CSORW4231_001_2022_1 - ANALYSIS OF ALGORITHMS I hw4

Benjamin Jenney

TOTAL POINTS

107.5 / 110

QUESTION 1

Problem 1 35 pts

1.1 Part (a) **2.5 / 5**

1.2 Part (b) **30 / 30**

QUESTION 2

2 Problem 2 **30 / 30**

- Should look for u-s and t-v path using BFS on the residual graph

QUESTION 3

Problem 3 40 pts

3.1 Part (a) **10 / 10**

3.2 Part (b) **30 / 30**

QUESTION 4

4 Extra Credit **5 / 5**

Analysis of Algorithms
HW 4

bej2117

April 2022

1 Problem 1

1.1 Part A

Set $k = floor(|S|/2)$ or $floor((|S|+1)/2)$. depending on whether $|S|$ is odd or even. A formal proof of correctness for this derives from the correctness of select (it's the same thing) described in CLRS. Informally, assuming correctness of the algorithm provided, if we give k = 1, we get the 1st smallest element, k = 2 we get the 2nd, k = 3 the 3rd, ..., k = floor(n/2) the n/2 smallest (the median), ..., k = n the nth smallest element.

1.2 Part B

We we are given the bounds that $n(3/4)^{j+1} \leq |z_{i,j}| \leq n(3/4)^j$
Observing *kth order statistic* we can clearly see that we only stand to cut out 1/4 of the problem statement if we choose a_i such that $|S|/4 \leq i \leq |S|(3/4)$.

Let E_1 be the event that we choose a_i s.t $i < |S|/4$, and let E_2 be the event that we choose a_i s.t $i > |S|(3/4)$. Since both events are mutually exclusive $Pr[E_1] + Pr[E_2] = 1/2$. Thus the expectation on the number of recursions to get a good a_i is 2: it's essentially a coin flip. The work outside of the for loop on any recursion is $O(1)$, and in the worst case we have a total of $O(n)$ work outside the for loop for all recursions.

Since the for loop works for each element in S, and for a problem of type j the size of the input is upper bounded by $n(3/4)^j$ the expected runtime for one problem of type j is 2 * the work $n(3/4)^j$ on any given recursion. Therefore the expected total runtime is $\Sigma_{j=1} 2n(3/4)^j = 2n\Sigma_{j=1}(3/4)^j = 2n(\frac{1}{1-\frac{3}{4}}) = 8n$, Therefore, excluding recursive calls, we get an upper bound $8cn$ for some constant c thus *kth order statistic* has an expected upper bound runtime of $O(n)$.

1.1 Part (a) **2.5 / 5**

Analysis of Algorithms
HW 4

bej2117

April 2022

1 Problem 1

1.1 Part A

Set $k = floor(|S|/2)$ or $floor((|S|+1)/2)$. depending on whether $|S|$ is odd or even. A formal proof of correctness for this derives from the correctness of select (it's the same thing) described in CLRS. Informally, assuming correctness of the algorithm provided, if we give k = 1, we get the 1st smallest element, k = 2 we get the 2nd, k = 3 the 3rd, ..., k = floor(n/2) the n/2 smallest (the median), ..., k = n the nth smallest element.

1.2 Part B

We we are given the bounds that $n(3/4)^{j+1} \leq |z_{i,j}| \leq n(3/4)^j$
Observing *kth order statistic* we can clearly see that we only stand to cut out 1/4 of the problem statement if we choose a_i such that $|S|/4 \leq i \leq |S|(3/4)$.

Let E_1 be the event that we choose a_i s.t $i < |S|/4$, and let E_2 be the event that we choose a_i s.t $i > |S|(3/4)$. Since both events are mutually exclusive $Pr[E_1] + Pr[E_2] = 1/2$. Thus the expectation on the number of recursions to get a good a_i is 2: it's essentially a coin flip. The work outside of the for loop on any recursion is $O(1)$, and in the worst case we have a total of $O(n)$ work outside the for loop for all recursions.

Since the for loop works for each element in S, and for a problem of type j the size of the input is upper bounded by $n(3/4)^j$ the expected runtime for one problem of type j is 2 * the work $n(3/4)^j$ on any given recursion. Therefore the expected total runtime is $\Sigma_{j=1} 2n(3/4)^j = 2n\Sigma_{j=1}(3/4)^j = 2n(\frac{1}{1-\frac{3}{4}}) = 8n$, Therefore, excluding recursive calls, we get an upper bound $8cn$ for some constant c thus *kth order statistic* has an expected upper bound runtime of $O(n)$.

