Algorithmic Discrete Mathematics 6. Exercise Sheet

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Mathematics
PD Dr. Ulf Lorenz
Dipl.-Math. David Meffert

SS 2012
27. and 28. Juni 2012
Version of June 21, 2012

Groupwork

Exercise G1 (Heap-Sort)
Use Heap-Sort with a min-heap to sort the array $(3,1,4,1,5,9,2)$.

Solution:

First step is to contruct the heap. This can be done by using 'buildheap' or just inserting all numbers and use 'heapify' after each insertion. So we get a heap where the minimal element is the root. We extract this root, put it in a new array and put the element on the right bottom of the heap to the root. Now the heap property will in general be violated. Therefore we use heapify to regain that property. The 'big' root element sinks down into the heap. Now the smallest element of the remainig elements is the root of the heap and the procedure start from the beginning. We do this till our heap is empty. Then our array is sorted.

For this special list the initial heap is of the form $(1,1,2,3,5,9,4)$. (First: root, second: left child of root, third: right child root, fourth: left child of left child of root,...) We extract the element 1 and 4 becomes the new root, so we get the remaining heap $(4,1,2,3,5,9)$. We swap 4 and 1 and 4 and 3 by using heapify and get to $(1,3,2,4,5,9)$. The rest of the sorting process works the same way as before.

Exercise G2 (Maximal flows)
Let $D=(V,E)$ be a directed graph with source s and sink t. The capacities on the edges should be natural numbers and the maximal flow x from s to t is given. Now we want to modify the capacity of one edge in the graph by

(a) increasing it by 1,

(b) decreasing it by 1.

Give an algorithm that calculates the maximal flow from s to t in that changed graph in $O(|V|+|E|)$.

Solution:

(a) If we increase the capacity of one edge by 1 then we just need to verify if there is an augmenting path in the changed graph. Since we only increased one edge be 1 the maximal flow can only increase by 1. Checking if there is a augmenting path can be done by using DFS which is $O(|V|+|E|)$.

(b) If we decrease the capacity of one edge by 1 there can be two different cases.
1.case: The flow is still valid and does not exceed any capacity on a edge in the graph. Then the maximal flow just stays the same.
2.case: The flow exceeds the capacity on the edge we decreased the capcity of 1. We denote this edge by (u,v). Now let P_1 be path from s to u and P_2 be a path from v to t. This paths should be disjoint and only contain edges with a positive actual flow. Such paths do exist, since otherwise there would be no flow on the edge (u,v) and can be found by using DFS again. So the runtime is $O(|V|+|E|)$ for finding both of these paths. Now reduce the flow on the whole path $P_1(u,v)P_2$ by 1, so the general flow for the graph is valid again, with a maximal flow reduced by 1. It could be the case now that this is not the maximal flow, but we know that the maximal flow can only be one flow unit bigger than the actual flow. So we just look for another augmenting path as in part (a) and get the maximal flow for the changed graph. This can be done in $O(|V|+|E|)$. Since we did 3 algorithm steps with $O(|V|+|E|)$ the whole procedure is in $O(|V|+|E|)$.

Exercise G3 (Mengers Theorem)
(a) A directed graph $D=(V,E)$ is called *strongly k-connected*, if for every pair of vertices (u,v) and every set of edges $B\subseteq E$ with $|B|\leq k-1$, there exists a directed path from u to v in $D'=(V,E\setminus B)$. Prove the edge version of Mengers

Theorem.
Mengers Theorem for edges: A directed graph $D = (V, E)$ is strongly k-connected, iff for every pair of vertices (s, t) there exist at least k directed paths from s to t which share no edges.

(b) A directed graph $D = (V, E)$ is called *k-vertex-connected*, if for every pair of vertices (u, v) and every set of vertices $W \subseteq V$ with $|W| \leq k - 1$ there exists a directed path from u to v in $D - W$. Prove the vertex version of Mengers Theorem.
Mengers Theorem for vertices: A directed graph $D = (V, E)$ is k-vertex-connected, iff for every pair of vertices (s, t) there exist at least k directed paths from s to t sharing no vertices.

(c) Do Mengers Theorems also hold for undirected graphs?

