Illinois Institute of Technology
Department of Computer Science

Solutions to Third Examination

CS 430 Introduction to Algorithms
Fall, 2014

Wednesday, December 3, 2014
10am–11:15pm, 121 Life Sciences
11:25am–12:40pm, 113 Stuart Building

Exam Statistics

88 students took the exam. 9 students were absent; their zeros are not counted in the statistics that follow. The range of scores was 10–77, with a mean of 38.52, a median of 36, and a standard deviation of 15.52. Very roughly speaking, if I had to assign final grades on the basis of this exam only, 60 and above would be an A (11), 39–59 a B (27), 25–38 a C (34), 15–24 a D (11), below 15 an E (5).

Problem Solutions

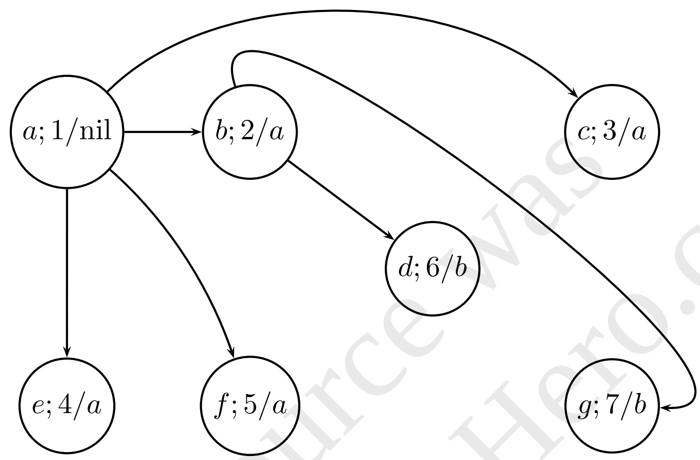
1. (a) We do a Make-Set operation for each job, then we do a Union for each "same processor" constraint, then we do a Find-Set for each job in a "different processor" constraint. If any of the Find-Set operations has the two jobs in the same set, the constraints cannot be satisfied:

```
for i := 1, 2, ..., n do
    Make-Set(i)
end for
for i := 1, 2, ..., m do
    Union(s_i, t_i)
end for
for i := 1, 2, ..., k do
    if Find-Set(s_i) = Find-Set(t_i) then
        return Constraints cannot be satisfied
    end if
end for
return Constraints can be satisfied
```

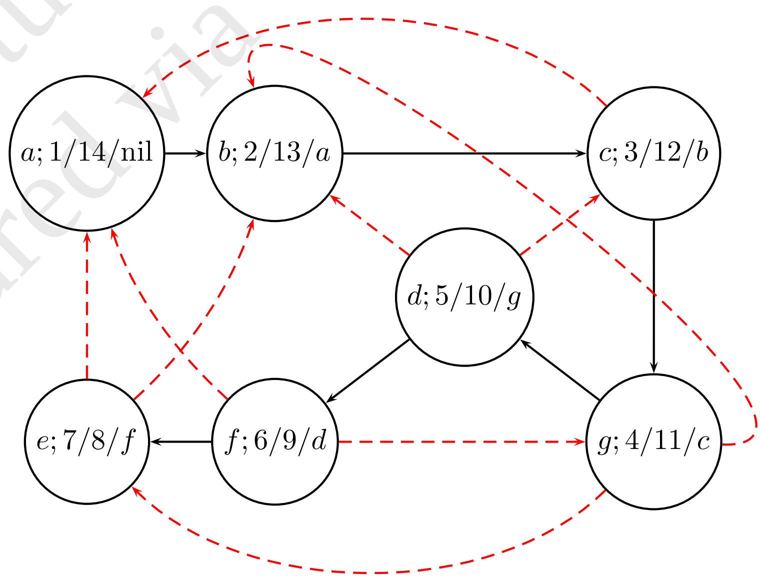
   (b) If we use weighted union and path compression, according to Theorem 21.14 on page 581 of CLRS3 each of the operations Make-Set, Union, Find-Set uses $\alpha(n)$ amortized time, so the sequence of $n + m + 2k$ operations use time $O((n + m + k)\alpha(n))$.

