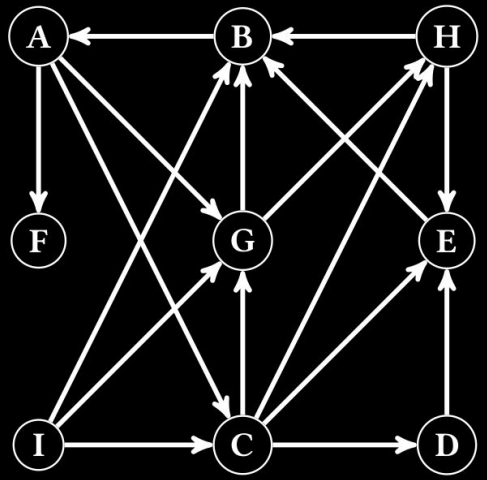
Algorithms and Data Structures

Exercise Session 11

https://n.ethz.ch/~ahmala/and

Exercise 11.1 *Breath-first search* **(1 point)**.

Execute a breadth-first search (Breitensuche) on the following graph[1] starting from vertex A. Use the algorithm presented in the lecture. When processing the neighbors of a vertex, process them in alphabetical order.



(a) For each vertex, give its enter- and leave-number.

(b) Give the distances from A to any vertex.

(c) Give the ordering of the vertices that results from sorting them by enter-number (or equivalently by leave-number).

(d) Draw a scale from 1 to 18 and mark for every vertex v the interval I_v we get from the above execution of breadth-first search. Is it possible that for some vertices $v \neq w$ we have $I_v \subseteq I_w$ (for a general graph G)? Why/Why not?

Dijkstra in Dense Graphs

Dijkstra's algorithm performs n iterations.

On each iteration it selects an unmarked vertex v with the lowest value $d[v]$, marks it and checks all the edges (v, to) attempting to improve the value $d[to]$.

Dijkstra in Dense Graphs (m = n*n)

Dijkstra's algorithm performs n iterations.

On each iteration it selects an unmarked vertex v with the lowest value $d[v]$, marks it and checks all the edges (v, to) attempting to improve the value $d[to]$.

Dijkstra in Dense Graphs (m = n*n)

Vertex search requires O(n)

Dijkstra's algorithm performs n iterations.

On each iteration it selects an unmarked vertex v with the lowest value d[v], marks it and checks all the edges (v, to) attempting to improve the value d[to].

Dijkstra in Dense Graphs (m = n*n)

Vertex search requires O(n)

Dijkstra's algorithm performs n iterations.

On each iteration it selects an unmarked vertex v with the lowest value d[v], marks it and checks all the edges (v, to) attempting to improve the value d[to].

each relaxation can be performed in O(1)

relaxation = insertion = improvement

Dijkstra in Sparse Graphs (m << n*n)

Dijkstra's algorithm performs n iterations.

On each iteration it selects an unmarked vertex v with the lowest value $d[v]$, marks it and checks all the edges (v, to) attempting to improve the value $d[to]$.

Dijkstra in Sparse Graphs (m << n*n)

Vertex search requires O(log n)

Dijkstra's algorithm performs n iterations.

On each iteration it selects an unmarked vertex v with the lowest value d[v], marks it and checks all the edges (v, to) attempting to improve the value d[to].

Dijkstra in Sparse Graphs (m << n*n)

Vertex search requires O(log(n))

Dijkstra's algorithm performs n iterations.

On each iteration it selects an unmarked vertex v with the lowest value d[v], marks it and checks all the edges (v, to) attempting to improve the value d[to].

each relaxation can be performed in O(log(n))

relaxation = insertion = improvement

Dijkstra in Sparse Graphs (m << n*n) + Fibonacci Heap

Vertex search requires O(log(n))

Dijkstra's algorithm performs n iterations.

On each iteration it selects an unmarked vertex v with the lowest value d[v], marks it and checks all the edges (v, to) attempting to improve the value d[to].

each relaxation = insertion = improvement can be performed in O(1)

relaxation = insertion = improvement

Exercise 11.2 *Shortest paths with cheating* **(1 point)**.

Let $G = (V, E)$ be a weighted, directed graph with weights $c : E \to \mathbb{R}_{\geq 0}$. We consider a variation of the shortest path problem in G, where we are allowed to 'cheat' by setting a certain number of weights to 0. Formally, for $k \in \mathbb{N}$, we write C_k for the set of all weight functions $\gamma : E \to \mathbb{R}_{\geq 0}$ on G with $\gamma(e) \neq c(e)$ for at most k edges $e \in E$.[2]

Given $s, t \in V$, we wish to find a path $P = (v_1 = s, v_2, \ldots, v_\ell = t)$ in G which minimizes:

$$c_k(P) := \min_{\gamma \in C_k} \gamma(P), \text{ where } \gamma(P) := \sum_{i=1}^{\ell-1} \gamma\big((v_i, v_{i+1})\big).$$

We call such a path a 'shortest path from s to t with k cheats.'

