Algorithms and Data Structures, Academic Year 2010/2011

International Bologna Master in Bioinformatics

Please complete the following exercises by applying the concepts that have been illustrated to you during the classes. The score associated with each exercise and the expected time for completion is reported in the first line. Do NOT copy/exchange results (the parameters of each exercise are different).

Exercise 0 (2 points): write your name and surname in the first row of all the sheets you use.

Name:_____ Surname:_____

Exercise 1 (48 points, 60 minutes): please design the data structures that you would implement to efficiently realize a "random walk on a graph" game as follows. A set of N threads is executed, and each thread mimics the steps of one person walking randomly over the vertexes of a undirected graph G(V,E) defined as an input of the problem. One thread in vertex V can travel in one step only to one of the randomly selected vertexes directly connected to vertex V in the graph G. Imagine that you stop the execution after some time: your task is to design the graph G and the additional data structures required to efficiently answer to the following questions.

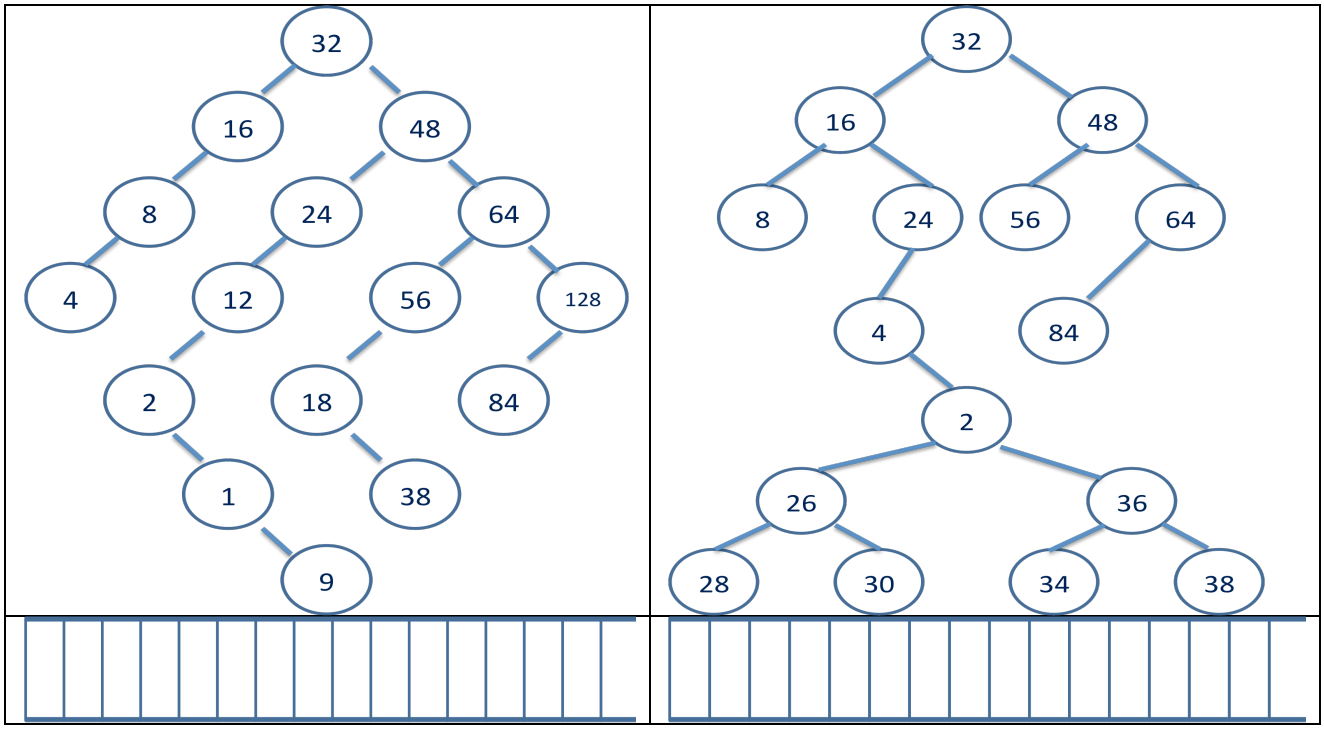
1. How many vertexes have been traversed by thread X, so far? (input X in [1..N])
2. How many times any vertex V has been cumulatively "visited" by all the walking threads, so far?
3. Which vertex is the most visited one?

