Department of Computer Science — University of Wisconsin-Whitewater
Advanced Algorithm Final Exam

Abdoul-Nourou Yigo
Dr. Athula Gunawardena

December 2018

1 Question1

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove (i.e give a counter example) each of the following conjecture.

(a) $f(n) = O(g(n))$ implies $g(n) = O(f(n))$

Proof:(By contradiction on n) Suppose that the above condition is true for all $n \geq 0$. We know that $f(n) = O(g(n))$ would mean that $g(n)$ is growing faster than $f(n)$.
Also, $g(n) = O(f(n))$ which would imply that $f(n)$ is growing faster than $g(n)$ Since the same state of the condition could not be satisfy at the same time. The implication could not stand.
As a result, $f(n) = O(g(n))$ does not imply that $g(n) = O(f(n))$. $\blacksquare$

(b) $f(n) = O(g(n))$ implies that $g(n) = \Omega(f(n))$

Proof: We have $f(n) = O(g(n))$ implies that $g(n)$ is growing faster $f(n)$. Since $g(n) = \Omega(f(n))$ which implies that $f(n)$ is growing slower than $g(n)$. In other words, it satifies the first condition that saying that $g(n)$ grows faster than $f(n)$. As a result, the above condition stands. In other words, $f(n) = O(g(n))$ implies that $g(n) = \Omega(f(n))$

(c) $f(n) = O((f(n)^2)$

Proof: According to the notation of O we have
$O(g(n)) = \{m(n) : \exists\ c > 0,$ and $n_0 > 0$ such that $0 \leqslant f(n) \leqslant cg(n))$ for all n $\geqslant n_0$ $\}$

According to the above relation if f(n) $\leqslant 1$ for sufficient large value of n we could have this relation $0 \leqslant f(n) \leqslant c(f(n)^2)$. For instance, $f(n) = O(f(n)^2)$. However, this conjecture can not stand if $fn) < 1$. ■

(d) $max(f(n), g(n)) = \theta(f(n) + g(n))$

Proof:
Suppose that m(n) = max(f(n),g(n)), therefore we could write this relation,
$\theta(m(n)) = \{m(n) : \exists c_1, c_2,$ and $n_0 > 0$ such that $0 \leqslant c_1(f(n) + g(n)) \leqslant m(n) \leqslant c_2(f(n) + g(n))$ for all n $\geqslant n_0$ }

Since m(n) is the maximum of f(n) and g(n) for any n in m(n) we can write these relations $m(n) \geqslant g(n)$ and $m(n) \geqslant f(n)$ adding these two inequations we got $2m(n) \geqslant f(n) + g(n)$. Moving the coefficient to the other side we have $m(n) \geqslant 1/2(f(n) + g(n))$. According to the the θ-notation we have $0 \leqslant c_1(f(n) + g(n)) \leqslant m(n)$ for all $n \geqslant n_0$ since f(n)and g(n) are symptotically nonnegative. Therefore, we can select $c_1 = 1/2$, so we can have this relation $0 \leqslant 1/2(f(n) + g(n))$ for all $n \geqslant n_0$.

Similarly, since f(n) and g(n) are asymtotically nonnegative $\exists$, c_2 such that $f(n) \geqslant 0$ and $g(n) \geqslant 0$ for $n \geqslant n_0$. Therefore, $f(n) + g(n) \geqslant 0$. We know that any particular n in m(n) can be either from f(n) or g(n), so the addition of both sets would be greater than m(n). Therefore, we can have this relation $(f(n) + g(n)) \geqslant m(n)$. As a result, by identification $c_2 = 1$. To conclude we have this final relation $0 \leqslant 1/2(f(n) + g(n)) \leqslant m(n) \leqslant (f(n) + g(n))$ $\Rightarrow 0 \leqslant 1/2(f(n) + g(n)) \leqslant max(f(n), g(n)) \leqslant (f(n) + g(n))$ and $max(f(n), g(n)) = \theta(f(n) + g(n))$. ■

