1 Introduction and Objective

This tutorial marks the end of the first \( \frac{1}{3} \) of CS2010: Various Data Structures (DSes). In the past three weeks, we have learned about tree-related Data Structures: Binary Heap, Binary Search Tree, and AVL Tree as a form of Balanced Binary Search Tree. In this tutorial, we discuss more DSes that were discussed in Lecture 05: Union-Find Disjoint Sets (UFDS), Bitmask, and three basic Graph DSes: Adjacency Matrix, Adjacency List, and Edge List.


As we will have the supposedly easier 4% Online Quiz 1 on Thursday of Week 06 and the supposedly more challenging 10% Written Quiz 1 on Saturday of Week 06, we have made the questions in this tutorial slightly shorter so that students can discuss Online/Written Quiz 1 related materials with the tutor if needed.
2 Tutorial 04 Questions

Basic Stuffs About UFDS, Bitmask, and Graph DSes

Q1. First, tutor will choose one of the random UFDS [http://visualgo.net/ufds.html](http://visualgo.net/ufds.html) menu Samples) or initialize an UFDS with $N$ disjoint sets ($N$ is chosen at random). Then, tutor will ask students to perform $\text{FindSet}(i)$ and $\text{UnionSet}(i, j)$ operations on that UFDS. We assume that both ‘path compression’ and ‘union by rank’ heuristics are used.

This is an open ended exercise. Tutor is expected to re-explain quickly about these two operations. Tutor will highlight the importance of two heuristics: ‘path-compression’ (to compress the path at every $\text{FindSet}(i)$ operation, if applicable) and ‘union-by-rank’ (to keep the resulting combined tree as short as possible) in UFDS. Note that the $\text{Initialize}(N)$ and $\text{IsSameSet}(i, j)$ operations are quite trivial and can be skipped.

Q2. Second, tutor will select a random (and small) positive integer $S$ (in base 10) [http://visualgo.net/bitmask.html](http://visualgo.net/bitmask.html) menu Set S) and describe the binary (base 2) representation of that integer. Then, tutor will ask students to perform several basic bit manipulation operations on that random integer: Setting on a certain bit, Checking if a certain bit is on, Toggle the state of a certain bit (a NEW operation, check VisuAlgo), Clear a certain bit (a NEW operation, check VisuAlgo), identify the Least Significant Bit of that random integer, that is, the last bit that is on of that random integer (also a NEW operation, check VisuAlgo).

This is also an open ended exercise. Tutor is expected to re-explain quickly about how an integer (in base 10) is actually stored as binary (base 2) in computer memory. Then tutor will highlight the importance of left shift operation to position bit 1 at the target bit, then highlight that Set/Check/Toggle/Clear operations are very similar. Set and Check have been discussed in Lecture 05. Discuss that Toggle and Clear are very similar. Tutor will spend some time to explain the LSOne operation that involves Two’s complement, use VisuAlgo to help you understand this operation.

Q3. Third, tutor will draw random small graph on whiteboard and ask students to store that graph in either Adjacency Matrix, Adjacency List, or Edge List data structure. Then, the tutorial group can compare that answer by drawing the same small graph on [http://visualgo.net/graphds.html](http://visualgo.net/graphds.html).

This is also an open ended exercise. Tutor will actually do one more round on top of this published task: The reverse challenge, e.g. from a random small edge list, ask student to draw the one of the possible graph drawing out of that edge list and tell them that when we are given the graph drawing, it is easy to store it into a graph DS (and lost the graph drawing itself), but when we are only given the content of a graph DS, there are infinitely many possible graph drawings that fit into the data stored in that graph DS as the actual layout is not actually stored. Follow up question: What if we also want to store the graph drawing layout like in VisuAlgo?
Not-So-Basic Stuffs

Q4. Given \( n \) disjoint sets initially in a UFDS, is it possible to call \texttt{unionSet}(i, j) and/or \texttt{findSet}(i) operations to get a single tree with actual height \( h \) that represents a certain set? Both ‘path-compression’ and ‘union-by-rank’ heuristics are used.

This question has been integrated in VisuAlgo Online Quiz :).

The answer is yes if \( h \leq \log_2 n \). The reason is because we can only increase height of the resulting combined UFDS tree if both subtrees previously have the same rank (due to ‘union-by-rank’ heuristic). We need to do a very careful \texttt{unionSet}(i, j) operations without ever calling \texttt{findSet}(i) operations (thus no ‘path-compression’ ever happen). The careful union operations are as follows: From \( n \) rank 0 subtrees, union the root of these subtrees into \( n/2 \) rank 1 subtrees. Then pair the root of these subtrees into \( n/4 \) rank 2 subtrees... And so on until we have 1 rank \( \log_2 n \) tree. Tutor will point that this is the challenge of Slide 18 (VisuAlgo UFDS Exercise (2)) of \url{http://www.comp.nus.edu.sg/~stevenha/cs2010/lectures/L05-The-Foundations.pdf} that is skipped during Lecture 05.

Q5. Describe a simple \( O(n) \) algorithm to count how many bit(s) in an integer \( S \) that is/are on where \( n \) is the number of bits present in \( S \).

Can you do that in \( O(k) \) where \( k \) is the answer that we are looking for (the number of bit(s) that is/are actually on in \( S \))?

This question appear in last year final exam :).

The \( O(n) \) algorithm is obvious, call \texttt{CheckBit}(i, S) operation from bit 0 to bit \( n - 1 \).

The \( O(k) \) algorithm is probably not that obvious: call \texttt{LSOne}(S) and subtract the returned value that from \( S \) until \( S \) is 0. We can only do this \( k \) times :). Again, this it an output-sensitive algorithm. Note to tutor: It is important that the discussion of \texttt{LSOne}(S) operation is done properly in step 3 above.

Q6. Draw a Directed Acyclic Graph with \( n \) vertices and \( n \times (n - 1)/2 \) edges.

This question has been integrated in VisuAlgo Online Quiz :).

Easy with experience. Just draw a line containing \( n \) vertices labeled from 0 to \( n - 1 \). Then draw directed edge from a vertex \( i \) to a vertex \( j \) for all pair of \( (i, j) \) where \( i < j \), that’s all.

That is the end of Tutorial 04 for now. Students can stay back and ask tutor about questions in Online Quiz 1 or in past Written Quiz 1 papers (no ongoing PS this week).