Recall that a naive implementation of Dijkstra's algorithm finds the length of a shortest path in a weighted graph (without cheating) in time $O(|V|^2)$.

(a) Describe an algorithm which finds the length of a shortest path from s to t with k cheats in time $O(|E|^k \cdot |V|^2)$. Prove that your algorithm is correct, and achieves the desired runtime.

Hint: *Apply Dijkstra's algorithm to G several times, for different weight functions.*

(b) Describe an algorithm which finds the length of a shortest path from s to t with k cheats in time $O\big((k|V|)^2\big)$. Prove that your algorithm is correct, and achieves the desired runtime.

Hint: *Construct a new graph $G' = (V', E')$ whose vertex set V' consists of $k+1$ copies of V. Choose the edges E' and weights c' in a clever way, and apply Dijkstra's algorithm to G'.*

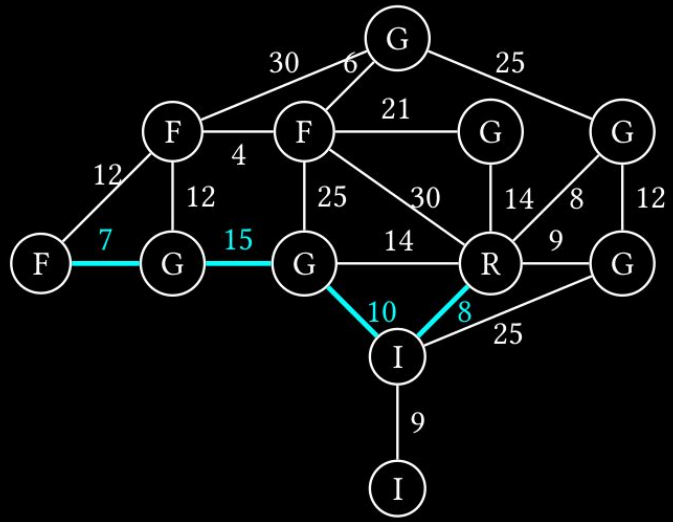
Exercise 11.3 *Language Hiking.*

Alice loves both hiking and learning new languages. Since she moved to Switzerland, she has always wanted to discover all four language regions of the country in a single hike – but she is not sure whether her week of vacation will be sufficient.

You are given a graph $G = (V, E)$ representing the towns of Switzerland. Each vertex V corresponds to a town, and there is an (undirected) edge $\{v_1, v_2\} \in E$ if and only if there exists a direct road going from town v_1 to town v_2. Additionally, there is a function $w : E \to \mathbb{N}$ such that $w(e)$ corresponds to the number of hours needed to hike over road e, and a function $\ell : V \to \{\mathrm{G}, \mathrm{F}, \mathrm{I}, \mathrm{R}\}$ that maps each town to the language that is spoken there[3]. For simplicity, we assume that only one language is spoken in each town.

Alice asks you to find an algorithm that returns the walking duration (in hours) of the shortest hike that goes through at least one town speaking each of the four languages.

For example, consider the following graph, where languages appear on vertices:



The shortest path satisfying the condition is marked in red. It goes through one R vertex, one I vertex, two G vertices and one F vertex. Your algorithm should return the cost of this path, i.e., 40.

(a) Suppose we know the order of languages encountered in the shortest hike. It first goes from an R vertex to an I vertex, then immediately to a G vertex, and reaches an F vertex in the end, after going through zero, one or more additional G vertices. In other terms, the form of the path is RIGF or RIG...GF. In this case, describe an algorithm which finds the shortest path satisfying the condition, and explain its runtime complexity. Your algorithm must have complexity at most $O((|V| + |E|) \log |V|)$.

Hint: *Consider the new vertex set* $V' = V \times \{1, 2, 3, 4\} \cup \{v_s, v_d\}$*, where* v_s *is a 'super source' and* v_d *a 'super destination' vertex.*

(b) Now we don't make the assumption in (a). Describe an algorithm which finds the shortest path satisfying the condition. Briefly explain your approach and the resulting runtime complexity. To obtain full points, your algorithm must have complexity at most $O((|V| + |E|) \log |V|)$.

Hint: *Consider the new vertex set* $V' = V \times \{0, 1\}^4 \cup \{v_s, v_d\}$*, where* v_s *is a 'super source' and* v_d *a 'super destination' vertex.*

Exercise 11.4 *Driving from Zurich to Geneva* **(1 point)**.

