Reading material

At the lecture we will continue with dynamic programming. We will talk about sequence alignment and shortest paths in general graphs. You should read KT section 6.6 and 6.8.

Exercises

[w] means it is a warmup exercise and [$*$] that it is a difficult exercise.

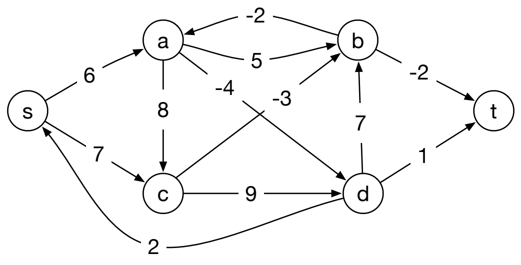
1 Sequence alignment [w] Consider the strings `APPLE` and `PAPE` over the alphabet $\Sigma = \{\texttt{A}, \texttt{E}, \texttt{L}, \texttt{P}\}$ and a penalty matrix P:

	A	E	L	P
A	0	1	3	1
E	1	0	2	1
L	3	2	0	2
P	1	1	2	0

Compute the sequence alignment of the two strings when the penalty for a gap $\delta = 2$. Fill the dynamic programming table below, and explain how the minimum cost sequence alignment is found in it.

	j	0	1	2	3	4	5
i			A	P	P	L	E
0							
1	P						
2	A						
3	P						
4	E						

2 Shortest path[w] Find the shortest path from s to t in the graph below using the Bellman-Ford algorithm.



3 Longest palindrome subsequence A *palindrome* is a (nonempty) string over an alphabet Σ that reads the same forward and backward. For example are abba and racecar palindromes.

A string P is a *subsequence* of string T if we can obtain P from T by removing 0 or more characters in T. For instance, abba is a subsequence of bcadfbbba.

Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string.

To do this first give a recurrence for the problem and then write pseudo code for an algorithm based on your recurrence and dynamic programming. Argue that your recurrence is correct and analyse the running time and space usage of your algorithm.

4 Defending Zion Solve KT 6.8

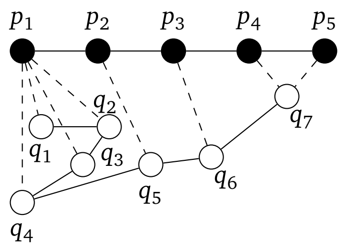
5 Trading Cycles Solve KT 6.13.

6 Discrete Fréchet distance Consider Professor Bille going for a walk with his dog. The professor follows a path of points $p_1, \dots, p_n$ and the dog follows a path of points $q_1, \dots, q_m$. We assume that the walk is partitioned into a number of small steps, where the professor and the dog in each step either both move from p_i tp p_{i+1} and from q_j to q_{j+1}, respectively, or only one of them moves and the other one stays.

The goal is to find the smallest possible length L of the leash, such that the professor and the dog can move from p_1 and q_1, resp., to p_n and q_n. They cannot move backwards, and we only consider the distance between points. The distance L is also known as the discrete Fréchet distance.

We let $L(i, j)$ denote the smallest possible length of the leash, such that the professor and the dog can move from p_1 and q_1 to p_i and q_j, resp. For two points p and q, let $d(p, q)$ denote the distance between the them.

In the example below the dotted lines denote where Professor Bille (black nodes) and the dog (white nodes) are at time 1 to 8. The minimum leash length is $L = d(p_1, q_4)$.



6.1 Give a recursive formula for $L(i, j)$.

6.2 Give pseudo code for an algorithm that computes the length of the shortest possible leash. Analyze space and time usage of your solution.

6.3 Extend your algorithm to print out paths for the professor and the dog. The algorithm must return where the professor and the dog is at each time step. Analyze the time and space usage of your solution.

Puzzle of the week: 100 Ants [1]There are 100 ants on a rod of length 1 metre. The ants are arbitrarily positioned on the rod and are travelling at 1 metre/minute either right or left at the start. The ants are also perfectly elastic, so that if two ants collide they simply turn round and carry on at 1 metre/minute in the opposite direction. If an ant reaches the end of the rod it falls off. The question is, what is the longest time it can take for all 100 ants to fall off the rod?

You can assume the ants are points on the rod and that the rod is simply a one dimensional line (i.e. ants can only go left or right). You are asked to find the longest time it can take for all the ants to fall off the rod over all possible start states and starting directions of the ants.

[1]I got this puzzle from Raphäel Clifford