ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Final Exam January 16, 2013

Algorithms

January 16, 2013

- You are only allowed to have an $A4$ page written on both sides.
  (Vous avez droit à un formulaire manuscrit sur une page $A4$ recto-verso.)
- Calculators, cell phones, computers, etc... are not allowed.
  (Aucun matériel électronique (calculatrices, téléphones portables, ordinateurs, etc) n'est permis.)

Last name :

First name :

Section :

Exercise 1 / 5 points	Exercise 2 / 5 points	Exercise 3 / 10 points	Exercise 4 / 20 points	Exercise 5 / 20 points	Exercise 6 / 20 points

Exercise 7 / 20 points

Total / 100

Problem 1 [5 points]. Consider the natural implementation of the three sorting algorithms (Insertion-Sort, Merge-Sort, Heap-Sort) that we studied in class. Which one of these implementations is *not* "in-place"?

Solution. Merge-sort.

Problem 2 [5 points]. Simplify and arrange the following functions in increasing order according to asymptotic growth.

$$3^N, \sqrt{4^N}, \log^2 N, \sqrt{N}, N^2, \log N, 20N$$

Solution.

- First the logs: $\log N, \log^2 N$
- Second the polynomials: $\sqrt{N}, 20N, N^2$
- Last the exponential functions: $\sqrt{4^N} = 2^N, 3^N$

Problem 3 [10 points]. Let $f(n)$ and $g(n)$ be the functions defined for positive integers as follows:

Function $f(n)$**:**
1: $ans \leftarrow 0$
2: **for** $i = 1, 2, \ldots, n-1$ **do**
3: **for** $j = 1, 2, \ldots, n-i$ **do**
4: $ans \leftarrow ans + 1$
5: **end for**
6: **end for**
7: **return** ans

Function $g(n)$**:**
1: **if** $n = 1$ **then**
2: **return** 1
3: **else**
4: **return** $g(\lfloor n/2 \rfloor) + g(\lfloor n/2 \rfloor)$
5: **end if**

1. What is, in Θ notation, the running time of these algorithms, given that addition runs in time $\Theta(1)$? *(6 points)*
2. What is, in Θ notation, the running time of algorithm $g(n)$ if we at line 4 replace $g(\lfloor n/2 \rfloor) + g(\lfloor n/2 \rfloor)$ by $2g(\lfloor n/2 \rfloor)$? *(4 points)*

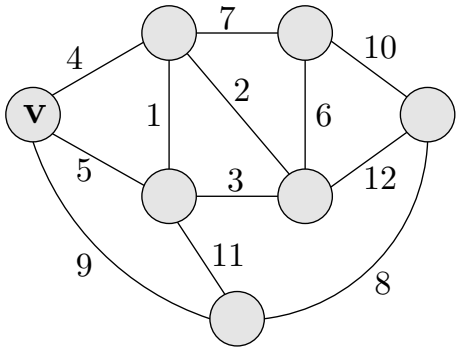
   (Assume that multiplication runs in time $\Theta(1)$.)

Solution.

1. Running time of $f(n)$ is $\Theta(n^2)$. The running time of $g(n)$ is proportional to the recursion $T(n) = 2T(n/2) + 1$ and $T(2^x) = 2^{x+1} - 1$. Therefore the running time of $g(n)$ is $\Theta(n)$.
2. Now the running time is proportional to the recursion $T(n) = T(n/2) + 1$ and $T(2^x) = x + 1$. Therefore the running time of the modified version is $\Theta(\log n)$

Problem 4 [20 points]. **Spanning Trees**

Consider the following minimum spanning tree instance, i.e., an undirected connected graph with weights on the edges.



1. Write the weights of the edges of the minimum spanning tree in the order they are added by Prim's algorithm starting from vertex v. *(5 points)*

2. Write the weights of the edges of the minimum spanning tree in the order they are added by Kruskal's algorithm. *(5 points)*

3. A bottleneck edge is an edge of highest weight in a spanning tree. A spanning tree is a minimum bottleneck spanning tree if the graph does not contain a spanning tree with a smaller bottleneck edge weight. *Show that a minimum spanning tree is also a minimum bottleneck spanning tree.* *(10 points)*

Solution.

1. 4, 1, 2, 6, 9, 8

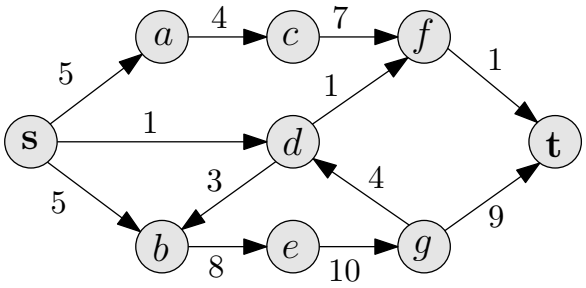
2. 1, 2, 4, 6, 8, 9

3. Suppose that there exists a minimum spanning tree T that is not a minimum bottleneck spanning tree. Identify the edge e in T of maximum weight and remove it. This leads to a graph with two components and there must exist an edge e' that connects them with $w(e') < w(e)$ (take an edge of a minimum bottleneck spanning tree crossing the cut defined by the two components).