2 Question2

(a) $T(n) = \sqrt{2}T(\frac{n}{2}) + log(n)$
We have $a = \sqrt{2}$, $b = 2$, $f(n) = log(n)$
$n^{log_b a} = n^{log_2 \sqrt{2}}$
$n^{log_b a} = n^{\frac{1}{2}}$
$n^{log_b a} > f(n)$
By using case 1, we have $f(n) = O(n^{(\frac{1}{2} - \epsilon)}) \Rightarrow \epsilon > 0$
$f(n) = O(n^{log_2 \sqrt{2} - \epsilon})$ $f(n) = O(n^{\frac{1}{2} - \epsilon})$
Therefore, we have $\epsilon < \frac{1}{2}$, when we select $\epsilon = 0.25$, so $f(n) = O(n^{0.25})$.
Since we know that $log(n)$ is asymptotically less than any positive power of n.We could say that $log(n)$ is $O(n^{0.25})$.
As a result, $T(n) = \theta(n^{log_2 \sqrt{2}}) \Rightarrow T(n) = \theta(\sqrt{n})$

(b) $T(n) = 6T(\frac{n}{3}) + n^2 logn$

We have $a = 6$, $b = 3$, $f(n) = n^2 logn$
$n^{log_b a} = n^{log_3 6} \Rightarrow n^{log_3 6} = n^{log_3 3*2}$
$n^{log_3 3*2} = n^{log_3 3+log_3 2}$
$n^{1+log_3 2} = n^{1.63}$
Therefore, $f(n) > n^{log_b a}$ by using case 3 we need to check the regularity condition
$af(\frac{n}{b}) \leqslant cf(n)$, where $c < 1$
$6((\frac{n}{3})^2 log(\frac{n}{3})) \leqslant cn^2 log(n)$
$6((\frac{n^2}{9})log(\frac{n}{3})) \leqslant cn^2 log(n)$
$2((\frac{n^2}{3})log(\frac{n}{3})) \leqslant cn^2 log(n)$
$\frac{2}{3}(n^2 log(\frac{n}{3})) \leqslant cn^2 log(n)$ since $n^2 log(\frac{n}{3}) < f(n)$ we could write this relation:
$\frac{2}{3}(n^2 log(n)) \leqslant cn^2 log(n)$ by identification we would have $c = \frac{2}{3}$
Since the regulariy condition is satisfied we could have
$T(n) = f(n) \Rightarrow T(n) = \theta(n^2 logn)$

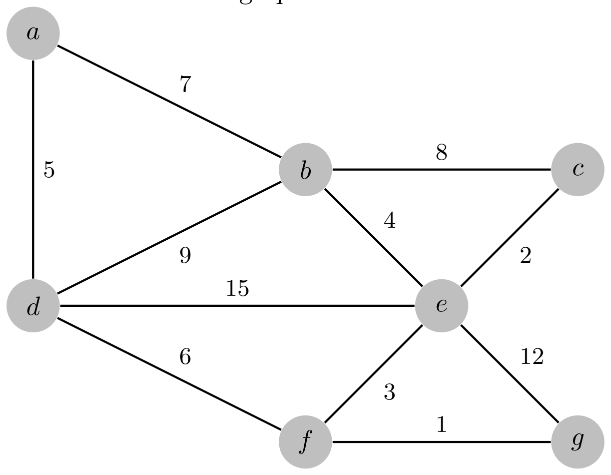
(c) $T(n) = 64(n/8) - n^2 logn$
We have $a = 64$, $b = 8$ and $f(n) = -n^2 logn$
Since f(n) is not positive, it is not possible to use the master theorem.

3 Question3

Let $G = (V, E, W)$ be a weighted connected (undirected) graph, where V is the set of vertices, E is the set of edges, $W_e \in W$ for each $e \in E$ is nonnegative weight of the edge e. A spanning tree is defined as usual, namely $T = (V, T_E)$ where T_E is a subset of E and T is a tree. The width of a spanning tree T is $\{w_e \mid e \in T_E\}$. Thus it is the most "expensive" edge in the tree. T is a Minimum width spanning tree of G if it is a spanning tree and achieves the minimum width among all the spanning trees of G. Show with an example that even if the weights of $G = (V, E, W)$ are all distinct, there could be more than one Minimum width spanning tree.

