Ollscoil na hÉireann
The National University of Ireland

Coláiste na hOllscoile, Corcaigh
University College, Cork

In-Class Quiz 2
2014

CS4407 Algorithm Analysis

Prof. Ian Gent (extern)
Prof. G. Provan
Prof. B. O'Sullivan (HoD)

Attempt all questions

Total marks: 50

50 minutes

Please answer all questions
Points for each question are indicated by [xx]

1. **[15]** A directed graph G = (V, E) is singly-connected if there is at most one directed path from u to v for all vertices u, v ∈ V .
    a. Give an efficient algorithm to determine whether or not a directed graph is singly connected.
    b. Define the complexity of your algorithm.

    Solution. For each vertex u ∈ V , perform a DFS on the given graph G. Check if there are any foward edges or cross edges (in the same component) 1 in any of the searches. If no such edges exist, then the graph is singly connected, else not.

    Time complexity: O(|V |(|V | + |E|)).

2. **[15]** Consider a set of *m* people and *n* jobs, $m<n$, where each person ranks the subset of $k \leq n$ jobs for which she is suitable. A matching is an assignment of a person to a job, and a maximum-weight matching is a matching whose ranking is highest among all ranked matchings.
    a. [**10**] Show the pseudo-code for an O(*n* log *n*) time greedy algorithm to solve this problem.
    b. [**5**] Either show that this algorithm is optimal, or provide a counter-example to show that it is not optimal.

3. **[20]** Given a graph G and a minimum spanning tree T, suppose that we decrease the weight of one of the edges not in T. Give an algorithm for finding the minimum spanning tree in the modified graph. What is the complexity of this algorithm?

Call the edge whose cost we decreased e=(u,v), and its new cost is w*. The only possible change resulting from this decrease is that e may now belong to the new MST, and instead another edge should be thrown out.

Method 1:

First, we first test if the value of $w^*>w_k$, where w_k is the most expensive edge in T; if so, then e will not be included in T and we just return T. Else, we test if e can replace some edge in T, possibly in a cycle induced by including e. First, we must test the cycle property Let T be the MST of G before the cost decrease. Then, we run BFS in time O(n) to find the unique path connecting u and v on T; call it P. Once we have found the path, we can check in time O(n) whether any edge on P has cost more than w*. If so, we replace the most expensive edge of P with e to obtain the new MST. Otherwise, we know that even after the cost decrease, e should not be part of the MST by the cycle property (because it is most expensive on at least one cycle, namely $P \cup \{e\}$).

Method 2:

First, we first test if the value of $w^*>w_k$, where w_k is the most expensive edge in T; if so, then e will not be included in T and we just return T.
Else,
Set $T'=T-\{a \in T|\ w(a)>w^*\}$, where a is an edge in T, and w(a) is the weight of edge a.
Starting with T', run Kruskal's algorithm to generate the new MST that will augment T'.

4. [20] Consider the following scheduling problem:
    INPUT: A set $S = \{(x_i, y_i) | 1 \leq i \leq n\}$ of intervals over the real line.

OUTPUT: A maximum cardinality subset *T* of *S* such that no pair of intervals in *T* overlap.

Consider the following greedy algorithm:

1. $T=\emptyset$.
2. Repeat until *S* is empty.
3. Select the interval *I* that overlaps the least number of other intervals in *S*.
4. Add *I* to the initial solution set *T*: $T \leftarrow T \cup I$
5. Remove all intervals from *S* that overlap with *I*.
6. Return *T*.

a) [5] What is the invariant for this problem?

The invariant is the fact that no intervals in T will overlap. This invariant is true at line 1 ($T=\emptyset$), and remains true because of line 5.

b) [5] What is the complexity of the algorithm?

If we can compute an overlap in O(1) time, then lines 3 and 5 are each $O(|S|)$, and we go through the loop at most $O(|S|)$ times, so the total complexity is $O(|S|^2)$.

c) [10] Prove or disprove that this algorithm solves the problem correctly.

The following counterexample shows the algorithm is sub-optimal. We will choose interval A first (2 overlaps), and this choice means that we can never choose the subset {B,C,D,E}, which gives 4 intervals. We end up having only 3 intervals.