COMP 3804 — Assignment 3

Due: Thursday March 23, 23:59.

Assignment Policy:

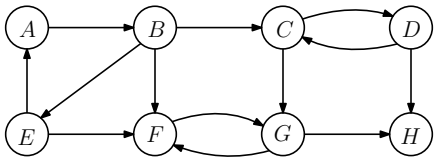
- Your assignment must be submitted as one single PDF file through Brightspace.

> Use the following format to name your file:
>
> `LastName_StudentId_a3.pdf`

- **Late assignments will not be accepted. I will not reply to emails of the type "my internet connection broke down at 23:57" or "my scanner stopped working at 23:58", or "my dog ate my laptop charger".**
- You are encouraged to collaborate on assignments, but at the level of discussion only. When writing your solutions, you must do so in your own words.
- Past experience has shown conclusively that those who do not put adequate effort into the assignments do not learn the material and have a probability near 1 of doing poorly on the exams.
- When writing your solutions, you must follow the guidelines below.
  - You must justify your answers.
  - The answers should be concise, clear and neat.
  - When presenting proofs, every step should be justified.

Question 1: Write your name and student number.

Question 2: Consider the following directed graph:



(2.1) Draw the *DFS*-forest obtained by running algorithm DFS. Classify each edge as a tree edge, forward edge, back edge, or cross edge. In the *DFS*-forest, give the *pre*- and *post*-number of each vertex. Whenever there is a choice of vertices, pick the one that is alphabetically first.

(2.2) Draw the *DFS*-forest obtained by running algorithm DFS. Classify each edge as a tree edge, forward edge, back edge, or cross edge. In the *DFS*-forest, give the *pre*- and *post*-number of each vertex. Whenever there is a choice of vertices, pick the one that is alphabetically last.

Algorithm DFS(G):
for each vertex v
do $visited(v) = false$
endfor;
$clock = 1$;
for each vertex v
do if $visited(v) = false$
  then EXPLORE(v)
  endif
endfor

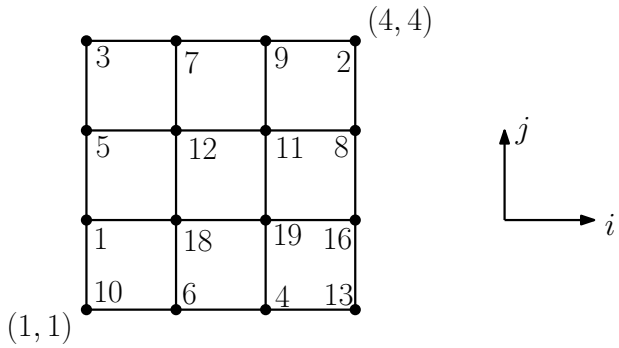
Algorithm EXPLORE(v):
$visited(v) = true$;
$pre(v) = clock$;
$clock = clock + 1$;
for each edge (v, u)
do if $visited(u) = false$
  then EXPLORE(u)
  endif
endfor;
$post(v) = clock$;
$clock = clock + 1$

Question 3: Let $G = (V, E)$ be a directed graph. After algorithm DFS(G) has terminated, each vertex has a *pre*- and *post*-number. Let u and v be two distinct vertices in V. Prove the following:

- If $pre(u) < pre(v) < post(u)$, then there is a directed path in G from u to v.
- Assume that G is directed and acyclic. If $post(u) < pre(v)$, then there is no directed path in G from u to v.

Question 4: Let $n \geq 2$ be an integer and let G be the $n \times n$ undirected grid graph: The vertices of G are the grid points (i, j) for $1 \leq i \leq n$ and $1 \leq j \leq n$. Each vertex (i, j) has neighbors $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$ (provided their coordinates are in the set $\{1, 2, \ldots, n\}$).

Each vertex of this graph stores a number; we assume that all these n^2 numbers are distinct. A vertex is called *awesome*, if the number stored at this vertex is larger than the numbers stored at all of its neighbors. In the example below, $n = 4$ and both vertices $(1, 1)$ and $(3, 2)$ are awesome.



