הבינתחומי הרצליה
IDC HERZLIYA

Data Structures & Algorithms – Problem set 4 - Solutions

Due: 2.1.2022

Honor code:

1. Do not copy the answers from any source.
2. You may work in small groups but take no written notes, and write your solution on your own (mention collaborators in the submission).

Submission guidelines:

- Submit your solution via Moodle as a PDF file only. Other formats will not be graded.
- Typed submissions will get a bonus of 5 points.
- If you choose not to type your solution, make sure the scan is easy to read. We will deduct points for hard-to-read submissions.

Question 1 (10 points)

a. Given two BST T_1, T_2, suggest an efficient (i.e. with minimal running time) algorithm for checking whether the elements of T_1 and T_2 are the same (same values, regardless of position in each tree). Write the runtime of the algorithm.

b. Suggest a way to augment an AVL tree so that the following query can be answered in time $O(1)$:

Given a node v, compute the maximal difference between the keys of two nodes in the tree T_v. Here, T_v is the subtree rooted at v.

Your answer should detail any changes made to the AVL tree studied in class. This includes possible changes to existing functions of the tree.

You may assume that the values stored in the nodes are numeric.

Question 2 – (10 points)

Consider 3 hash tables H_1, H_2, H_3 of size 13, with an associated hash function $h(x) = x \ mod \ 13$.

All three hash tables handle collision with open addressing, where for H_1 we use linear hashing i.e $h_i(x) = x + i \ mod \ 13$, for H_2 we use quadratic hashing $h_i(x) = x + i^2 \ mod \ 13$ and for H_3 we use double hashing s.t $h'(x) = x \ mod \ 7 + 1$.

a) Insert the following set of keys into the three tables, in the given order (from left to right): 65,10,25,49,11,13,37 draw the resulted tables, and write the number of probes performed for each element.

b) What is the load factor $\alpha = \frac{n}{m}$ of the tables?

Question 3 (15 points):

Your local grocery shop decided to enter the 21st century and arrange all orders in a data structure. Each order is composed from a vegetable type and the delivery address. Orders are served on a first-come first-serve basis. That is, the first order to be shipped will be the first received order, the second order to be shipped will be the order which was received second, and so on. Orders are shipped one by one. Moreover, to avoid future shortages, the grocery owner wants to know the following piece of information: For each vegetable type in his/her store, how many different orders, which have not been served yet, demand this specific vegetable.

Design a data structure that supports the following operations in the given time:

a. insert(vegetable, address) – Inserts a new order. Running time-O(1) expected.

b. ship() – Ship the next order in line. Running time - O(1) expected.

c. query(vegetable) - Returns the number of different orders, which have not been served yet, which demand this specific vegetable. Running time - O(1) expected.

It is known that there are n different types of vegetables in the grocery. The data structure may hold at most m orders at any given time. We know that $m < n$. The data structure may use O(m) memory.

Question 4 (15 points + 10 bonus):

Determine True/False. Prove your answers.

a) For an undirected graph $G = (V, E)$, The number of vertices with an odd degree in G is even.

b) Let $G = (V, E)$ be an undirected connected graph. Let x be the minimum amount edges one needs to add to G so that the resulting graph has an Euler cycle. Then $x \leq \lfloor \frac{n}{2} \rfloor$.

c) Let G be an undirected connected graph whose vertices are numbered from 1 to $|V|$. When running an algorithm that can choose between two vertices, the algorithm will always choose the vertex with the smaller index. BFS and DFS on G starting from the vertex with index 1 will always output the same spanning tree.

d) (5 points bonus) A directed graph G, for which each vertex is either a sink (out-degree=0) or a source (in-degree=0), is a bipartite graph.

e) (5 points bonus) Let $G = (V, E)$ be a simple directed graph (G has no self-loops, and each directed edge (u, v) may appear at most once), s.t. for every $v \in V : d_{in}(v) + d_{out}(v) > \frac{n}{2}$. Then G has a Hamiltonian path.

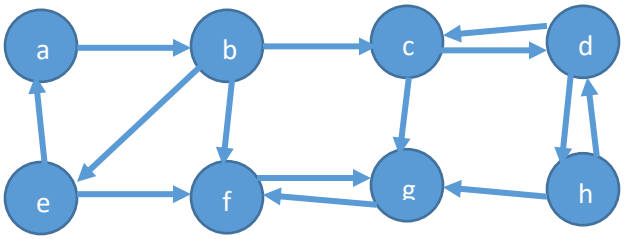
הבינתחומי הרצליה
IDC HERZLIYA

Question 5 (25 points):

Definition: Let $G = (V, E)$ be a directed graph. A strongly connected component W is a subset of the nodes, $W \subseteq V$, such that every node in W is reachable from any other node in W.

The strongly connected components are actually a partition of the node set V.

To make sure you understand the definition, consider this graph:



Calculate its strongly connected components. You should find out there are exactly 3.

Definition: Let $G = (V, E)$ be a directed graph. The SCC graph $G^{SCC} = (V^{SCC}, E^{SCC})$ is defined like this:
The set of nodes V^{SCC} is the set of strongly connected components of G. This means every node in G^{SCC} represents a set of nodes in G.
Let $W_1, W_2 \in V^{SCC}$ be strongly connected components of G. There is an edge in E^{SCC} from W_1 to W_2 iff there is an edge in G from some vertex in W_1 to some vertex in W_2.

Consider the graph above and calculate its SCC graph. You should find out it's a directed triangle.

Let $G = (V, E)$ be a directed graph.

a) Prove:
Let $W, W' \in V^{SCC}$ be strongly connected components such that there is an edge from W to W' in G^{SCC}. Then for any nodes $u \in W$, $u' \in W'$ there is a path from u to u' in G.

b) Prove:
Let $W, W' \in V^{SCC}$ be strongly connected components such that there is a path from W to W' in G^{SCC}. Let $u \in W$ and $u' \in W'$ nodes in G. Then there is a path in G from u to u'. (use (a))

c) Let $G = (V, E)$ be a directed graph. Prove that its SCC graph, G^{SCC}, is acyclic. (use (b))

Question 6 (25 points):

Given is a directed unweighted graph $G = (V, E)$. Some of the edges are colored red. Let $E' \subseteq E$ be the set of red edges. Let $s \in V$. Suggest an efficient algorithm for finding the length of a shortest path from s to each of the other vertices in the graph, fulfilling the following condition: The path includes at most two red edges. In other words, every $v \in V$ should be labeled by the length of a shortest path from s to v in which there are at most two edges from E' and any number of edges from $E \setminus E'$. Describe the algorithm; justify its correctness.

We devise here a possible direction for solution. You may either continue in this direction or find a completely different algorithm.

Construct $\boldsymbol{G^* = (V^*, E^*)}$, where: $\boldsymbol{V^* = V^1 \cup V^2 \cup V^3}$ consists of three copies of the original vertex set $\boldsymbol{V}$.

Define $\boldsymbol{E^*}$ as follows: for any $\boldsymbol{(u, v) \in E \setminus E'}$ add the edges $\boldsymbol{(u^1, v^1)}$, $\boldsymbol{(u^2, v^2)}$, $\boldsymbol{(u^3, v^3)}$ to $\boldsymbol{E^*}$. For any $\boldsymbol{(u, v) \in E'}$ add the edges $\boldsymbol{(u^1, v^2)}$, $\boldsymbol{(u^2, v^3)}$ to $\boldsymbol{E^*}$.