Analysis of Algorithms

1. It is possible to build $b_n = (n+1)^{-1}\binom{2n}{n}$ binary trees of n nodes. Which of the following is true and why? Explain.

   (a) $b_n \in \Theta(2^n)$

   (b) $b_n \in \Theta(n^2 \log n)$

   (c) $b_n \in \Omega(4^n/2^n)$

   (d) $b_n \in O(n \log^2 n)$

   (e) none of the above

2. Consider the knapsack problem, i.e., that of placing n objects with profits $p_1, \ldots, p_n$ and weights $w_1, \ldots, w_n$, respectively, in a knapsack of size W. Show that the greedy algorithm (which puts objects into the knapsack according to nonincreasing ratios p_i/w_i) yields an arbitrary bad solution for the 1/0 (whole or nothing) knapsack problem. (That is, given any $\epsilon > 0$, find an example where the ratio of the greedy solution quality to the optimal solution quality is less than ϵ.)

3. A noncomparison sorting algorithm, "Good-Sort," uses the following steps to sort n elements of a list in which each list element is a string of a given fixed length, k, drawn from a given ordered alphabet $s_0 < s_1 < \cdots < s_m$. Thus the jth element of the list is

   $$e_{j1} e_{j2} \cdots e_{jk}$$

   where $e_{jk} \in \{s_o, \ldots, s_m\}$.

   Procedure Good-Sort —

   For $i = k$ down to 1 do

   (a) For $j = 1, \ldots, n$ put the jth element in Bucket B_l if $e_{ji} = s_l$.

   (b) Rearrange the list according to the contents of the buckets in order $B_0, B_1, \ldots, B_m$.

   (c) Apply Step (a) to the new list.

   Example: To sort the list 42, 32, 38, 39, 27 (note that $k = 2$ and $\{s_0, \ldots, s_9\} = \{0, 1, \ldots, 9\}$), the algorithm makes 2 passes.

   Pass 1: (for $k = 2$) gives

   Bucket B_0, B_1, B_3, B_4, B_5, and B_6 are empty

Contents of B_2: 42, 32

Contents of B_7: 27

Contents of B_8: 38

Contents of B_9: 39

Rearranged list: 42, 32, 27, 38, 39.

Pass 2: (for $k = 1$) gives

Buckets B_0, B_1, B_5, B_6, B_7, B_8 and B_9 are empty.

Contents of B_2: 27

Contents of B_3: 32, 38, 39

Contents of B_4: 42

Rearranged list: 27, 32, 38, 39, 42; which is the sorted list.

Answer the following questions regarding Good-Sort:

i. Prove the correctness of the Good-Sort algorithm.

ii. For binary strings, i.e., $s_i \in \{0, 1\}$, design and analyze a recursive Good-Sort algorithm based on partitioning the list into two sublists according to the *most significant* bit.

4. Let T be a given binary-search-tree with n nodes. Let $A(T, n)$ be the average behavior of the tree for a given set of probabilities $p_1, \ldots, p_n$ associated with keys $k_1, \ldots, k_n$ and probabilities $q_0, q_1, \ldots, q_n$ associated with *implicit* external nodes representing an unsuccessful search. For example, q_1 = the probability that $x \in (k_1, k_2)$ is searched in T.

   Suppose

   $$p_i = \tfrac{p}{n} \quad \text{for} \quad i = 1, \ldots, n;\ 0 < p < 1.$$

   $$q_i = \tfrac{1-p}{n+1} \quad \text{for} \quad i = 0, \ldots, n.$$

   Show that

   (a) $A(T, n) = EPL(T)\frac{1+p/n}{n+1} - p$ where $EPL(T)$ represents the external path length of tree T.

   (b) Use the result obtained in (a) to prove that

   $$A(T, n) \geq [(n+1)\log_2(n+1)]\left[\frac{(1+\frac{p}{n})}{n+1}\right] - p \in \Omega(\log n)$$

5. For a weight-connected graph $G = (V, E)$ with possible negative cycles, consider the problem of finding the shortest path from the source node $s \in V$ to node $v \in V$ for all v.

   (a) Explain why we cannot (should not) use Dijkstra's algorithm. What would go wrong if we did use it?

   (b) An outline of an algorithm is presented below:

      i. Find $c = \min\{w(e); e \in E\}$.

      ii. If $c < 0$, then adjust weights of all edges to $w^*(e)$ = new weight of edge e = old weight of edge $e - c = w(e) - c$.

      iii. Apply Dijkstra's algorithm to G with weights w^*.

      iv. Let $p(s, v)$ be the shortest path from s to v, obtained in Step (iii), of length l_v. Obtain the weight of this path in the original graph by adding $l_v \times c$ to the current weight.

   Can this algorithm be used to obtain the shortest paths from s to v for all v?

   (c) Compare the complexities of the Bellman and Ford algorithms with Dijkstra's algorithm. Give reasons and details.

6. (a) Show the contents of an open-address hash table consisting of 23 memory spaces [$T[0]$ to $T[22]$] after the insertion of the keys 2, 11, 55, 23, 53, 9, 61, 46, 1, 3, 25, 19, 17, 34, 10, 38, and 48, where the hash function is

$$h(k) = k \bmod 23$$

   using linear probing to resolve collisions.

   (b) Do the above using double-hashing with

$$h_1(k) = k \bmod 23 \quad \text{and} \quad h_2(k) = (k \bmod 21) + 1.$$

   (c) What precaution, if any, must be used to generate appropriate hash functions? Why?