CSC373— Algorithm Design, Analysis, and Complexity — Spring 2018

Assignment 1: Greedy Algorithms   Due: 9am, Thurs, Jan 25, 2018

Submit answers to the course MarkUs page, `https://markus.teach.cs.toronto.edu/csc373-2018-01`. Your solutions must be in one PDF file named A1.pdf. Your solutions can be neatly hand-written and scanned. You do not need to repeat the questions themselves.

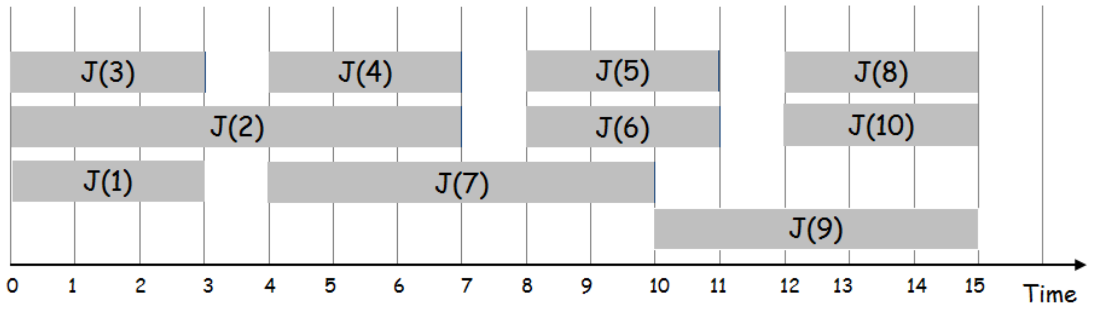
You are encouraged to work in groups of size at most five. If you do work in a group then you must, of course, submit this work to MarkUs as a group.

1. Schedule Blocking. The input information for this problem is the same as the interval scheduling problem considered in class. That is, suppose you are given a set of jobs, $\{J(k)\}_{k=1}^{K}$ where each job $J(k)$ must run in the (real-valued) time interval $(s_k, f_k) \subset \mathbb{R}$. Here we assume that these interval endpoints satisfy $0 \leq s_k < f_k$ where s_k and f_k are non-negative integers for all k.

We will refer to these jobs simply by their indicies $1, 2, \ldots K$. We will abuse this notation slightly and simply refer to $J(k)$ as "job k".

A subset of jobs $C \subseteq \{1, 2, \ldots K\}$ is said to be **compatible** iff for each pair $k, j \in C$, with $k \neq j$, we have $(s_k, f_k) \cap (s_j, f_j) = \emptyset$. Otherwise C is said to be incompatible. A subset $C \subseteq \{1, 2, \ldots K\}$ is said to be a **blocking subset** iff C is compatible and, for each $k \notin C$ with $1 \leq k \leq K$, the set $C \cup \{k\}$ is incompatible. That is, for a blocking subset C, no additional job k can added to the set C to form a larger compatible set (i.e., after scheduling the jobs in C, all other jobs are blocked from being scheduled).

Problem. Given input intervals of the above form $\{J(k)\}_{k=1}^{K}$, find a minimimum-sized blocking subset B.



For the example above we have $K = 10$ input intervals. Note that, since we consider the jobs to be executed in open time intervals, the two jobs $J(7)$ and $J(9)$ are compatible even though one starts and the other ends at time 10. Observe that $C = \{3, 7, 10\}$ forms a blocking subset since: a) C is compatible and, b) there is no additional job that is compatible with jobs already in C. Can we find a blocking subset that has size less than $|C| = 3$? Indeed, $B = \{2, 9\}$ is such a blocking subset of size two, and this turns out to be of the minimum possible size for this example. So a possible solution for this case would be $B = \{2, 9\}$.

[5pts] 1a) Your instructor, Professor Jot, suggests you investigate using a greedy algorithm that first sorts the jobs by the number of conflicts, and then processes the jobs in that order. He points out that jobs 2, 7 and 9 in the above example each conflict with four other jobs (e.g., job 2 conflicts with jobs 1, 3, 4 and 7), while all the other jobs just conflict with two other jobs. He suggests that perhaps a greedy solution can be developed from this idea, although he also notes that, even on the current example, you still need to figure out how to break the tie between the maximally conflicting jobs 2, 7 and 9.

After you try this idea at home for awhile you realize that such an approach cannot work.

To clearly demonstrate this to Prof. Jot, show an example where there is only one job with the maximum number of conflicts (so any such greedy-by-conflict approach would have to choose this job first), yet this job is not in any minimum-sized blocking subset for that example.

[10pts] 1b) Specify a greedy algorithm for solving the minimum-sized blocking subset problem. Your algorithm must have the following general form.

```
Input: The intervals $J(k) = (s_k, f_k)$, where $0 \le s_k < f_k$ are integers for $1 \le k \le K$.
Preprocessing requiring at most $O(K^2)$ time, e.g., perhaps sorting.
$B \leftarrow \{\}$                # $B$ is the set of indicies for the blocking set so far.
$U \leftarrow \{1, 2, \ldots, k\}$   # $U$ is the set of unprocessed job indicies.
**Loop Invariant:** $L(B, U)$ (see part 1c) must be true.
while $U \neq \emptyset$:
    Greedily decide how to update $B$ and $U$ in a way that does not typically require
        looking at all the elements in $U$ (since that's what makes it greedy).
    Remove at least one element from $U$ and possibly update $B$.
    **Loop Invariant:** $L(B, U)$ (see part 1c) must be true.
end while
**Assertion**: The set $B$ is a minimum-sized blocking subset.
Return $B$
```

Here $\leftarrow$ denotes an assignment operation.