Solution:

(a) $\Rightarrow$: We show that D is strongly k-connected implies the existance of the k directed paths having disjoint edges first. We do it by showing the following equivalent statement:
There exist vertices s, t such that there are at most $k-1$ directed paths from s to t having disjoint edges $\Rightarrow$ D is not strongly k-connected.
So let (s, t) be such a pair of vertices as described above with at most $k - 1$ directed paths. We set the capacity of all edges to 1 and look for flows from s to t. The maximal flow we get by this procedure can be intepreted as the number of directed paths from s to t having disjoint edges. Now we use the Max-Flow-Min-Cut Theorem to get a cut $(S, \bar{S})$. The value of that cut is at most $k - 1$, since the maximal flow can be at most $k - 1$. Now we set B as the set of all edges in D connecting S and $\bar{S}$. By the previous argumentation $|B| \leq k - 1$. Now we delete all edges in the set B from D. By the cut property of $(S, \bar{S})$ there can't be any more directed paths from s to t in $(V, E \setminus B)$. Hence D is not strongly k-connected.
$\Leftarrow$: We show that the existance of k directed paths from s to t for arbitary vertices s, t yields that D is strongly k-connected.
So let D be such a graph and $s, t \in V$. For any subset $B \subset E$ with $|B| \leq k - 1$ we can only delete edges of at most $k - 1$ directed paths having disjoint edges. So there remains at least one more directed path from (s, t) which did not use any of the edges in B. So the graph D is strongly k-connected.

(b) $\Rightarrow$: We want to show that for every k-vertex-connected graph D and vertices $s, t \in V$ there exist a least k directed paths from s to t sharing no vertices. Again we show the following equivalent statement:
If there exist vertices $s, t \in V$ where we don't have k directed paths from s to t sharing no vertices, then D is not k-vertex-connected.
So let $s, t \in V$ be such vertices, where we have a most $k-1$ directed paths from s to t sharing no vertices. As in part (a) we want to use the Max-Flow-Min-Cut Theorem. Therefore we double each vertex $i \in V$ and name the vertices i_1 and i_2. All edges which where directed to i now are directed to i_1 and all edges which started in i now start in i_2. Additionally we add edges from i_1 to i_2 with capacity 1. All other edges get capacity ∞.We call this graph D'. Now we determine the maximal flow f from s_2 to t_1 in D'. This flow equals the number of directed paths from s to t in the orginal graph, which share no vertices. Therefore we introduced the doubling of all the vertices. By the Min-Cut-Max-Flow Theorem we find a cut $(S, \bar{S})$ in D' with value f. Now let U be the set of all edges in the minimal cut $(S, \bar{S})$. This set U contains an edge (i_1, i_2) for every path from s_2 to t_1 and all these edges correspond to a vertex i in the orginal graph D. So deleting all edges of B in D' is like deleting the corresponding vertices in D. We call this vertex set W. Since by assumption $|B| \leq k - 1$ we get $|W| \leq k - 1$, so we may delete this vertex set in order to look for k-vertex-connectivity. By construction there is no more directed path from s_2 to t_1 in $D' \setminus B$. By translating this to the vertex setting we conclude that there exists no more directed path from s to t in our graph $D \setminus W$. This means our graph D is not k-vertex-connected.
$\Leftarrow$: Let D be a directed graph where for each pair of vertices $s, t \in V$ there exist at least k directed paths from s to t, sharing no vertices. So we delete any vertex set $W \subset V$ from D with $|W| \leq k - 1$. In the worst case we deleted a vertex from$k - 1$ different paths from s to t. Since all those paths do no share vertices there must be at least one more path where we did not delete any vertex from. So there is at least one directed path in $D - W$. This means D is k-vertex-connected.

(c) We did not use that our graph we worked with is directed, so the proofs work the same way for undirected graphs. We just have to substitute directed with undirected at every place there.

Homework

Exercise H16 (10 points)
Look at the following variant of the quicksort algorithm.
Algorithm QuickSort(a,l,r)

Input: An array a of length n with $a[i] \in \mathbb{Z}$, lower and upper bounds l, r with $1 \leq l \leq r \leq n$.
Output: The array a with $a[l] \leq a[l+1] \leq \ldots \leq a[r]$.

(1) Set $i = l - 1$ and $j = r$.

(2) `While` $i < j$ `Do`

(3) `Do` i = i + 1 `While` $a[i] \leq a[r]$.

(4) `Do` j = j - 1 `While` ($a[j] \geq a[r]$ and $j \geq i$).

(5) `If` $j > i$ `Then` Swap $a[j]$ and $a[i]$.

(6) `End While`

(7) Swap $a[i]$ and $a[r]$.

(8) `If` $l < i - 1$ `Then` QuickSort $(a, l, i-1)$.

(9) `If` $i + 1 < r$ `Then` QuickSort $(a, i+1, r)$.

(10) return a.

The iteration starts by using QuickSort(a,1,*length*(a)).

(a) Use Quicksort to sort the number array $(12, 3, 8, 13, 5, 2, 9, 4, 5, 3, 7)$.

