University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour, M. Fuchs, A. Malyusz

Algorithm Theory
Sample Solution Exercise Sheet 7

Due: Friday, 8th of December, 2023, 10:00 am

Exercise 1: Fibonacci Heap - Amortized *(6 Points)*

Suppose we "simplify" Fibonacci heaps such that we do *not* mark any nodes that have lost a child and consequentially also do *not* cut marked parents of a node that needs to be cut out due to a `decrease-key`-operation. Is the *amortized* running time

(a) ... of the `decrease-key`-operation still $\mathcal{O}(1)$? *(2 Points)*

(b) ... of the `delete-min`-operation still $\mathcal{O}(\log n)$? *(4 Points)*

Explain your answers.

Sample Solution

Two reasonable answers would be as follows.

(a) Yes. Not having to cut all your marked ancestor nodes only makes `decrease-key` faster. In fact each individual `decrease-key` operation has now runtime $\mathcal{O}(1)$. *(2 Points)*

(b) No. The reason is that we loose the recursive property that a given node with rank i has i children that have at least ranks $i-2, i-3, ...$, respectively. This was required to show that each tree of a given rank has a *minimum size* of F_{i+2} (where F is the Fibonacci series) which grows exponential in i. Consequentially the maximum rank can not be too large, just $\mathcal{O}(\log n)$, as a tree with higher rank would require more than n nodes.

Now, if a node can loose an arbitrary number of children without being cut, the above property can not be guaranteed anymore. In particular, in extreme cases we could end up with a tree with rank $n-1$. Since `delete-min` has amortized runtime linear in the maximum rank, it will have a higher amortized running time (i.e., $\omega(\log n)$). *(4 Points)*

Exercise 2: Cuts and Flows *(14 Points)*

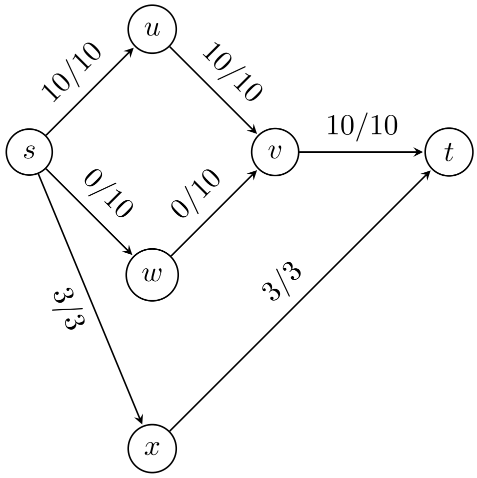
Note that the following three tasks are independent of each other.

(a) Given a flow-network G with nonnegative integer capacities on edges, we call an edge $e \in E$ saturated if its flow value equals its capacity, i.e., if $f(e) = c_e$. Prove or disprove the following statements.

(1) If an edge e is crossing a minimum s-t-cut of G, any execution of the Ford-Fulkerson algorithm will saturate e. *(2 Points)*

(2) Given some maximum flow of G that saturates edge e, then e is crossing at least one minimum s-t cut of G. *(2 Points)*

(b) Let the figure below represent a flow-network G with positive integer capacities on edges and a maximum flow f^*, where we denote the flow value f^* and the capacity c by f^*/c on the corresponding edge.



In the lecture we have seen that given a maximum flow, one can compute a minimum s-t cut $(A^*, V \setminus A^*)$, where A^* is the set of nodes that can be reached from s on a path with positive residual capacities in the residual graph. Give the cut $(A^*, V \setminus A^*)$ in G. *(3 Points)*

(c) Given a flow-network G with a source s, sink t, and nonnegative integer capacities on edges.

(1) Consider a minimum s-t cut $(S, V \setminus S)$ of G. Prove that $(S, V \setminus S)$ is not unique *if and only if* there exists an edge e crossing the cut $(S, V \setminus S)$ such that after increasing the capacity of e by 1, the capacity of the new minimum s-t cut is the same as the capacity of the old minimum s-t cut. *(5 Points)*

Remark: A minimum s-t cut $(S, V \setminus S)$ in G is said to be unique if and only if the capacity of the cut $(S, V \setminus S)$ is strictly less than the capacity of any other s-t cut $(F, V \setminus F)$ in G.

