

CPSC 510 Algorithms Winter 2012

Midterm Exam Practice Problems

1. Provide tight asymptotic bounds using Θ -notation for the given recurrence. Assume that $T(n)$ is constant for sufficiently small n .

a. $T(n) = 2T(n - 2) + n^3$

b. $T(n) = 16T(n/4) + T(16) + n^2$

2. A typist was busy typing a string consisting only of n lowercase letters (no punctuation symbols, numbers, spaces). However, the typist accidentally pressed the CAPS LOCK key once causing the remaining letters in the string to be incorrectly converted to uppercase. Your job is to create an efficient algorithm that finds the index of the first letter in the string that was converted to uppercase. You may assume that there is at least one uppercase letter in the string. You also may assume there is a constant-time function IS-UPPER(c) that determines whether a character c is an uppercase letter or not.

Examples:

input: aaaBBB	returns: 4
input: ABCD	returns: 1
input: abcdA	returns: 5

Your algorithm must be asymptotically optimal in the worst case with respect to n as defined above. Your answer must include a clear description of the algorithm and an analysis of its worst case running time. The worst-case running time bound should be as tight as possible.

3. An ordered graph is a directed graph with vertices v_1, v_2, \dots, v_n with the following properties:
- Each edge (v_i, v_j) goes from a vertex with a lower index to a vertex with a higher index ($i < j$).
 - For each vertex, except v_n , there is at least one edge leaving it.

Develop an efficient dynamic programming algorithm that determines the length of the longest path that begins at v_1 and ends at v_n . The length of a path is equal to the number of edges on the path. Note that the algorithm only needs to return the length of the longest path. It is not necessary to return the path (sequence of vertices).

Your algorithm must be asymptotically optimal to a standard dynamic programming solution. It is NOT necessary for you to analyze (or even state) the running time of the algorithm.

4. At a summer camp, a counselor is arranging a triathlon. A triathlon is a race with three stages in this order: swimming, biking, and running. The swimming pool used for the event is small such that only one swimmer can be in the pool at a time. To address this issue, the triathlon will stagger the start times such that the first swimmer will start at 7:00am, the second swimmer will start once the first swimmer has completed. The third swimmer will start once the second swimmer finishes, and so forth. There are no such restrictions for the biking and running phases of the race – as many participants can be simultaneously biking and running as necessary.

The triathlon is non-competitive – it is important for everyone to finish. The counselor also wants to complete the triathlon as quickly as possible. Each participant i has a projected swimming time s_i , projected biking time b_i , and projected running time r_i ; you can assume the participants have the same times during the actual race. Your job is to devise a greedy algorithm that produces an optimal schedule for the swimming portion of the race for n participants. An optimal schedule is one that minimizes the completion time, the time in which the last participant finishes. For example, consider the two competitors:

Competitor	s_i	b_i	r_i
1	3 (time units)	5	7
2	6	4	1

If competitor 1 went first, he/she would finish 15 (3+5+7) time units after the start. Competitor 2 would have to wait 3 time units (when competitor 1 finished swimming). Then he/she would take 11 additional time units (6+4+1) to complete the triathlon, meaning that competitor 2 would be finished 14 time units after the start of the race. In this case, the last participant is competitor 1 so the completion time would be 15 units.

Your answer must include a clear description of the algorithm and a proof of how your greedy choice leads to an optimal solution. It is NOT necessary for you to analyze (or even state) the running time of the algorithm.