(b) How does a number array have to look like, so that Quicksort has a runtime of $O(n^2)/O(n \log n)$. Describe it in general and give an example for both cases by using the first 10 natural numbers.

Solution:

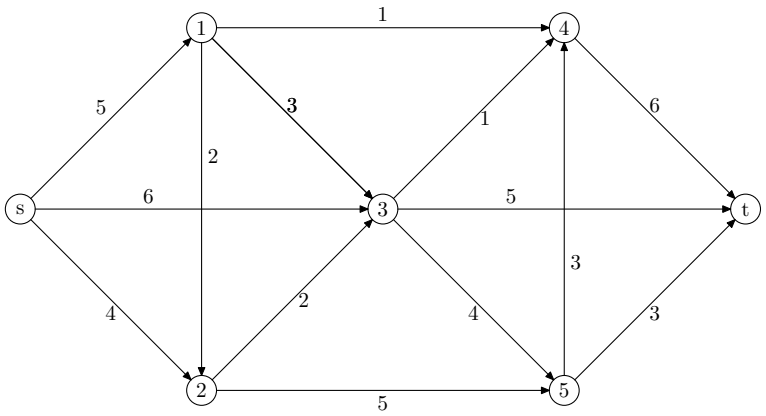
(a) We give a description what happens while using Quicksort. In the first run we swap 12 and 3, 8 and 5, 13 and 4 and finally 9 and 7. Then 7 is already at the right place in the array. At that point in the algorithm the array is given by $(3, 3, 5, 4, 5, 2, \mathbf{7}, 13, 8, 12, 9)$. Now we have to sort the remaining left and ride side of the number 7. We do it for the left side. The right side works the same way. The array $(3, 3, 5, 4, 5, \mathbf{2})$ becomes $(\mathbf{2}, 3, 5, 4, 5, 3)$ Now 2 is a the right place. The next step we swap 5 and 3. So $(3, 5, 4, 5, \mathbf{3})$ becomes $(3, \mathbf{3}, 4, 5, 5)$. The array (3) with one element is already sorted and the other array $(4, 5, 5)$ still needs three Quicksort steps where always the element on the right side of the array is deleted till the element 4 is the last one left. In those last steps no more swaps are involved.

After doing the same thing with the right part $(13, 8, 12, 9)$ we get the list $(2, 3, 3, 4, 5, 5, 7, 8, 9, 12, 13)$.

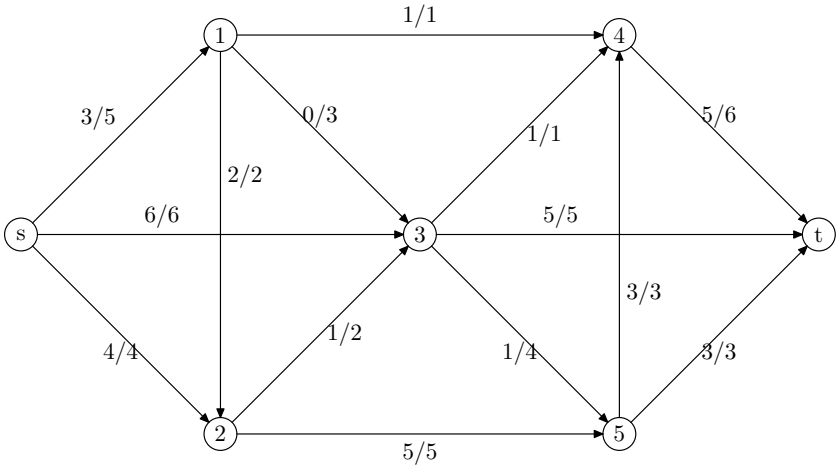
(b) The worst case for Quicksort, such that is has runtime $O(n^2)$ is a already sorted list. This way the element on the right hand side always stays there and Quicksort is used with a list with only one less element than before. So we need $n - 1$ Quicksort iterations. An example would be $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$.

If we want a runtime of $O(n \log n)$ for Quicksort we have to split the list of numbers in half every time. This way we only have to do half of comparisons of the step before. An example is $(1, 2, 4, 3, 8, 6, 7, 10, 9, 5)$.

Exercise H17 (Finding maximal flows) (10 points)
Find the maximal flow from s to t in the following directed graph and prove its maximality.



Solution: The maximal flow is given by 13. One can proof that by the minimal cut which consists of $S_1 = \{s, 1, 2, 3, 5\}$ and $S_2 = \{4, t\}$. Another way to do it is to use the Ford-Fulkerson algorithm and find augmenting paths till there are no more left. One possible solution is:



Exercise H18 (Right or Wrong?) (10 points)
Let $D = (V, E)$ be a directed graph with integer capacities C_e for the edges $e \in E$. Let s be the source in this graph and t the sink. Which of the following statements are right? Which are wrong? Give a proof or a counterexample.