(2) Give a polynomial-time algorithm to decide whether G has a unique minimum s-t cut or not. *(2 Points)*

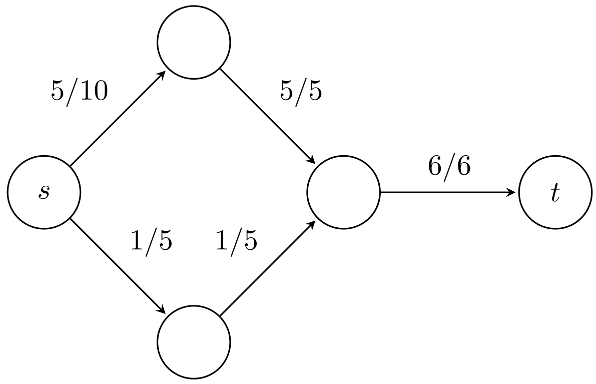
Sample Solution

(a) (1) Let $(S, V \setminus S)$ be any min cut and $|f^*|$ be the max-flow value (computed by Ford-Fulkerson).

If we are only considering the edges e in the min s-t cut that are going from S to $V \setminus S$, then the statement is *true*. Indeed, due to the min-cut-max-flow theorem, we have $|f^*| = c(S, V \setminus S)$. And if e in this cut wouldn't be saturated, then we would have that
$|f^*| = |f^{*out}(S)| - |f^{*in}(S)| \leq |f^{*out}(S)| < c(S, V \setminus S) = |f^*|$. Contradiction.

Otherwise, if we want to also consider the cut edges e going from $V \setminus S$ to S, then the statement is *false*. Indeed, $|f^{*in}(S)|$ must be equal to zero, otherwise if $|f^{*in}(S)| \neq 0$, then we get $|f^*| < |f^{*out}(S)|$, and according to what we just proved in the previous paragraph we have that $|f^{*out}(S)| = c(S, V \setminus S) = |f^*|$, which leads us to a contradiction. Thus all cut edges e going from $V \setminus S$ to S must have flow 0, thus they are not saturated (we can say that because wlog we can also assume the edge capacities are positive integers).

(2) False. The edge with $f(e) = c_e = 5$ is saturated but not part of any min cut. (Note that since the max flow is 6 and there is only one cut with capacity 6, this is the only min cut in the given network.)



(b) $A^* = \{s, w, v, u\}$.

(c) (1) Let C be some min cut of G, prove that $C := (S, V \setminus S)$ is not unique if and only if the capacity of the new minimum cut after increasing some edge in C by 1 is the same.
Forward direction: Let $C' \neq C$ be another minimum cut in G (C is not unique), hence, there must exist $e \in C \setminus C'$, otherwise $C = C'$, which is a contradiction. Moreover, if this e was the edge that we increased its capacity by 1, then the capacity of the new minimum cut must be the same as the old one in particular C' is a still a minimum cut in the new graph, since if there exists another cut in the new graph that has a capacity less than C' it will be a better minimum cut than C in the original graph, which is a contradiction. Thus, the forward direction is true.
Backward direction: Assume C is unique. Let $e \in C$ be the edge that we choose for increasing its capacity by 1, and hence the capacity of the new minimum cut stays the same. This means that there must exist a new cut $C' \neq C$ such that $e \in C \cap C$ and the cut size of C is the same as C'. But this implies that if we restore the original capacity of e, then C' is a cut in G with capacity strictly less than that of C, which is a contradiction. Thus, the backward direction is true.

(2) The algorithm first runs the Ford-Fulkerson algorithm, computes the residual edges at the same time, and finally stores the max flow value which is equal to the min cut value. Then it finds the set A^* by running e.g. a BFS algorithm over positive residual capacity edges starting from s. We know from the minimum cut-max flow constructive proof that $C := (A^*, V \setminus A^*)$ is a minimum cut, then for every edge e crossing we do the following: We increase its capacity value by 1, then run Ford-Fulkerson another time in the new graph and compare the new max flow with the old capacity value of C, and according to the iff statement in the previous part: if both values are the same, then we know that C not unique and stop, else we continue by checking all edges crossing C and if for each edge the capacity of the new min cut was more than the old one by 1, then we know that C is unique.
This can be done in polynomial in n time (since $|C| \leq m$, a variant of Ford-Fulkerson like Edmonds-Karp can be used to achieve a polynomial in n running time, and the BFS will also take poly in n).