   Adding e' yields a new spanning tree of lower cost which contradicts that we started with a minimum spanning tree T.

Problem 5 [20 points]. **Flows and Cuts**

Consider the following flow network:

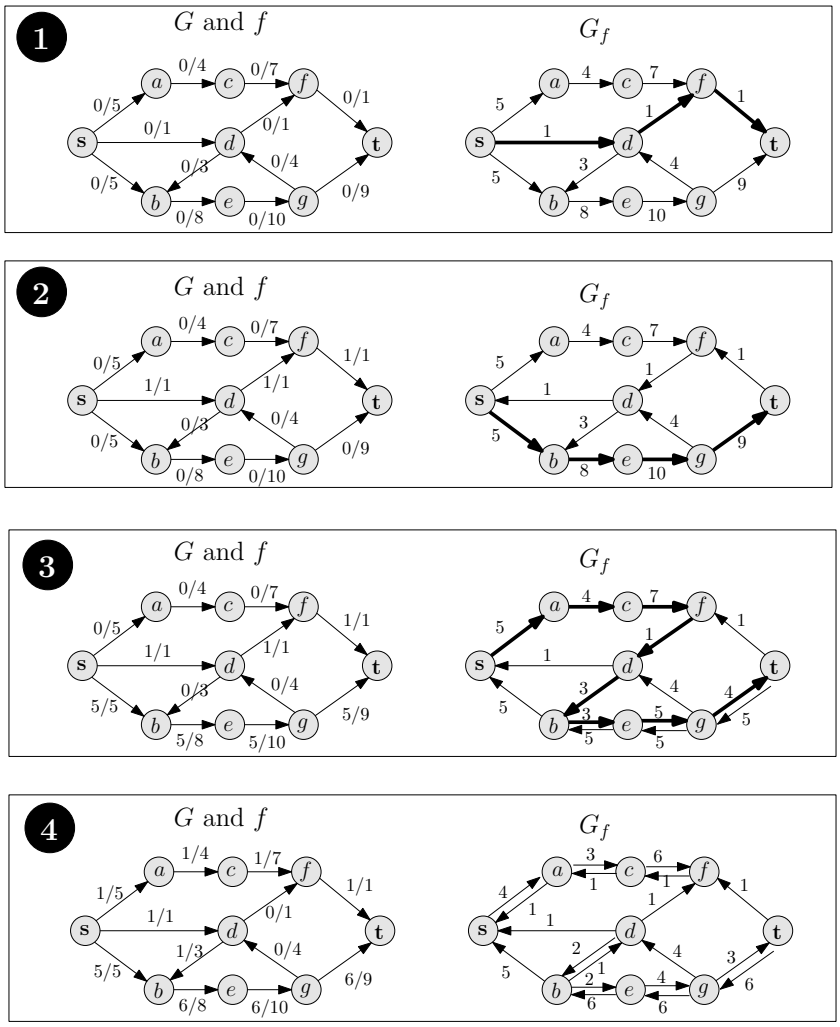


1. Find a max-flow and a min-cut by running the Ford-Fulkerson algorithm that in each iteration chooses a shortest augmenting path (a shortest path can be found by using BFS). *(15 points)*

   (In each iteration draw the residual network and explain how you found the min-cut.)

2. What is the value of the max-flow? What is the capacity of the min-cut? *(5 points)*

Solution.



1. Min-Cut is the vertices reachable from s in the residual network of the last step, i.e., $\{s, a, c, f\}$.

2. Value of flow and capacity of cut is 7.

Problem 6 [20 points]. Let $x_1...x_m$ and $y_1...y_n$ be two strings. For $0 \leq i \leq m$ and $0 \leq j \leq n$, let $c[i,j]$ be the length of the longest common subsequence of $x_1...x_i$ (or the empty string if $i = 0$) and $y_1...y_j$ (or the empty string if $j = 0$). In other words, $c[i,j]$ is defined to be the maximum k such that there exist indices $1 \leq i_1 < i_2 < \cdots < i_k \leq i$ and $1 \leq j_1 < j_2 < \cdots < j_k \leq j$ satisfying $x_{i_\ell} = y_{j_\ell}$ for all $\ell = 1, \ldots, k$.