   Because section 21.4 was only suggested reading, the following analysis also received full credit: There are n Make-Set operations, m Union operations, and $2k$ Find-Set operations. According to Theorem 21.1 on page 566 of CLRS3, with weighted union, each of the MAKE-SET operations is worst-case time $O(1)$; the UNION and FIND-SET operations are worst-case time $O(\log n)$, so the sequence of $n + m + 2k$ operations are worst-case time $O(n + (m + k) \log n)$.

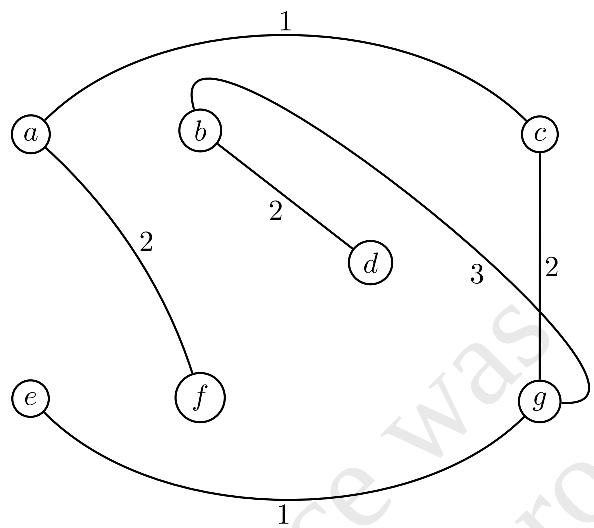
2. (a) A possible breadth-first search tree starting at vertex a (there are many), with edges ignored by BFS omitted and each vertex labeled with the discovery time/predecessor vertex $v.d/v.\pi$, is



(b) A possible depth-first search tree starting at vertex a (there are many), with tree edges shown as black solid lines and back edges shown as red dashed lines; each vertex is labeled with the discovery time/finishing time/predecessor vertex $v.d/v.f/v.\pi$, is



(c), (d), and (e) The minimum spanning tree is unique, so both Kruskal's algorithm and Prim's algorithm give the identical (that is, the only) cost 11 spanning tree:



3. (a) Dijkstra's algorithm is unchanged! All one need do is change the *relaxation step*. In CLRS3 (page 649) RELAX(u, v, w) is given as

   if $v.d > u.d + w(u, v)$ **then**
   $\quad v.d = u.d + w(u, v)$
   $\quad v.\pi = u$
   end if

   so we change it to

   if $\max\{u.d, w(u, v)\} < v.d$ **then**
   $\quad v.d = \max\{u.d, w(u, v)\}$
   $\quad v.\pi = u$
   end if

   (b) Negative edge weights make no difference because we are taking the max, not adding, so going around a negative "thickness" cycle does not change the thickness (as it does in the case of lengths, which are added).

   (c) The triangle inequality still holds, so the proof of correctness has only the most minor changes to Theorem 24.6 (pages 659–661).

4. Given an algorithm for HAM-CYCLE, we can use it to solve a HAM-PATH problem on graph G by adding a single vertex s which has an edge to every vertex of G; call this augmented graph G'. Now if G' has a Hamiltonian cycle if and only if G has a Hamiltonian path, so solving HAM-CYCLE on G' is an algorithm for HAM-PATH on G.

On the other hand, given an algorithm for HAM-PATH, we can use it to solve a HAM-CYCLE problem on graph G as follows: If G has a Hamiltonian cycle containing edge e, the G' obtained from G by deleting e has a Hamiltonian path (that starts one of the endpoints of e and ends at the other). We try this test for every edge of G; if none of them yields a Hamiltonian path, G does not contain a Hamiltonian cycle. This requires at most $|V|(|V|-1)/2$ applications of the HAM-PATH algorithm—a polynomial number.

5. The last positions (those columns corresponding to the clauses) *must* have a 4 in the target number because we need at least one true variable in each (a 1 in the literal's row). So if we allowed the last positions in the target to be 1, 2, or 3 we could get those values using only the slack rows with all the literals being 0 (false). The *only* way to get a 4 is to have at least one literal be true.