Analysis of Algorithms - Final

Philip Warton

December 7, 2020

Problem 1

(a)

∞ - Even in a graph with two vertices and an edge between them, DFS will infinitely add this edge to the queue going in the opposite direction.

(b)

$O(V + E)$

(c)

$O(V + E)$

(d)

$O(V + E)$

(e)

∞ - If there is some directed cycle, DFS will indefinitely traverse it.

Problem 2

See attached drawings.

Problem 3

(a)

Can be Prim's, if each the center vertex is chosen first then this edge will be the cheapest. Can also be Boruvka's, if this is the first edge that the for loop encounters the two components will be connected immediately.

(b)

Can obviously be Kruskal's, as it is the first 3 cheapest edges. Can also be Boruvka's, if these are the first 3 edges our for loop encounters then they will be immediately added to the graph.

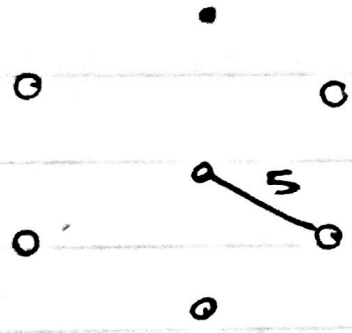
(c)

This can only be Boruvka's, since it connects each vertex to something exactly once, this would be a valid iteration of Boruvka's algorithm. Since it is not connected, it is not Prim's and it is not Kruskal's since the edge with weight 3 was not chosen yet.

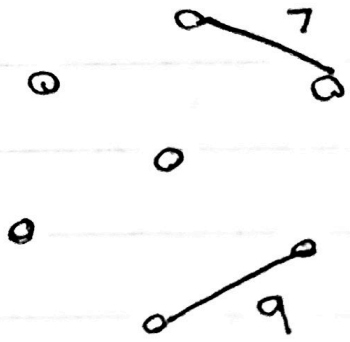
(d)

None. Clearly Prim's and Kruskal's can be ruled out immediately. Boruvka's algorithm can also be ruled out since it would connect the center vertex to something before it would connect two size 2 components.

(a)



(b)



Problem 4

(a)

The weight of the minimum spanning tree will require $n-1$ edges, and since each edge has a weight of 1, this means the weight of this MST will be $n-1$.

(b)

The weight is $2\frac{1}{2} + (n-3) = n-2$.

(c)

We have options of

$$\frac{5}{2} + (n-6)$$
$$\frac{4}{2} + (n-5)$$
$$\frac{3}{2} + (n-4)$$

It is impossible to achieve any other weights.

Problem 5

(a)

Unknown. Famously unproven if $P = NP$ or $P \neq NP$.

(b)

True.

(c)

True

(d)

True

(e)

False

(f)

False

(g)

True

(h)

True

Problem 6

(a)

Assume that we have some algorithm $Dijkstra(G, a, b)$ that returns the shortest path from $a \to b$ in some graph G with positive edges. Then take our negative edge $(u, v) \in E$ and remove it so we have the graph $G \setminus (u, v)$. We compute three lengths, $x = Dijkstra(G \setminus (u, v), a, b)$, $y = Dijkstra(G \setminus (u, v), a, u)$ and $z = Dijkstra(G \setminus (u, v), v, b)$. Then we say that the minimum length is either the shortest path passing through this negative edge, or the shortest path avoiding it. So

$$\text{Shortest Path} = \min\{x, y + l(u, v) + z\}$$

For the running time, we can find and remove the negative edge in $O(E)$ time. Then we can run Dijkstra's algorithm 3 times in, $3 \cdot O(E \log V) = O(E \log V)$ time. Then to choose the minimum takes constant time. This gives us $O(E \log V)$ running time in total.

