tree.

c. [8 points] Make the left child of the root be the root by performing a single rotation. Draw the binary search tree that results, and label your tree with the keys from part(a). Is it possible to label the nodes with colors so that the tree is a red-black tree? Justify your answer.

2 Amortized Analysis [20 points]

In this problem we consider two stacks A and B manipulated using the following operations (n denotes the size of A and m the size of B):

- $PushA(x)$: Push element x on stack A.
- $PushB(x)$: Push element x on stack B.
- $MultiPopA(k)$: Pop $\min\{k, n\}$ elements from A.
- $MultiPopB(k)$: Pop $\min\{k, m\}$ elements from B.
- $Transfer(k)$: Repeatedly pop an element from A and push it on B, until either k elements have been moved or A is empty.

Assume that A and B are implemented using double-linked lists such that $PushA$ and $PushB$, as well as a single pop from A or B, can be performed in $O(1)$ time worst-case.

a. **[6 points]** What is the worst-case running time of the operations $MultiPopA$, $MultiPopB$ and $Transfer$?

b. **[14 points]** Define a potential function $\Phi(n, m)$ and use it to prove that the operations have amortized running time $O(1)$.

3 Dynamic Programming [15 points]

A *palindrome* is a string that reads the same from front and back. Any string can be viewed as a sequence of palindromes if we allow a palindrome to consist of one letter.

Example: "bobseesanna" can e.g. be viewed as being made up of palindromes in the following ways: "bobseesanna"="bob"+"sees"+"anna" "bobseesanna"="bob"+"s"+"ee"+"s"+"anna" "bobseesanna"="b"+"o"+"b"+"sees"+"a"+"n"+"n"+"a"

We are interested in computing MinPal(s) defined as the minimum number of palindromes from which one can construct s (that is, the minimum k such that s can be written as $w_1 w_2 \ldots w_k$ where $w_1, w_2, \ldots, w_k$ are all palindromes).

Example: MinPal("bobseesanna")=3 since "bobseesanna"="bob"+"sees"+"anna" and we cannot write "bobseesanna" with less than 4 palindromes.

It is easy to see that

$$MinPal(s[i,j]) = min_{i \leq k < j}\{s[i,k] + MinPal(s[k+1,j])\}$$

a. [5 points] Show that the straightforward implementation of MinPal(s) based on the above formula results in an exponential time algorithm.

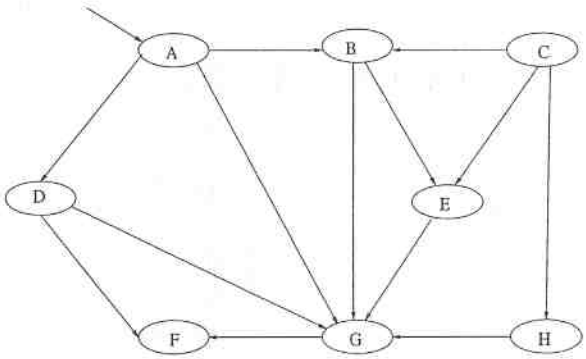
b. [15 points] Describe and analyze a better algorithm.

4 Greedy Algorithms [15 points]

Suppose you are given two sets A and B, each containing n positive integers. You can choose to reorder each set however you like. After reordering, let a_i be the ith element of set A, and let b_i be the ith element of set B. You then receive a payoff of $\Pi_{i=1}^{n} a_i^{b_i}$. Give an algorithm that will maximize your payoff. Prove that your algorithm maximizes the payoff, and state its running time.

5 Graph [30 points]

Given a DAG as follows:



a. **[4 points]** Show the ordering of vertices by TOPOLOGICAL-SORT when it is run on the above DAG. Assume we start from node A first.

b. [8 points] If we add a directed edge from node F to node A. Is this graph is still a DAG? And please perform DFS again from node A and mark every edge as Tree edge (T), Back edge(B), Forward edge(F) or Cross edge(C).

Edge	Type(T/B/F/C)
AB	
BG	
GF	
FA	
AD	
DF	
DG	
CB	
BE	
EG	
CE	
CH	
HG	

c. [6 points] Find the strongly connected components of the graph after FA is added.

d. **[12 points]** Another way to perform topological sorting on a directed acyclic graph G=(V,E) is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph from the graph. Explain how to implement this idea so that it runs in time O(V+E) and prove the correctness. What happens to this algorithms if G has cycles?