[5pts] 1c) Specify the loop invariant $L(B, U)$, noted in the code above, that is true for your algorithm and is key to proving your algorithm is correct (i.e., it terminates with the last assertion above being true). Briefly explain why you think this loop invariant is true and how it can be used to prove your algorithm is correct. (You do not need to hand in a detailed proof, just the key ideas for proving your loop invariant. These must be sufficiently clear so the marker believes that you could fill in the details.)

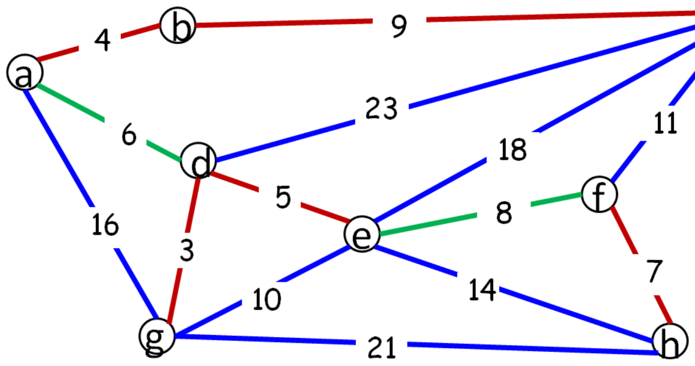
2. Minimum Spanning Tree. Given a weighted, undirected, and connected graph $G = (V, E, w)$, where the weights $w(e)$ are all distinct integers (i.e., for any two different edges e and f in E we have $w(e) \neq w(f)$), consider the following algorithm for computing an MST:

```
Input $(G, V, w)$ as above.
Let $F = \{(v, \emptyset) \mid v \in V\}$, which is the set of all trees
    formed by a single vertex. So $F$ is a forest with size $|F| = |V|$.
**Assertion 0:** The loop invariant $L(F)$ is true (see question 2b)
while $|F| > 1$:
    Set $J \leftarrow \emptyset$.
    For each tree $T \in F$:
        Let $S$ be the set of vertices in $T$.
        Let $D$ be the set of edges $e = (u, v) \in E$ such that $u \in S$ and $v \notin S$.
        Let $e = (u, v)$ be the edge in $D$ with the minimum weight $w(e)$.
        Set $J \leftarrow J \cup \{e\}$.
    end for
    Define $F'$ to be the new set of connected subgraphs formed by adding all the edges in $J$
        to the forest $F$. This will join trees in $F$ together. See example below.
    **Assertion 1**: $|F'| < |F|$ and $F'$ is a forest (see question 2a)
    Set $F \leftarrow F'$
    **Loop Invariant:** $L(F)$ is true (see question 2b)
end while
**Assertion 2**: $F = \{T\}$ where T is an MST for $(G, V, w)$ (see question 2a)
Return the tree $T \in F$.
```

Here again $\leftarrow$ denotes an assignment operation.

An example graph (G, V, w) is shown below. The initial forest is simply $F = \{T = (\{v\}, \emptyset) \mid v \in V\}$. The edges in red are added after the first pass through the loop. For example, the edge (a, b) is included in J on this first pass because it is the edge with the smallest weight between the tree $(\{a\}, \emptyset) \in F$ and other vertices. (The same edge is also selected when the tree $(\{b\}, \emptyset) \in F$ is considered.) The edge (b, c) is selected when the tree $(\{c\}, \emptyset) \in F$ is considered, and so on. After this first pass, $F' = \{T_i \mid T_i = (V_i, E_i), 1 \le i \le 3\}$, where

$$\begin{aligned} V_1 =& \{a, b, c\}, \quad E_1 = \{(a, b), (b, c)\}, \\ V_2 =& \{d, e, g\}, \quad E_2 = \{(d, g), (d, e)\}, \\ V_3 =& \{f, h\}, \quad E_3 = \{(f, h)\}. \end{aligned}$$



These trees are denoted by the components connected by just the red edges in the figure above. On the next pass through the loop, the edge (a, d) is added to J since it is the minimum weight edge leaving $T_1 = (V_1, E_1)$ (and also T_2), while (e, f) is added to J as the minimum weight edge leaving T_3. These edges in J after the second pass through the loop are shown in green in the above figure. The construction of F' now results in just one connected component, say $F' = \{(V, E_T)\}$, where E_T consists of exactly the red and green edges above. The claim is that this subgraph, $T = (V, E_T)$, is the MST for this example.

[5pts] 2a) Sketch the proof of Assertion 1 above. You may assume that the current F is a forest (and you should probably note this in your loop invariant in part (2b)). You should include the key steps in your proof but you do not need to provide all the details. That is, you need to be clear enough that the marker believes you could complete a detailed proof.

[5pts] 2b) Precisely specify the loop invariant, $L(F)$. in the algorithm above such that it provides the key information necessary to prove that the algorithm eventually returns an MST. Note that your choice must also make Assertion 0 true. (Here you don't need to prove that this loop invariant must be true, just state it clearly. You use this loop invariant in parts 2c and 2d below.)

[5pts] 2c) Sketch a proof by induction that shows the loop invariant $L(F)$ is true initially and also after each iteration of the loop. Explain the major steps but omit the details, as described in part (2a).

[2pts] 2d) Given that your loop invariant, $L(F)$, is true after the last iteration of the algorithm, carefully prove Assertion 2, that is, the returned T is an MST for the input graph G. (Hint: This proof should be very short since if you have an appropriate loop invariant.)