For each implementation of the solutions to the questions above, please provide a motivation for your design, and a sketchy discussion of average/worst-case complexity in space and computation.

(use additional sheets for this exercise)

Name:_____ Surname:_____

Exercise 2 (10 points, 15 minutes): please provide the ordered sequence of visited nodes in a in-order visit of the left binary tree, and post-order visit of the right tree, respectively:



Exercise 3 (40 points, 45 minutes): please provide the ordered sequence of visited nodes in a Breadth-First-Search (DFS) visit of the undirected graph G, starting from node A, by using the Adjacency Set implementation (also show the Adjacency Set data structure, and plot the graph).

G=(V,E), V={A, B, C, D, E}, E={[A,B][A,D][B,C][B,D][C,D][C,E][D,E]},

Exercise 1 (48 points, 60 minutes): please design the data structures that you would implement to efficiently realize a "search the most used word in a text" service as follows. A set of N words is read from a file. The function *next_word()* provides you the next word from the file (returns NULL if the file is completely read or empty). Your task is to design the pseudo code of the algorithm and the additional data structures required to efficiently answer to the following questions.

1. Which one is the most used word (ties should be solved in alphabetical order);
2. Which one is the second most used word?
3. How many different words are contained in the file? (same words counted only once)

For each implementation of the solutions to the questions above, please provide a motivation for your design, and a sketchy discussion of average/worst-case complexity in space and computation.

(use additional sheets for this exercise)

Exercise 3 (40 points, 45 minutes): please provide the ordered sequence of visited nodes in a Breadth-First-Search (BFS) visit of the undirected graph G, starting from node A, by using the Adjacency Set implementation (also show the Adjacency Set data structure, and plot the graph).

G=(V,E), V={A, B, C, D, E, F}, E={[A,B][A,D][B,C][B,D][C,D][C,E][D,E][B,F][D,F]},

Exercise 1 (48 points, 60 minutes): please design the data structures that you would implement to efficiently realize a "cataloguing system" service as follows. A set of (1..N) service classes is identified, and a system is receiving service requests to be inserted in a storage system of a server. Each service request has an ID (an increasing integer number assigned upon reception), a date indicating the day DD/MM/YYYY in the future when the service is to be provided (that is, not the day the request is received), and the integer value indicating the service class (1..N). Your task is to design the pseudo code of the algorithm and all the data structures required to efficiently answer to the following requirements:

1. Define the *insert_new_request(ID, DD/MM/YYYY, Class)* function which stores the request in the memory of a server, by defining the data structure which provides a good compromise between the FASTER solution for a quick insertion, and FASTER extraction of the first en service request for a given day, belonging to a given service class, as required by the function extract_request() below:
2. Implement the pseudo code for the *extract_request(DD/MM/YYYY, Class)* which realizes the FASTER solution for extraction of the first request in the storage scheduled for day DD/MM/YYYY (if any) belonging to the specific service class passed as argument.
3. Discuss the execution and space complexity of the realized functions.

For each implementation of the solutions to the questions above, please provide a motivation for your design, and a sketchy discussion of average/worst-case complexity in space and computation.

(use additional sheets for this exercise)

Exercise 1 (48 points, 60 minutes): please design the data structures that you would implement to efficiently realize a "search the matching DNA sequence" problem as follows. A set of GCAT characters is used to create words representing DNA sequences. One suspect DNA sequence is provided as input of the problem. A number of DNA sequences is read from a file. Each sequence is a string of characters terminated by a Newline character and may have a variable number of characters. The function *next_DNA_seq()* provides you the next word from the file (returns NULL if the file is completely read or empty). Your task is to design the pseudo code of the algorithm and the additional data structures required to efficiently answer to the following questions.

1. Return "true" if there is a sequence in the file matching the suspect sequence.
2. If the sequence is not found in the file, return the sequence with the highest number of GCAT characters matching the suspect sequence, character by character, in their respective positions.

For each implementation of the solutions to the questions above, please provide a motivation for your design, and a sketchy discussion of average/worst-case complexity in space and computation.

(use additional sheets for this exercise)