(a) If all capacities are even, then there exists a maximal flow f from s to t, such that $f(e)$ is even for all edges $e \in E$.

(b) If all capacities are odd, then there exists a maximal flow f from s to t, such that $f(e)$ is odd for all edges $e \in E$.

(c) If f is a maximal flow from s to t, then either $f(e) = 0$ or $f(e) = c_e$ holds for all edges $e \in E$.

(d) There exists a maximal flow from s to t, such that either $f(e) = 0$ or $f(e) = c_e$ holds for all edges $e \in E$.

(e) If all capacities on the edges are different, then the minimal cut is unique.

(f) If we multiply each capacity with the positive real number $\lambda \in \mathbb{R}_+$, then every minimal cut in there original graph is a minimal cut in the modified graph.

(g) If we add the positive number $\lambda \in \mathbb{R}_+$ to each capacity, then every minimal cut in the orginal graph is a minimal cut in the modified graph.

Solution:

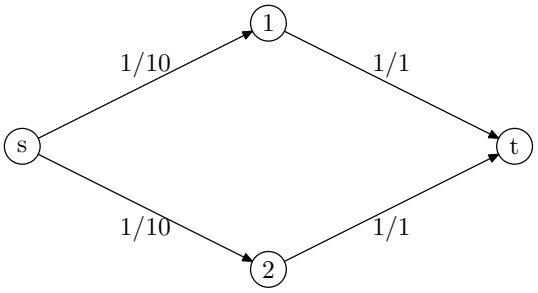


Figure 1: graph 1

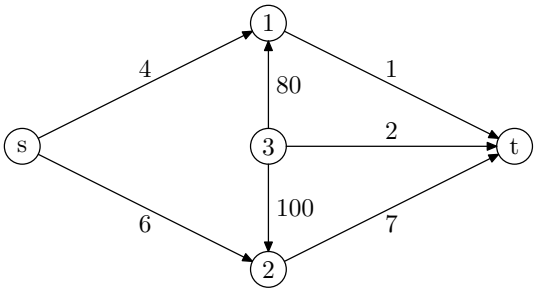


Figure 2: graph 2

(a) This statement is true. At the beginning the flow $f = 0$ is even. Every time we find an augmenting path it adds an even flow to all edges in that path since all capacities are even. The resulting graph after that augmenting process still has even capacities since we substracted only even numbers from even numbers. So by induction there is a maximal flow such that all flows on the edges are even.

(b) This statement is wrong. Look at the graph D with $V = \{s, 1, 2, u, t\}$, $E = \{(s,1),(s,2),(1,u),(2,u),(u,t)\}$. The capacities for those edges should be 1 except for the edge (u,t) with capacity 3. So all capacities are odd , but the maximal is 2 with $f_{(u,t)} = 2$.

(c) This statement is wrong, see graph 1 for a counterexample.

(d) This statement is wrong again, see Graph 1 for counterexample, since there only exists one possible maximal flow graph which does not fulfil the needed condition.

(e) This statement is wrong, see graph 2 for a counterexample. A minimal cut has value 10 but there are two cuts namely $(\{s\}, \{1,2,3,t\})$ and $(\{s,1,2,3\}, \{t\})$ having that value.

(f) This statement is right. We denote the value of a cut W by $c(\delta^+(W))$. This means we sum up the values of all edges going from vertices in W to $\bar{W} = V \setminus W$. Let S be a minimal cut, so we have

$$c(\delta^+(S)) \leq c(\delta^+(W))$$

for all cuts W. In the changend graph $(D = (V,E), \hat{c})$ with $\lambda \in \mathbb{R}_+$ we get

$$\hat{c}(\delta^+(S)) = \lambda \cdot c(\delta^+(S)) \leq \lambda \cdot c(\delta^+(W)) = \hat{c}(\delta^+(W))$$

for all cuts W. So S is still a minimal cut in the graph $(D = (V,E), \hat{c})$.

(g) This statement is wrong, as we can see be looking at graph 2 again. As we have seen there a two cuts $(\{s\}, \{1,2,3,t\})$ and $(\{s,1,2,3\}, \{t\})$ with value 10. If we add for example 2 to every edge then $(\{s\}, \{1,2,3,t\})$ would still be minimal with value 14. The cut $(\{s,1,2,3\}, \{t\})$ would have value 16 and therefore would not be minimal anymore.