CPS 230 Final Exam

Fall 1999

2-5pm, Monday December 13th
Closed book exam

Please make sure to write your solutions in a clear and precise way!

30 points **Problem 1**

- Show that $\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$
- Solve the recurrence
  $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1) + n(n+1)(n-1) & \text{if } n \geq 2 \end{cases}$

30 points **Problem 2**

Prove that $\Omega(n \lg n)$ is a lower bound—in the (algebraic) decision tree model—for the following problem:

Given reals $x_1, x_2, ..., x_n$, decide if the following holds for all $i \neq j$: $(|j-i| \bmod 2 = 0) \Rightarrow (x_i \neq x_j)$.

(Hint: Use reduction from one of the problems for which we have already proved a $\Omega(n \lg n)$ lower bound—Sorting, Element Distinctness, $A \cap B = \emptyset$ problem, $A = B$ problem, $A \subset B$ problem.)

35 points **Problem 3**

In this problem we consider a database of measurements of the heights of a group of people. Each entry in the database is an element of the form

(social security number, age, height)

which registers the height of a person at a certain age. The database can have several measurements for a given person but only one measurement for a given person at a given age. We do *not* assume that the height of a person is monotonically increasing with age, but it can be assumed that the age of a person is an integer in the interval [0..150].

We want to support the following operations:

- *Init*: Create an empty database.
- *Insert*(ssn, age, height): Insert a new or update an existing measurement for a given person.
- *Delete*(ssn): Delete *all* information for a given person.
- *CurrentHeight*(ssn): Return the latest measurement for a given person.
- *Tallest*: Return the tallest height currently recorded in the database.
- *Population*(age): Return the number of measurements (entries) taken at a certain age currently recorded in the database.

Describe a data structure such that *Init*, *Tallest*, and *Population* runs in $O(1)$ time, and such that *Insert*, *Delete*, and *CurrentHeight* runs in $O(\log n)$ time, where n is the number of people in the database.

30 points **Problem 4**

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.

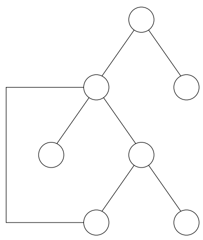
Assume that A and B are implemented using double-linked lists such that *PushA* and *PushB*, as well as a single pop from A or B, can be performed in $O(1)$ time worst-case.

a) What is the worst-case running time of the operations *MultiPopA*, *MultiPopB* and *Transfer*?

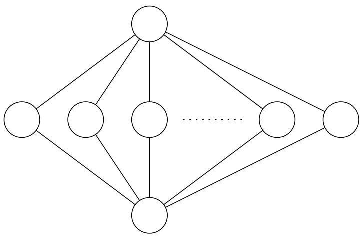
b) Define a potential function $\Phi(n, m)$ and use it to prove that the operations have amortized running time $O(1)$.

40 points **Problem 5**

Consider an undirected weighted graph which is formed by taking a binary tree and adding an edge from exactly one of the leaves to another node in the tree. We call such a graph a *loop-tree*. An example of a loop-tree could be the following:



Consider also a so-called *pole-graph* which is an undirected weighted graph consisting of two poles connected through a layer of nodes as follows:



Let n be the number of vertices in a loop- or pole-graph and assume that the graphs are given in the normal edge-list representation without any extra information. In particular, the representation does not contain information about which vertex is the root (in the case of a loop-graph) or which vertices are the poles (in the case of a pole-graph).

a) How long time would it take Prim's or Kruskal's algorithms to find the minimal spanning tree of a loop- or pole-graph?

b) Describe and analyze a more efficient algorithm for finding the minimal spanning tree of a loop-graph. Remember to prove that the algorithm is correct.

c) Describe and analyze a more efficient algorithm for finding the minimal spanning tree of a pole-graph. Remember to prove that the algorithm is correct.

35 points **Problem 6**

Consider the numbers $(A_n)_{n>0} = (1, 1, 3, 4, 8, 11, 21, 29, 55, \ldots)$ defined as follows:

$$\begin{aligned} A_1 &= A_2 = 1 & \\ A_n &= B_{n-1} + A_{n-2} & n > 2 \\ B_1 &= B_2 = 2 & \\ B_n &= A_{n-1} + B_{n-2} & n > 2 \end{aligned}$$

A_n can be computed using the following recursive procedures:

```
ComputeA(n)
   if n<3 then
      return 1
   else
      return ComputeB(n-1)+ComputeA(n-2)
   fi
end

ComputeB(n)
   if n<3 then
      return 2
   else
      return ComputeA(n-1)+ComputeB(n-2)
   fi
end
```

a) Show that the running time of ComputeA(n) is exponential in n.

b) Describe and analyze a more efficient algorithm for computing A_n.