Proof: Let define a graph $G = (V, E, W)$ be a weighted connected (undirected) graph. We know that the requirement for a spanning tree is the possibility to cover all the vertices without creating a cycle. From that principle, we could construct different spanning trees in G. From G, we could define several spanning trees. Let define T_1, T_2, T_3 and T_4 as the spanning trees in G. Let's define w_1, w_2, w_3 and w_4 the widths of the spanning trees we have previously T_1, T_2, T_3 and T_4 respectively.We could write these relations:

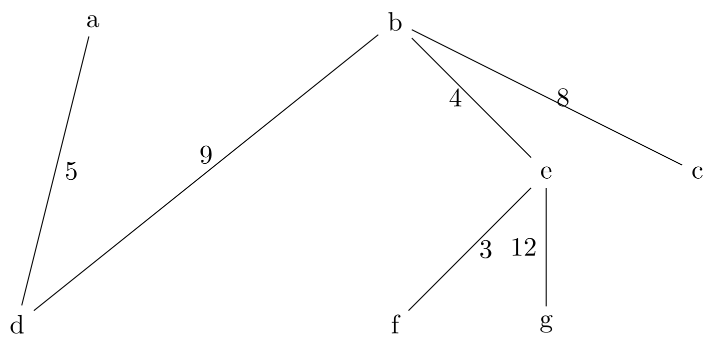
Let define a graph G



$T_1 = (V, T_{1E})$
$T_2 = (V, T_{2E})$
$T_3 = (V, T_{3E})$
$T_4 = (V, T_{4E})$
$\{w_{1e} \mid e \in T_1E\}$
$\{w_{2e} \mid e \in T_{2E}\}$
$\{w_{3e} \mid e \in T_{3E}\}$
$\{w_{4e} \mid e \in T_{4E}\}$

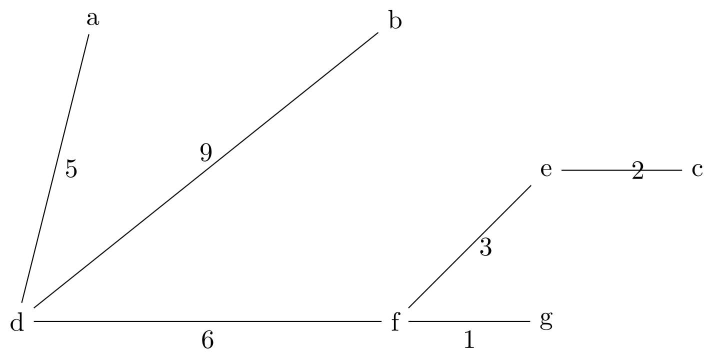
Since we know that the weights in G are all distinct, we could have different widths in the respective trees. Therefore, The set of widths w_1, w_2, w_3 and w_4 would be distinct. As a Result, to find more than one minimum width spanning tree in our case, we will pick two of the smallest widths in the set of trees. Therefore, we can construct the following spanning trees:

This is T_1



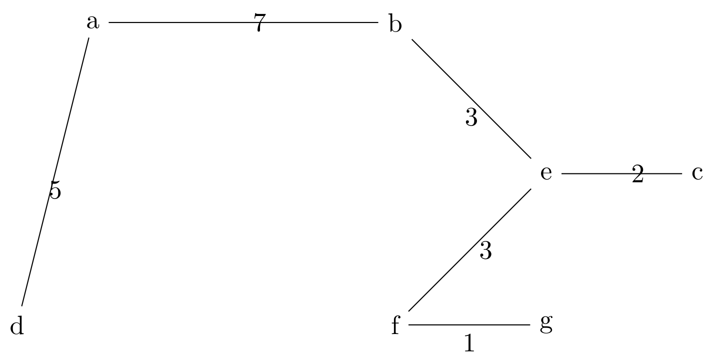
$w_1 = 12$

This is T_2



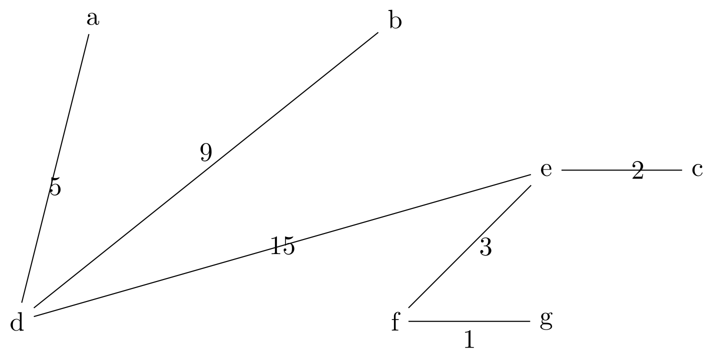
$w_2 = 9$

This is T_3



$w_3 = 7$

This is T_4



$w_4 = 15$

From the above spanning trees, we can select T_2 and T_3 as the minimum width spanning trees. Therefore, we got more than minimum width spanning tree. ■

4 Question4

Assume that the for loop of 5-7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing time for each vertex.