(b)

We do something similar with this scenario, only we have many more possible shortest paths. Remove both negative edges, $(u, v), (a, b) \in E$. Let $G' = G \setminus \{(u, v), (a, b)\}$. Then we say let

$$\begin{aligned}
x_1 &= Dijkstra(G', s, t) \\
x_2 &= Dijkstra(G', s, u) \\
x_3 &= Dijkstra(G', s, a) \\
x_4 &= Dijkstra(G', v, t) \\
x_5 &= Dijkstra(G', b, t) \\
x_6 &= Dijkstra(G', v, a) \\
x_7 &= Dijkstra(G', b, u)
\end{aligned}$$

Then we say let

$$x = \min\{x_1, x_2 + l(u, v) + x_4, x_3 + l(a, b) + x_5, x_2 + l(u, v) + x_6 + l(a, b) + x_5, x_3 + l(a, b) + x_7 + l(u, v) + x_4\}$$

This value x will be our shortest path. Call $e_1 = (u, v), e_2 = (a, b)$. We are checking each shortest path that passes through any ordering, permutation, or subset of $\{e_1, e_2\}$. Our runtime will be the same as for part (a) since we are simply removing negative edges in $O(E)$ time and then using Dijkstra's some constant number of times based on input size. So we have a running time of $O(E \log V)$.

Problem 7

AIS is NP-Complete.

Proof. AIS $\in$ NP Suppose we have a graph G and we know that it yields some AIS of size k, and suppose that we have this AIS, call it $S \subset V$. Then we can clearly check in $O(E)$ time if more than 1 edges have both endpoints in S with a simple for loop.

AIS is NP-Hard To show this, we wish to show that the $3 - SAT$ problem can be reduced to this AIS problem. Take some $3 - CNF$ string $(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6) \wedge \cdots$. Then we say that this has k-clauses. We use $f : 3CNF \to 3CNF$ where

$$f((x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6) \wedge \cdots) = (a \vee b \vee c) \wedge (\overline{a} \vee \overline{b} \vee \overline{c}) \wedge (x_1 \vee x_2 \vee x_3) \wedge \cdots$$

That is, we append some unsatisfiable string to the begining of our input. Then we graph each clause as a complete graph with 3 vertices and 3 edges. For any conflicting variables (i.e. x and $\overline{x}$), we add an edge between them. If the circuit is satisfiable, then it is possible to choose k disjoint vertices from our last k node triangle subgraphs (that is, exclude the first two) See Figure 1. For further details on why this is the case see the proof for the the independent set algorithm being NP-Hard. Then if we must choose two more vertices, if they are chosen within the first two subgraphs representing $(a \vee b \vee c) \cap (\overline{a} \vee \overline{b} \vee \overline{c})$ then it is guaranteed that exactly one edge will lie within our set, making it almost independent, or an AIS (so it is always possible). Thus if we can produce a $k + 2$ size AIS within this graph G then we have constructed the circuit is satisfiable. Otherwise the circuit is unsatisfiable

Having shown both NP-Hardness and AIS $\in$ NP, it follows that AIS is NP-Complete. $\square$

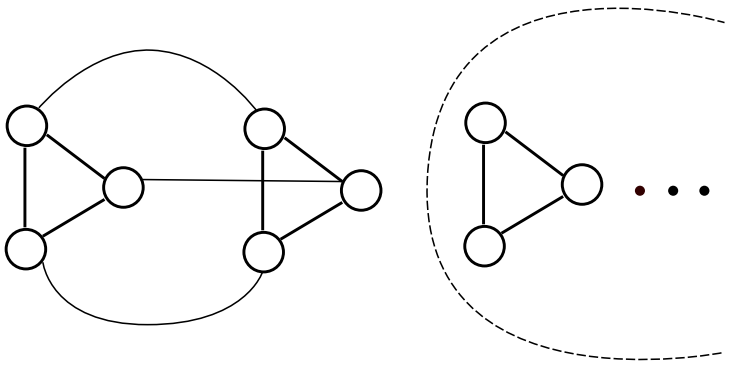


Figure 1: Illustration of Reduction