Bob is currently in Zurich and wants to visit his friend that lives in Geneva. He wants to travel there by car and wants to use only highways. His goal is to get to Geneva as cheap as possible. He has a map of the cities in Europe and which ones are connected by highways (in both directions). For each highway connecting two cities he knows how much fuel he will need for this part (depending on the length, condition of the road, speed limit, etc.) and how much this will cost him. This cost might be different depending on the direction in which he travels. Furthermore, for some connections between two cities, he has the option to take a passenger with him that will pay him a certain amount of money. Again this might be different depending on the direction he travels. We assume that this option is only availabe to him between cities directly connected by a highway and that the passengers want to travel the direct road and would not agree to making a detour. Also Bob has a small car, so he can only take at most one passenger with him. It is possible that he gains more money from this than he has to pay for the fuel between two given cities but we assume that he has no way to gain an infinite amount of money, i.e. there is no round-trip from any city that earns him money.

(a) Model the problem as a graph problem such that you can directly apply one of the algorithms in the lecture, without modifications to the algorithm:

   (1) Describe your graph. What are the vertices, what are the edges and the weights of the edges?

   (2) What is the graph problem that we are trying to solve?

   (3) Solve the problem using an algorithm discussed in the lecture (without modification).

(b) Now we change the problem slightly. Bob got a list from his friend of certain highways that are in a bad condition. To not damage his car, he decided that he want to use at most one of these highways. Again, model the problem as a graph problem such that you can directly apply one of the algorithms in the lecture, without modifications to the algorithm:

   (1) Describe your graph. What are the vertices, what are the edges and the weights of the edges?

   (2) What is the graph problem that we are trying to solve?

   (3) Solve the problem using an algorithm discussed in the lecture (without modification).

Exercise 11.5 *Ancient Kingdom of Macedon.*

The ancient Kingdom of Macedon had n cities and m roads connecting them, such that from one city, you can reach all other $n - 1$ cities. All roads were *roman roads*, i.e. stone-paved roads that did not require any maintenance, and no two roads were of the same length. With the technological developments in the Roman Kingdom, a new type of carriage was developed, called the *Tesla Carriage*, which was much faster than all the alternatives in the Ancient Macedon Kingdom. However, the Tesla Carriage required *asphalt roads* to operate, and such roads had to be maintained every year, or otherwise the asphalt would wear off, rendering the road unusable as if it was a roman road.

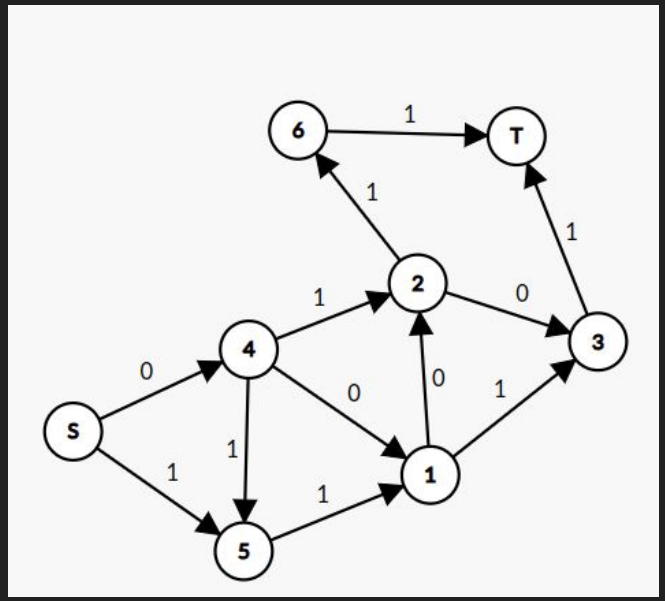
With the effort to modernize the kingdom, Phillip II promised the Ancient Macedonians that he will provide them with asphalt roads by paving some of the existing roman roads, such that every two cities can be reached with a Tesla Carriage. The price to pave a roman road or maintain an asphalt road is equal, and is proportional to the length of the road. To save money Phillip II decided to pave sufficient roman roads to fulfill his promise, while minimizing the overall yearly maintenance price.

Even in the first years, the new Tesla Carriages improved the lives of the average Ancient Macedonians, but at the same time, they also provided means for robbers to commit crimes and escape to another city. To resolve this, Phillip II decided to create checkpoints the second year, one at each asphalt road. Each of the checkpoint will have a fixed cost for both building and maintenance.

Assuming a fixed price k for each checkpoint, does Phillip II have to consider paving new roman roads, or he can maintain the same set of roads in order to make sure that the overall maintenance price of the roads and the checkpoints is still minimal? Prove your reasoning, or provide a counterexample.

Note: For simplicity, assume that the roman roads were paved all at once, on the first day of the year, and maintenance will be done the same day next year, again all at once. Also assume that checkpoints can also be built at once for all roads, as well a they can be maintained all at once in a day.

Shortest Path in a Binary Graph



Peer Grading

Exercise 11.1

You will find the file to peergrade in your polybox folder

While emailing your peer grading to me please include the group you corrected their work in cc.

https://docs.google.com/spreadsheets/d/1owPsJsd9THBWInwFcVjKCc0f_r6n4pGwwDKMDdwaCjM/edit?usp=sharing