Note: The processes would be attached.

5 Question5

Draw the contraint graph (page 667) for the following system of difference constraints. Use Bellman-Ford algorithm (p 651) to find a feasible solution or determine that no feasible solution exists for the given system.

$x_1 - x_2 \leqslant -2$
$x_1 - x_4 \leqslant -4$
$x_2 - x_3 \leqslant 2$
$x_2 - x_5 \leqslant 7$
$x_4 - x_2 \leqslant 2$
$x_5 - x_1 \leqslant -1$
$x_5 - x_4 \leqslant 3$
$x_6 - x_3 \leqslant -8$

Solving a system of difference constraints:

• A feasible solution for a system of difference constraints can be found by running Bellman-Ford algorithm, on constraint graph.

• If the Bellman-Ford identifies a negative cycle in constraint graph, then it returns FALSE. That is, there is no feasible solution for the given system of difference constraints.

• If the Bellman-Ford algorithm returns TRUE, the shortest path weights of vertices gives a solution.

Making constraint graph from the given system of difference constraints:

• To find the feasible solution in the provided system of differential constraints, first of all we need to construct the constraint graph corresponding to the given system of all we need to construct the constrain graph corresponding to the given system in the equations.

• For each constraint, $a - b \leqslant x$ (where x is some constant), make an edge from a to b and then give that edge weight of x.

• We are going to construct the graph with the above system:

6 Question6

Let f be a flow in a network and let α be a real number. The scalar flow product, denoted αf, is a function from $V \times V$ to R defined by:

$(\alpha f)(u, v) = \alpha f(u, v)$

Prove that the flows in a network from a convex set. That is, show that f_1 and f_2 are flows, then so is $\alpha f_1 + (1 - \alpha) f_2$ for all α in the range $0 \leqslant \alpha \leqslant 1$

Flow: The flow between two vertices is a real valued function. It is denoted by $V \times V \longrightarrow R$.

We have the scalar product that is denoted by αf and defined as follows:

$(\alpha f)(u, v) = \alpha . f(u, v)$

Convex Set: A convex set comprises a set of points in which the line formed by any two points always falls within the set.

Therefore, we need to prove that the flows in a network form a convex set. It is to show that if f_1 and f_2 are flows, then $\alpha f_1 + (1-\alpha)f_2$ is also a flow for α in the range $0 \leqslant \alpha \leqslant 1$.

The total flow between two vertices which is denoted by $V \times V \longrightarrow R$ has to satisfy three properties:

- Capacity constraint
- Skew symmetry
- Flow conversation

Now we are going to demostrate these three properties:

Capacity constraint: The flow in an edge cannot exceed its capactity
$f(u,v) \leqslant c(u,v)$
It is given that $0 \leqslant \alpha \leqslant 0$, therefore, we can have this relation $1-\alpha \geqslant 0$.
Thus, for any edge $(u,v) \in V$, the following can be observed:
$\alpha f_1(u,v) + (1-\alpha)f_2(u,v) \geqslant 0.f_1(u,v) + 0.(1-\alpha)f_2(u,v) \geqslant 0$.
Since we know that the flow in any edge $f(u,v)$ cannot exceed its capacity $c(u,v)$.
We could have these inequations:
$f_1(u,v) \leqslant c(u,v)$ and $f_2(u,v) \leqslant c(u,v)$
As a result we could solve this inequations:

$\alpha f_1(u,v) + (1-\alpha)f_2(u,v) \leqslant \alpha c(u,v) + (1-\alpha)c(u,v)$
$\leqslant \alpha c(u,v) - \alpha c(u,v) + c(u,v)$
$\leqslant c(u,v)$
Therefore, $0 \leqslant \alpha f_1(u,v) + (1-\alpha)f_2(u,v) \leqslant c(u,v)$
We have proven that the capacity property is satisfied.

Skew Symmetry: The flow in an edge is symetric on both sides.
In other words the flow from one vertex to another is the same as the flow in the opposite direction.
For $(u,v) \in V, f(u,v) = -f(v,u)$
So, we have $f_1(u,v) = -f_1(v,u)$ and $f_2(u,v) = -f_2(v,u)$ must hold.
Thus, for any edge

$$v \in V - \{s,t\}, \sum_{v \in V} f_1(v,u) = 0$$

Therefore, we can observe the following:

$$\begin{aligned}\alpha f_1(u,v) + (1-\alpha)f_2(u,v) &= -\alpha f_1(u,v) - (1-\alpha)f_2(u,v)\\ &= -\alpha f_1(u,v) - f_2(u,v) + \alpha f_2(u,v)\\ &= -\alpha f_1(u,v) - f_2(u,v) + \alpha f_2(u,v)\\ &= -(\alpha f_1(u,v) + (1-\alpha)f_2(u,v))\end{aligned}$$

We have proven that the skew symmetry property is satisfied.

