CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6651/4: Algorithm Design Techniques - Winter 2010

Quiz # 2 - October 1st, 2010

First Name	**Last Name**	**ID#**

Question 1 5 points

Perform the complexity of the Select assuming that, instead of a division into $\lceil n/5 \rceil$ groups of 5 elements, we decide to divide the n elements into $\lceil n/3 \rceil$ group of 3 elements.

Algorithm Select

Step 1. Divide n elements into $\lceil n/3 \rceil$ group of 3 elements. Note that one group may have less than 3 elements.

Step 2. Find the median of each group by first insertion sorting the elements of each group, and then picking the median from the sorted list of group elements, and

Step 3. Use Select recursively to find the median x of the $\lceil n/3 \rceil$ medians found in Step 2.

Step 4. Partition the input array around the median-of-medians x using the modified version of Partition. Let k be one more than the number of elements on the low side of the partition, so that x is the kth smallest element and there are $n - k$ elements on the high side of the partition.

Step 5. If $i = k$, then return x. Otherwise, use Select recursively to find the ith smallest element on the low side if $i < k$, or the $(i - k)$th smallest elements on the high side if $i > k$.

What is the conclusion of your complexity analysis?

Question 2

Give the mathematical definition of the ith order statistics. **.5 point**

What is the best worst case complexity (i.e., $\Theta()$ order of growth) of the best algorithm for computing the ith order statistics? **.5 point**

If we compute the median instead of the ith order statistics, does the complexity changes? If yes, provide the worst case complexity for computing the median of a set of integer values. **.5 point**

Question 3. Optimal Caching Problem 3.5 points

We recall the caching problem below.

- A set U of n pieces of data stored in **main memory**
- A fast memory, the **cache** can hold $k < n$ pieces of data at one time
- Assumption: The cache initially holds some set of k items
- A sequence of data items $D = d_1, d_2, \cdots, d_m$ drawn from U is presented to us
  $\hookrightarrow$ the sequence of memory references we must process
- When processing them, we must decide at all times, which k items to keep in the memory
- **Eviction schedule**: When item d_i is presented, we can access it very quickly if it is already in the cache; otherwise, we are required to bring it from main memory into the cache and, if the cache is full, to evict some other piece of data that is currently in the cache to make room for d_i.
  $\hookrightarrow$ This is called **a cache miss**, and we want to have as few as possible.

Recall the principle of the Farthest-in-Future Algorithm

Recall the definition of a reduced schedule.

Apply the Farthest-in-Future Algorithm on the following example: Sequence $\{a, b, c, d, a, d, e, a, d, b, c\}$, $k = 3$, items $\{a, b, c\}$ in the cache. Provide the content of the cache after each iteration. Does the output solution correspond to a reduced schedule?