1.2 Part (b) **30 / 30**

2 Problem 2

Insight
Imagine we had a network with all edges at capacity. The residual graph would have nothing but backward edges. Now imagine someone told you they were going to reduce the capacity of one of the edges by one. You would have to take *preventative measures* and reduce the flow on that path from s to t before the capacity was reduced else you'd get a burst pipe. The residual path would reflect this as there would now be forward edges from s to t on this reduced flow path and these forward edges would have a flow of 1 and a backward flow of c(e)-1 for all e on that path, i.e, we would have an augmentable path before the reduction of $c(e)$, after the reduction we know that there was nothing more to be done and our flow has been reduced. However, this 'all edges at capacity case' (let's call it H) is a special one: once the capacity on e was reduced the max flow value would also be reduced by one. In the general case we can't be certain that if Alice chose an edge in any arbitrary flow network it would necessarily reduce the overall flow.

Still, thinking about H gives us a special insight I believe: that increasing a forward edge in the residual reflects increasing the available space on the edge in the network flow, and reducing the backward edge similarly reflects less flow can come out of the edge. Which is to say that if we increase the forward edges, and decrease the backward edges in the residual graph that reflect the edges of an s-t path in the network flow we will have effectively reduced the flow on that s-t path. So, if for forward in G^{f^*} we add 1 and for backward we reduce 1 for each edge on the path that includes e in G_{f^*} we will have successfully diverted flow away from the problem edge, thus no broken pipes. We can then leverage this transformed residual graph, G'_{f^*} to let Ford-Fulkerson do its magic in $O(m)$ time.

Pseudocode
Input: G', G_{f^*}
Output: The readjusted max flow f.
1. Check if $f^*(e) = c_e$, if so continue to step 2, else return f.

2. If e is saturated, run a modified BFS that finds the $x - s$ path in the residual graph: $BFS(G_{f^*}, x, s)$, precisely while exploring u and visiting v check (u, v) for a forward edge and increase it by one, if the forward edge does not exist add it to G_{f^*} with value of 1 (if G_{f^*} is an adjacency list this is just a simple append, thus $O(1)$), similarly check (v, u) and decrease it by one.

3. Run $BFS(G_{f^*}, t, y)$, which is the same modified BFS in step 2, to search a $t - y$ path in the residual graph, and just as in step two, decrease backward edges by 1 and increase forward edges by one adding forward edges as need be. (Notice that if u, v do not share a backward edge they have no flow going into them in G and do not contribute to the flow).

4. Now Run $Ford - Fulkerson(G_{f^*}, f^*)$ with the only modification being that instead of initializing f to zeros we initialize f to be equivalent to f^* and return f.

• **Correctness**
step 1 If $f^*(e) < c(e)$ assuming positive capacities, then clearly we can reduce the capacity of e without having to change the network flow, and besides a flow of 0 on a capacity of 0 clearly contributes nothing to the flow. *step 2 and 3* Correctness of the steps largely derives from the correctness of BFS as discussed in class and in slides. What modifications we have made are trivial $O(1)$ that do not rely on modifying any of the components of the generic BFS from lecture and slides. Notice we are effectively moving backward through an s-t path, this is because e is saturated, and thus there must exist a simple s-t path including e that pushes flow into e thus there must be backward edges on the path. If there wasn't we would have broken skew symmetry: somewhere on the path there would be a node passing off less then it received, or receiving less then its neighbor passed into it.
step 4 correctness is derived from the correctness of $Ford - Fulkerson$ as discussed in class. If an augment path is found we will add one to the $|f|$ and $|f| = |f^*|$, if not $|f| = |f^*| - 1$, in either case f will be a max flow for the graph transformed by Alice.

Time
Since we are leveraging f^* and the residual graph G_{f^*} by way of correctness of Ford-Fulkerson we will only run Ford-Fulkerson once: the resulting $|f|$ is at most one less then $|f^*|$ and $|f^*|$ =a min cut in G. Notice that if there turns out to be more augmenting paths related to e (the edge Alice chose) and there exists an augmenting path, after augmenting once e will become saturated and we will have blocked all other paths since its capacity has been reduced by at most 1, this is to be expected since we have only reduced the flow by one. As for the BFS calls in 2,3 we have already proven in lecture and in slides that BFS is $O(n + m)$ thus we maintain the bound as asked. The total time is $O(n + m) + O(n + m) + O(m) = O(n + m)$, or you could say $\Theta(n + m)$ in the worst case.

3 Problem 3

3.1 Part 1

Assuming we have a feasible circulation then supply constraints must hold. Expanding (ii) to regard edges we derive $\Sigma_{v \in V} s(v) = \Sigma_{(u,v) \in V} f(u, v) - \Sigma_{(v,w) \in V} f(v, w)$. As we can see v appears twice and therefore the two terms will cancel out. Since this is the case for all edges in the feasible circulation we have $\Sigma_{v \in V} s(v) = 0$.

