Second Midterm Exam, CMPSC 465, Spring 2009

- This is open book exam, you can also have notes (several sheets or one notebook). The use of communication equipment and calculators is not permitted the exam.

- When you describe an algorithm, you should also justify its correctness and your estimate of the running time.

- If you use the extra space on page 4 and 8, indicate what problem and subproblem it is.

1. Treaps, 5 pt.

   Draw a treap for the following set of keys with priorities (the largest priority should be at the root):

<table>
<thead>
<tr>
<th>key</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>20</td>
<td>6</td>
<td>2</td>
<td>15</td>
<td>4</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>

![Treap Diagram]

1
2. B-trees, 10 pt.

a. Draw a 2-3 tree that results from the tree below if we delete I.

```
M
\_ C
 |   |   \\ G
 C   \_   \_ F
  |     |     |   |
  | I   | N   | O   |
```

b. Draw a 2-3 tree that results from the tree below if we insert Q.

```
M
\_ C
 |   |   \\ G
 C   \_   \_ F
  |     |     |   |
  | I   | N   | O   |
```

```
M
\_ C
 |   |   \\ G
 C   \_   \_ F
  |     |     |   |
  | I   | N   | Q   |
```

a. Draw a heap that occupies positions 1 to 11 in array A

<table>
<thead>
<tr>
<th>position in A[n]</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>30</td>
<td>15</td>
<td>10</td>
<td>7</td>
<td>13</td>
<td>1</td>
<td>9</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>12</td>
</tr>
</tbody>
</table>

b. Draw a heap that results from applying DeleteMax to the heap you have drawn in a.

c. Draw a heap resulting from applying Insert(H, 14) to the heap drawn in b.

Heaps, cont.

d. How many levels do we have in a heap with 100 keys? Show your reasoning/computation.

The root level of a binary heap has one node, and each subsequent full level has twice as much nodes, so in a heap with 100 nodes we have 6 full levels with 1, 2, 4, 8, 16, 32 and an incomplete level with 100 - 63 = 37 nodes, hence 7 levels.
4. Problem Reduction, 10 pt.

Given a number array $A[n]$ we can in time $O(n)$ compute array $S[n]$ that allows to answer the following queries: what is the value of $\sum_{i=a}^{b} A[i]$? (We can have $S[-1] = 0$, $S[i] = S[i-1] + A[i]$, and the query answer is $S[b] - S[a-1]$.)

Use this to solve the following problem in time $O(n^2)$: find $i, j$ such that $0 \leq i < j < n$ and the following quantity is maximal:

$$\left( \sum_{k=i}^{j} A[k] \right)^2 - \sum_{k=i}^{j} A[k]^2$$

To achieve running time $O(n^2)$ it suffices to evaluate the above expression for every possible pair of $i$ and $j$, $0 \leq i \leq j \leq n$ and record when we get the largest value. Because there are $n^2$ such pairs, the only problem is to make each evaluation in time $O(1)$.

The idea that we can use is to pre-compute partial sums

- $S[0] = 0$
- for ($i = 0$; $i < n$; $i++$)
  - $S[i+1] = S[i] + A[i]$  

Then we get $\sum_{k=i}^{j} A[i]$ as $S[j+1] - S[i]$.

We also have sum of squares. Again, we pre-compute partial sums

- $SQ[0] = 0$
- for ($i = 0$; $i < n$; $i++$)

Now the expression can be evaluated as follows

- $expr = S[j+1] \cdot S[i]$
- $expr *= expr$
- $expr += SQ[j+1] \cdot SQ[i]$
5. Radix sort, 10 pt.

Sort given sequence of words alphabetically using Radix sort: ACB, B, CCC, ACA, C, AB, CA, BC, CAC, BCA, CB

Show the sequence that results from each round. If your method does not check all words in each round, indicate which words were checked in a round.

*We sort first by length: B, C, AB, CA, BC, CB, ACB, CCC, ACA, CAC, BCA.*

*Next, we list sequences of length 2, followed by those of length 3 that have A/B/C on the third position:*


*Next, we list sequences of length 1, followed by those of length 2/3 that have A/B/C on the second position (from the last list):*


*Lastly, we form and joint lists according to the first position:*


Draw the array of chain headers and the chains that form a hash table with hash function \( h(x) = x \mod 11 \) and which stores the following keys: 3, 5, 9, 12, 14, 17, 21, 33, 42, 50, 55, 62.

*I guess you know how to do it, but here is the picture:*
You are given cost matrix $C[n][n]$ for a directed graph, so $C[i][j]$ is the cost of edge $(i, j)$. Assume that $M$ is the maximum entry in the matrix, which means that there is an edge of cost $M$ and each edge has cost at most $M$.

Your task is to compute for each pair of nodes $i, j$ the minimal $x$ such that there exists a path from $i$ to $j$ with all edges having cost $x$ or smaller.

Solve this problem in a manner similar to Warshall algorithm for the transitive closure and Floyd algorithm for All-Pairs-Shortest-Paths.

a. Formulate a recursive subproblem. What are the basic cases?

b. Write the recurrence for non-basic cases.

c. Write pseudocode

d. What are the running time and memory requirements in $O()$ notation?

**a.** $R_{ij}^k$ is the least $x$ such that there is a path from $i$ to $j$ with all intermediate nodes in $\{0, \ldots, k-1\}$ and such that every edge has cost $x$ or less. The basic case is $R_{ij}^0 = C[i][j]$ (no intermediate nodes allowed, hence the best path is the direct edge).

**b.**

\[
R_{ij}^{k+1} = \min(R_{ij}^k, \max(R_{ik}^k, R_{kj}^k))
\]

(the first term in $\min$ is the best path that does not use $k$, the second, the best path through $k$ where the largest cost edge can be before $k$ or after).

c. The pseudocode can follow the lines of Warshall and Floyd:

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        R[i][j] = C[i][j]

for (k = 0; k < n; k++)
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            a = R[i][k]
            b = R[k][j]
            if (a < b)
                a = b
            if (a < R[i][j])
                R[i][j] = a
```

c. The running time is $O(n^3)$ and the memory requirements are $O(n^2)$. 