Cs445 — Homework #7
Due: 12/**9**/2015 midnight

Instructions.

1. Solution could **not** be submitted by students in pairs.
2. You could submit a pdf of the homework, either printed or hand-written and scanned, as long as it is **easily** readable.
3. If your solution is not clearly written, it might not be graded.
4. Prove the correctness of your answer. A correct answer without a proof might not be accepted.
5. If you have discussed the solution with other students, mention their names clearly on the homework. These discussions are not forbidden and are actually **encouraged**. However, you must write your whole solution yourself.
6. All questions have same weight.
7. You could refer to a data structure studied in class, and just mention briefly their guaranties. For example "*It is known that a Red-Black tree could support the insert/delete/find operations on a set of n elements in time $O(\log n)$.*
8. If your answer uses one of the data structures or algorithms that were studied in class, you could refer to it without having to repeat details studied in class. If you answer requires only modifications of one of the algorithm, it is enough to mention the required modifications, and what's the effect (if any) on the running time and on other operations that the algorithm performs.
9. In general, a complete answer should contain the following parts:
   (a) High level description of the data structures (if needed). *E.g. We use a binary balanced search tree. Each node contains, a key and pointers to its children. We augment the tree so each node also contains a field...*
   (b) High level description of the algorithms
   (c) Proof of correctness (why your algorithm provides what is required).
   (d) A claim about the running time, and a proof showing this claim.

1. Suggest modifications to the U/F data structure, such that you could also support the following operations

   (a) `report`(j) that prints all the elements in a set containing the atom j. The time for this operation is $O(k)$ where k is the number of elements in this set. The time complexities of the `Union`(i,j) and for the `Find`(i,j) operations should stay the same as in the original structure studied in class.

   (b) `Count`(j) that prints the number of elements in a set containing the atom j. The time for this operations is $O(\log k)$ where k is the number of atoms in this set. The time complexities of the `Union`(i,j) and for the `Find`(i,j) operations should stay the same as in the original structure studied in class.

2. The question refers to the Union/Find data structure for a set $\{1 \ldots n\}$ of atoms, where initially each set contains only a single atom. Suggest a sequence of `Union`(i,j) operations yielding a tree of height $\geq \log_2 n$, even if the path compression strately is used.

   As usual, n is an arbitrary large integer. You could assume however that $n = 2^r$ for an integer r.

3. In the structure studied in class, during union operation, the root of the tree that contains fewer nodes becomes a child of the root of the tree that contains more nodes. The following tactic is suggested for union: Each root maintains the height of its tree. On each union operation the node of the lower tree is becoming a child of the root of the taller tree. Prove that with this tactic, the maximum height of a tree is $\leq \log_2 n$. (the proof is similar to the one seen in class).

4. Let $G(V,E)$ be an undirected graph with weights assigned to its edges. The input is given where edges appears in an increasing order of weights. $E = \{e_1, e_2 \ldots e_m\}$ and

$$w(e_1) < w(e_2) < \cdots < w(e_m)\}$$

   Let G_r be the graph obtained from G by removing all edges with weight $> r$.

   Let k be given parameters. Suggest an $O(m\ \alpha(m))$-time algorithm for finding the minimum value of r, such that G_r has k connected components.

5. **Bonus 5 points** Same as previous question, but now find the minimum r such that in G_r every connected component has $\geq k$ vertices.