1. Complete the recurrence relation for $c[i,j]$ that can be used for dynamic programming:

$$c[i,j] = \begin{cases} \_\_\_\_\_ & \text{if } i = 0 \text{ or } j = 0 \\ \_\_\_\_\_ & \text{if } i,j > 0 \text{ and } x_i = y_j \\ \_\_\_\_\_ & \text{if } i,j > 0 \text{ and } x_i \neq y_i \end{cases} \qquad \textit{(10 points)}$$

2. Use the recurrence relation to return the length of the longest common subsequence of the two strings BABDBA and DACBCBA by filling in the table of $c[i,j]$ values below (in a bottom-up dynamic programming fashion). *(10 points)*

	j	0	1	2	3	4	5	6	7
i		y_j	D	A	C	B	C	B	A
0	x_i								
1	B								
2	A								
3	B								
4	D								
5	B								
6	A								

Solution.

1.

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1,j-1] + 1 & \text{if } i,j > 0 \text{ and } x_i = y_j \\ \max(c[i,j-1], c[i-1,j]) & \text{if } i,j > 0 \text{ and } x_i \neq y_i \end{cases}$$

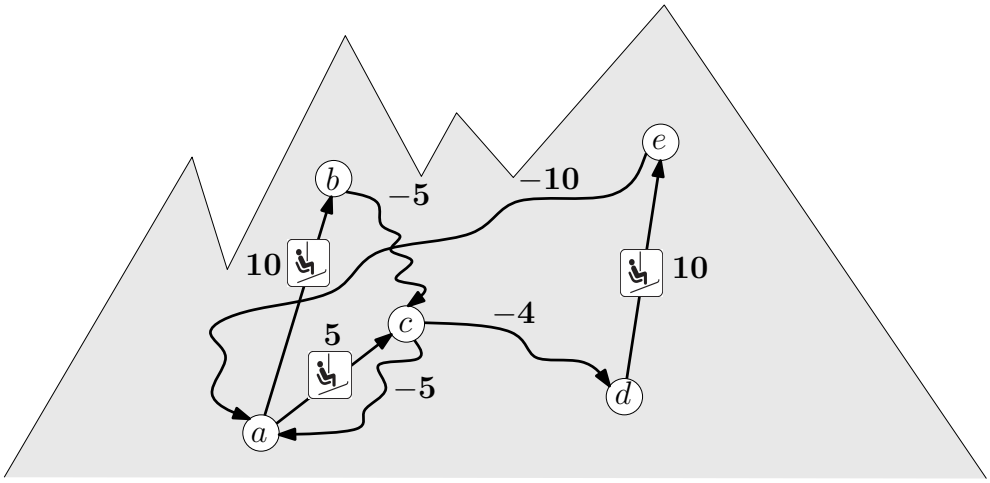
2.

	j	0	1	2	3	4	5	6	7
i		y_j	D	A	C	B	C	B	A
0	x_i	0	0	0	0	0	0	0	0
1	B	0	0	0	0	1	0	1	0
2	A	0	0	1	1	1	1	1	2
3	B	0	0	1	1	2	2	2	2
4	D	0	1	1	1	2	2	2	2
5	B	0	1	1	1	2	2	3	3
6	A	0	1	2	2	2	2	3	4

Longes commong subsequence is 4 (it is ABBA)

Problem 7 [20 points]. The famous alpine ski racer Lindsey Vonn has sent her assistant to measure the altitude differences of the ski lifts and slopes of a famous ski resort in Switzerland. The assistant returns after several days of hard work with a map of all ski lifts and all slopes together with the altitude difference between the start station and the end station of each lift and the altitude difference between the start station and end station of each slope. A slope starts from the end station of a lift and ends at the start station of a potentially different lift (see the figure below).

Lindsey wants to verify the map of her assistant by performing the following sanity check: starting from point A, no matter how we ski (using lifts and slopes), the altitude change when we return to A should be 0 (i.e., neither strictly negative nor strictly positive). Design an algorithm to perform this sanity check and analyze its running time in terms of the number of slopes and ski lifts ($|E|$) and the number of start and end stations ($|V|$). Your algorithm should run in polynomial time (in $|V|$ and $|E|$). *(20 points)*



Solution. First, run Bellman-Ford to detect negative cycles. Second, run Bellman-Ford to detect positive cycles (by inverting edge weights). The running time of Bellman-Ford and therefore our algorithm is $O(|V||E|)$.

Approach that runs in time $O(|V|+|E|)$ using BFS or DFS: First use DFS to detect which vertices from which we can ski to the start destination A. Then on these vertices perform a DFS or BFS and whenever you visit a new vertex while doing BFS or DFS, save its height compared to the starting point. Then, check that each edge has the right height difference.