- Prove that there always exists at least one awesome vertex.
- Give an algorithm that finds an awesome vertex in $O(n)$ time. Note that the graph G has n^2 vertices and $\Theta(n^2)$ edges.

Question 5: Let $G = (V, E)$ be a directed acyclic graph that is given to you using adjacency lists. We say that G is *nice* if for every two distinct vertices u and v, there is a directed path from u to v, or there is a directed path from v to u.

Give an algorithm that decides in $O(|V| + |E|)$ time whether G is nice. As always, justify your answer.

Hint: Find some examples of directed acyclic graphs with four vertices that are nice, and find some examples that are not nice. What property of their topological orderings distinguishes them?

Question 6: In class, we have seen a data structure for the UNION − FIND problem that stores each set in a linked list, with the header of the list storing the name and size of the set. Using this data structure, any operation FIND(x) takes $O(1)$ time, whereas any operation UNION(A, B, C) takes $O(\min(|A|, |B|))$ time.

Consider the same data structure, except that the header of each list only stores the name of the set (and not the size). Show that, in this new data structure, any operation FIND(x) can be performed in $O(1)$ time, and any operation UNION(A, B, C) can still be performed in $O(\min(|A|, |B|))$ time.

Question 7: A sequence $(b_1, b_2, \ldots, b_k)$ of numbers is called *amazing*, if there is an index i with $1 \leq i \leq k$, such that $(b_1, b_2, \ldots, b_i)$ is increasing and $(b_i, b_{i+1}, \ldots, b_k)$ is decreasing.

Let $S = (a_1, a_2, \ldots, a_n)$ be a sequence of numbers. Give an algorithm that computes, in $O(n^2)$ time, the length $LAS(S)$ of a longest amazing subsequence of S. The numbers in the subsequence are not necessarily consecutive in S.

For example, if

$$S = (10, 22, 9, 33, 21, 50, 41, 60, 80, 1),$$

then $LAS(S) = 7$, because $(10, 22, 33, 50, 60, 80, 1)$ is a longest amazing subsequence.

As always, argue why your algorithm is correct. You my use any result that was presented in class.

Question 8: After having taken many flights with ZoltanJet, Alma is ready to redeem her frequent flyer points: Alma can choose from a wide selection of beers! Each beer costs a certain number of points. Of course, being a beer connoiseur, each beer has a value to Alma. For example, Heineken has a low value, whereas Minerva Stout has a high value. Which beers does Alma choose?

There are n types $B_1, B_2, \ldots, B_n$ of beer. For each i with $1 \leq i \leq n$,

- it costs p_i points to acquire beer B_i,
- the value of beer B_i is equal to v_i.

Alma has P points to spend. We denote the beers that she chooses by the subset $I \subseteq \{1, 2, \ldots, n\}$ of their indices. (For each i, Alma cannot choose more than one bottle of beer B_i.) For example, choosing beers B_3, B_5, B_9 is denoted by $I = \{3, 5, 9\}$.

Since Alma has P points, we must have

$$\sum_{i \in I} p_i \leq P, \tag{1}$$

i.e., the total cost of all beers chosen is at most P. At the same time, Alma wants to maximize the total value of all chosen beers, i.e., choose I such that

$$\sum_{i \in I} v_i \tag{2}$$

is maximized.

To summarize, the input consists of two sequences $p_1, p_2, \ldots, p_n$ and $v_1, v_2, \ldots, v_n$ of positive integers, and a positive integer P. You may assume that each p_i is at most P. The goal is to compute a subset I of $\{1, 2, \ldots, n\}$ such that (1) is satisfied and the summation in (2) is maximized.

Give a dynamic programming algorithm (in pseudocode) that solves this problem in $O(nP)$ time. As always, argue why your algorithm is correct.

Hint: All input values are positive *integers*. Consider $S(i, j)$, which is the value of an optimal solution if Alma chooses beers from the set $\{B_1, B_2, \ldots, B_i\}$ and she can spend at most j points.