Flow Conversation: This property states that except the sourrce s and the sink t, the net flow entering a node v is 0.
For all

$$v \in V - s, t, \sum_{v \in V} f(u, v) = 0$$

Since, f_1 and f_2 are flows. We need to satisfy the flow conversation. Terefore, the following is observed for these two flows:

$$\sum_{u,v \in V-\{s,t\}} f_1(u, v) = 0$$

and

$$\sum_{u,v \in V-\{s,t\}} f_2(u, v) = 0$$

Thus, for this instance of the flow, the net flow at node except source and sink is the following:

$$\sum_{u,v \in V-\{s,t\}} (\alpha f_1(u, v) + (1-\alpha) f_2(u, v)) = \alpha \sum_{u,v \in V-\{s,t\}} f_1(u, v) + (1-\alpha) \sum_{u,v \in V-\{s,t\}} f_2(u, v)$$

Now we can substitute the first equations to get the following equations:

$$\alpha \sum_{u,v \in V-\{s,t\}} f_1(u, v) = \alpha.0 + (1 - \alpha).0 = 0$$

As we could see, the net flow 0; therefore, the property of flow conversation is satisfied.

We have proven that $\alpha f_1 + (1 - \alpha) f_2$ meets the three properties accordingly. Therefore, we could say that f_1 and f_2 are flows, then $\alpha f_1 + (1 - \alpha) f_2$ is also a flow for all α in the range $0 \leqslant \alpha \leqslant 1$. ■

7 Question7

Suppose someone has already computed a maximum flow f for a network flow problem given by a weighted directed graph $G = (V, E, W)$ with a source $s \in V$ and a destination $t \in V$, where every edge has a positive integer weight w_e. Now suddenly it is revealed that in fact the weight value W_e for one particular edge $e = (u, v)$ is wrong, and in fact the correct value is $w_e - 1$. Prove that if the computed maximum flow f has $f_e \leqslant w_e - 1$, then f is still a maximum flow.

Proof:

First of all, we need to consider some conditions. If the flow in (u, v) does not have its capacity saturated, the maximum flow would not be affected because we know that the maximum flow value is the same as to the minimum cut of the graph G and the edge (u, v) would not be part of the minimum cut in this situation. In contrary argumentation, if the the flow f in (u, v) has its capacity saturated then reducing the capacity of the edge (u, v) by 1 may or may not affect the maximum flow. In the case that, (u, v) is in the minimum cut portion, doing $w_e - 1$ would affect the maximum flow.

Considering there is an augmenting path of capacity of at least 1 unit exists from vertex u to v, then reducing the flow by 1 does not affect the max flow because edge (u, v) will not be part of the minimum cut; however, if no path from u to v exist in original flow graph then (u, v) is the minimum cut portion, so the maximum flow will reduce by 1. As a result, f is still the maximum flow.■

8 Question8

(a)

1) Recursuve Algorithm:

Because of recursive calls the recursion grows exponentially; as a result of that, the Big-O complexity is $O(2^n)$

2) Bottom-up Dynamic Programming Algorithm using an Array.
Since the result of previous calculation is saved in an array. It could be the matter of finding those results by scanning the array.
Therefore, the Big-O complexity would be $O(n)$

(b) Use the following dynamic programing algorithm to find $r[8]$ which is the maximum revenue obtainable by cutting up a rod of length 8 and selling the pieces

These are some procedures to show the cutting processes:

$j = 1$
$i = 1$, $p_1 + r_0 = 1 = r_1$

$j = 2$
$i = 1$, $p_1 + r_1 = 2$
$i = 2$, $p_2 + r_0 = 5$

$j = 3$
$i = 1$, $p_1 + r_2 = 1 + 5 = 6$
$i = 2$, $p_2 + r_1 = 5 + 1 = 6$

$i = 3$, $p_3 + r_0 = 8 + 1 = 8$

Note: An additional cutting procedures would be attached.