COMP 3711 Design and Analysis of Algorithms
Fall 2015 Midterm Exam Solution

Question 1: 1.1 $10^{10^{10}}, \log^9 n, n, n \log n, n^{1.1}/\log n$

1.2 (1) Insertion sort is better if the input is already sorted or almost sorted.

(2) $\Theta(1)$ extra space is available (Insertion sort uses $\Theta(1)$ working space, quicksort uses expected $\Theta(\log n)$ working space).

(3) The input size is very small, insertion sort is better.

1.3 Show that there exist at least one input such that the algorithm runs in $\Omega(n \log n)$.

1.4 (a) $\Theta(\log n)$, (b) $\Theta(n^2)$, (c) $\Theta(n \log n)$, (d) $\Theta(n)$

Question 2:

- Recursively divide the problem into two equal size subproblems, until the problem size is 1.
- Each subproblem returns the index pair (i, j) of the current subproblem. Base case can be solved trivially.
- For each subproblem, find the max element $p[r_{max}]$ of the right subarray and the min element $p[l_{min}]$ of the left subarray by linear scan. Then, compare it's left subproblem result, right subproblem result and $p[r_{max}] - p[l_{min}]$, and return the corresponding index pair (i, j) that makes max amount of money.
- If the result index pair (i, j) of the original input gives $p(j) - p(i) \leq 0$, then the solution is "no way". Otherwise, the index i, j is the solution.

```
FindMaxMoney(array p, int s, int e)
if s = e then (curr_i, curr_j) = (s, s); // O(1)
else
    m = ⌊(s+e)/2⌋;
    (l_i, l_j) = FindMaxMoney(p, s, m); // T(⌊n/2⌋)
    (r_i, r_j) = FindMaxMoney(p, m+1, e); // T(⌈n/2⌉)
    r_max = index of max_{m+1≤i≤e}{p[i]}; // O(n)
    l_min = index of min_{s≤i≤m}{p[i]}; // O(n)
    (curr_i, curr_j) = indices of max(p[l_j] − p[l_i], p[r_j] − p[r_i], p[r_max] − p[l_min]); // O(1)
end
return [curr_i, curr_j];
```

Call (i,j) = FindMaxMoney(p, 1, n). If $(p(j) - p(i) \leq 0)$ output "no way", else output (i, j).
Running time: $T(1) = 1$, $T(n) = 2T(n/2) + n$. So, $T(n) = O(n \log n)$.

Alternative solution ($O(n)$): Create array $B[1..n-1]$, where $B[i] = A[i+1] - A[i]$ for $1 \leq i \leq n-1$. Run $O(n)$ time MCS algorithm on B to obtain (i, j), then return $(i, j+1)$.

Question 3: Let b_i denotes the number of hats that are better than or equal to the hat of customer i. Let $X_i = 1$ if the i-th customer get back his own hat or a better one, otherwise $X_i = 0$. We have $E(X_i) = Pr(X_i = 1) = \frac{b_i}{n}$.

$$E(X) = E\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} E(X_i) = \sum_{i=1}^{n} \frac{b_i}{n} = \sum_{i=1}^{n} \frac{i}{n} = \frac{1}{n}\sum_{i=1}^{n} i = \frac{1}{n}\frac{n(n+1)}{2} = \frac{n+1}{2}$$

Question 4:

10	9	8	7	6	5	4	3	2	1
9	7	8	3	6	5	4	1	2	
8	7	5	3	6	2	4	1		
7	6	5	3	1	2	4			
6	4	5	3	1	2				
5	4	2	3	1					
4	3	2	1						
3	1	2							
2	1								
1									

Question 5: For each day i, stop at the furthest camping site, i.e. stop at the largest x_j such that x_j minus the start location of day i is at most d.

```
camping_sites = {}; curr_loc = x_0;
for i = 1 to n do
    if x_i − curr_loc > d then
        curr_loc = x_{i−1}; camping_sites.insert(x_{i−1});
    end
end
return camping_sites
```

Running time: One linear scan to the n camping site, each iteration runs in $O(1)$. So, the algorithm runs in $O(n)$.

Correctness: Let X be the solution returned by this greedy algorithm, and let Y be an optimal solution. Consider the first camping site where Y different from X. Suppose the camping site in X is located at x and the one in Y is located at y. By the greedy choice, we must have $x > y$. Now move y to x in Y. The resulting Y must still satisfy the requirement, travel at most d kilometers per day. Repeatedly applying this transformation will convert Y into X. Thus X is also an optimal solution.