2 Problem 2 **30 / 30**

- Should look for u-s and t-v path using BFS on the residual graph

4. Now Run $Ford - Fulkerson(G_{f^*}, f^*)$ with the only modification being that instead of initializing f to zeros we initialize f to be equivalent to f^* and return f.

• **Correctness**
step 1 If $f^*(e) < c(e)$ assuming positive capacities, then clearly we can reduce the capacity of e without having to change the network flow, and besides a flow of 0 on a capacity of 0 clearly contributes nothing to the flow. *step 2 and 3* Correctness of the steps largely derives from the correctness of BFS as discussed in class and in slides. What modifications we have made are trivial $O(1)$ that do not rely on modifying any of the components of the generic BFS from lecture and slides. Notice we are effectively moving backward through an s-t path, this is because e is saturated, and thus there must exist a simple s-t path including e that pushes flow into e thus there must be backward edges on the path. If there wasn't we would have broken skew symmetry: somewhere on the path there would be a node passing off less then it received, or receiving less then its neighbor passed into it.
step 4 correctness is derived from the correctness of $Ford - Fulkerson$ as discussed in class. If an augment path is found we will add one to the $|f|$ and $|f| = |f^*|$, if not $|f| = |f^*| - 1$, in either case f will be a max flow for the graph transformed by Alice.

Time
Since we are leveraging f^* and the residual graph G_{f^*} by way of correctness of Ford-Fulkerson we will only run Ford-Fulkerson once: the resulting $|f|$ is at most one less then $|f^*|$ and $|f^*|$ =a min cut in G. Notice that if there turns out to be more augmenting paths related to e (the edge Alice chose) and there exists an augmenting path, after augmenting once e will become saturated and we will have blocked all other paths since its capacity has been reduced by at most 1, this is to be expected since we have only reduced the flow by one. As for the BFS calls in 2,3 we have already proven in lecture and in slides that BFS is $O(n + m)$ thus we maintain the bound as asked. The total time is $O(n + m) + O(n + m) + O(m) = O(n + m)$, or you could say $\Theta(n + m)$ in the worst case.

3 Problem 3

3.1 Part 1

Assuming we have a feasible circulation then supply constraints must hold. Expanding (ii) to regard edges we derive $\Sigma_{v \in V} s(v) = \Sigma_{(u,v) \in V} f(u, v) - \Sigma_{(v,w) \in V} f(v, w)$. As we can see v appears twice and therefore the two terms will cancel out. Since this is the case for all edges in the feasible circulation we have $\Sigma_{v \in V} s(v) = 0$.

Now applying this to the original equation we get: (.) $\Sigma_{v \in V} s(v) = 0$.

Since nodes that are sources have $s(v) > 0$ and nodes that are sinks have $s(v) < 0$ we know that all other nodes have $s(v) = 0$. As such nodes in $V - \{S, T\}$ do not contribute to the sum. Therefore we can derive an expression equivalent to (.):

$$\Sigma_{v \in S} s(v) = -\Sigma_{v \in T} s(v)$$

. By way of equivalency, this implies that supply leaving S must be equal to the amount received by T, and the amount received by T must be equivalent to the amount leaving S. As such a circulation is feasible if and only if

$$\Sigma_{v \in S} s(v) = -\Sigma_{v \in T} s(v)$$

.

3.2 Part 2

(a) Give the inputs to the two problems.

supply network input: $G = (V, E, s(v), c_e)$ where for each edge in E there is an associated capacity c_e and for each node in V there is a supply value $s(v)$
Max Flow input: $G' = (V', E', s(v), c'_e)$

(b) Describe in English the reduction transformation and argue that it requires polynomial time. (You do not need to give pseudocode.)

G and G' should be equivalent in the sense that if there is a feasible supply circulation in G there is a max flow in G'.

Constructing G':

1. We add two nodes s', t' to V this new set of nodes will be V' belonging to G'. Besides these new nodes, V' resembles V in every way including the sources in S and the sinks in T.

2. For each node in $G'.S$, we attach an edge *from* s' *to* $v \in S$, and for each node in $G'.T$ we add and edge *from* $v \in T$ *to* t'. Let these new set of edges be E'. Besides these new edges E' resembles E in every way.

3. while creating the (s', v) and (v, t') edges in E' set the capacities c'_e equal to $s(v)$ for each $v \in S$ and $-s(v)$ for each $v \in T$ respectively.

Claim: The reduction transformation G to G′ is polynomial time

Besides the editions of s', t', edges from s' to $v \in T$, and edges from $v \in T$ to t' with these updated capacities, the construction of G' is identical to G. Clearly

3.1 Part (a) **10 / 10**

Now applying this to the original equation we get: (.) $\Sigma_{v \in V} s(v) = 0$.

Since nodes that are sources have $s(v) > 0$ and nodes that are sinks have $s(v) < 0$ we know that all other nodes have $s(v) = 0$. As such nodes in $V - \{S, T\}$ do not contribute to the sum. Therefore we can derive an expression equivalent to (.):

$$\Sigma_{v \in S} s(v) = -\Sigma_{v \in T} s(v)$$

. By way of equivalency, this implies that supply leaving S must be equal to the amount received by T, and the amount received by T must be equivalent to the amount leaving S. As such a circulation is feasible if and only if

$$\Sigma_{v \in S} s(v) = -\Sigma_{v \in T} s(v)$$

.

3.2 Part 2

(a) Give the inputs to the two problems.

supply network input: $G = (V, E, s(v), c_e)$ where for each edge in E there is an associated capacity c_e and for each node in V there is a supply value $s(v)$
Max Flow input: $G' = (V', E', s(v), c'_e)$

(b) Describe in English the reduction transformation and argue that it requires polynomial time. (You do not need to give pseudocode.)

G and G' should be equivalent in the sense that if there is a feasible supply circulation in G there is a max flow in G'.

Constructing G':

1. We add two nodes s', t' to V this new set of nodes will be V' belonging to G'. Besides these new nodes, V' resembles V in every way including the sources in S and the sinks in T.

2. For each node in $G'.S$, we attach an edge *from* s' *to* $v \in S$, and for each node in $G'.T$ we add and edge *from* $v \in T$ *to* t'. Let these new set of edges be E'. Besides these new edges E' resembles E in every way.

3. while creating the (s', v) and (v, t') edges in E' set the capacities c'_e equal to $s(v)$ for each $v \in S$ and $-s(v)$ for each $v \in T$ respectively.

Claim: The reduction transformation G to G′ is polynomial time

Besides the editions of s', t', edges from s' to $v \in T$, and edges from $v \in T$ to t' with these updated capacities, the construction of G' is identical to G. Clearly

this requires iterating through V and E at most 1 times. Step 2 and 3 can be combined into one loop that iterates at most $(|S| + |T|) \leq |V|$ times and the creation of a new edge with updated capacity c'_e takes constant time. Therefore it is clear that the transformation from G to G' can be done in polynomial time. In fact, though I do not exactly show it here, it's not hard to figure for oneself that it can be done in $\Theta(n + m)$ time in the worst case. ■

(c) Prove carefully equivalence of the original and the reduced instances. Letting k denote the necessary condition for a feasible circulation found in Part 1. we make the claim that *There is a feasible supply circulation if and only if* $|f'| = k$

($\Rightarrow$) Given any feasible circulation in G we can construct a flow f' in G' with $|f'| = k$ by saturating all the edges out of s'.

1. If we make a cut $(\{s'\}, \{V' - \{s'\})$ we see that the sum of capacities along the cut $= \Sigma_{v \in S} s(v)$.
2. If we saturate each edge out of s' we have $\Sigma_{e \text{out of} s'} c'_e = \Sigma_{v \in S} s(v)$.
3. If we make a cut $(\{V' - \{t'\}, \{t'\})$ we see that the sum of capacities along the cut $= -\Sigma_{v \in T} s(v)$.
4. If we saturate each edge out of t' we have $\Sigma_{e \text{into} t'} c'_e = -\Sigma_{v \in T} s(v)$.

$f^{out}(v) - f^{in}(v) = s(v)$, for non source/sinks in the original circulation we have $f^{out}(v) - f^{in}(v) = 0$ which implies: (*) $f^{out}(v) = f^{in}(v)$. By way of construction in step 3, we have c'_e equal to $s(v)$ for each $v \in S$ and $-s(v)$ for each $v \in T$ respectively. As such in the transformed graph (*) also applies to the nodes in S and T in the original graph. Therefore 1,3 the sum of capacities for cuts in 1 and 3 equal the max flow value, and if we saturate as described in 2, we will be pushing a max flow consequently this proves t receives a max flow as well and as consequence we prove 2, and 4. As such we prove the forward direction.

($\Leftarrow$) Given an integral flow f' in G' where $|f'| = k$ we have a feasible circulation in G. Since $|f'| = k$ it must be the case that every edge out of s' and into t' are saturated. As a result if we removed the both s' and t' and there incident edges, we will be left with a graph identical to G. Which is to say that for all nodes v in G $f^{out}(v) - f^{in}(v) = s(v)$. Thus we have proved the backward direction.
■

3.2 Part (b) **30 / 30**

4 Extra Credit **5 / 5**