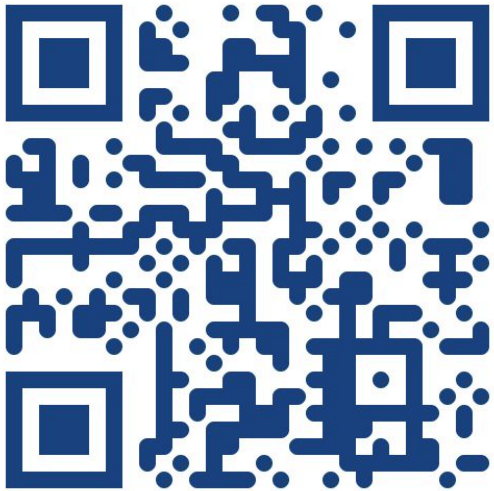
Algorithms and Probability

Exercise Session 12



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

Minitest

Time limit: 10 minutes

Number of questions: 10

Threshold to get 1 point: 6 correct answers

Threshold to get 2 points: 8 correct answers

https://moodle-app2.let.ethz.ch/mod/quiz/view.php?id=1056642

PASSWORD:

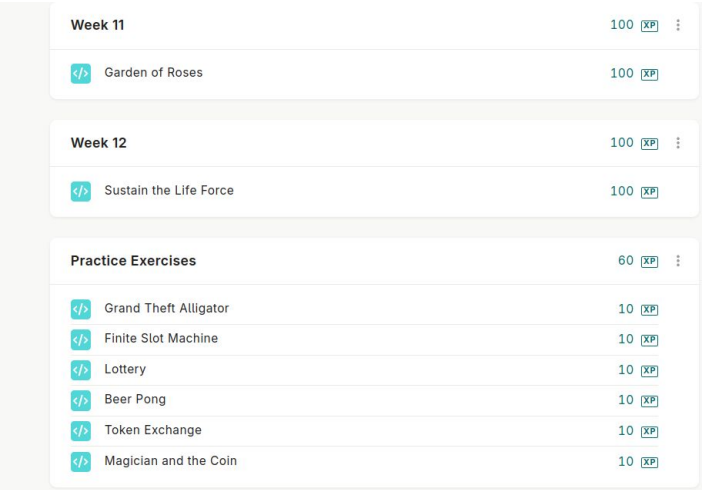
****

PASSWORD:

outplayed

Previous Exams will be added in Moodle in June

Exam Format same as last years



Polymensa

Polymensa produces a large variety of different lunch menu items. Unfortunately, Polymensa can only produce its foods in limited quantities, so Polymensa often runs out of popular items, making students sad.

To minimize sadness, Polymensa implements a sophisticated lunch-ordering system. Students text in their acceptable choices before lunch time. Then Polymensa uses an algorithm to preassign lunches to students. Students who do not get one of their choices should receive a **$10 voucher.**

Polymensa would like to **minimize the number of vouchers** they give out. Give an efficient algorithm for Polymensa to assign lunches to students.

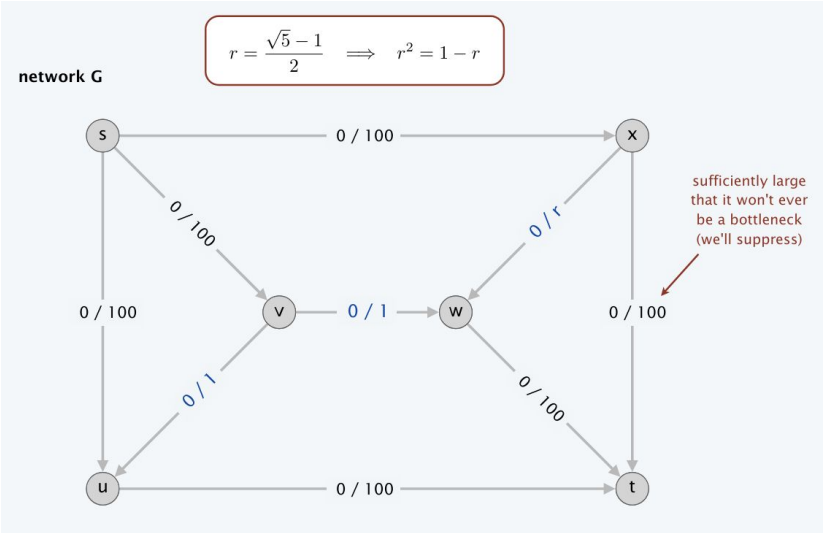
In general, suppose that, on a given day, Polymensa has produced **m types of food items b_1, . . . , b_m**, and the **quantity of each type of food item b_j is exactly q_j** .

Suppose that **n students s_1, . . . , s_n** text in their preferences, where each student **s_i submits a set S_i** of one or more acceptable lunch choices. The algorithm should assign each student either one of his/her choices or a $10 voucher. It should minimize the number of vouchers.

- Food assignment arising from flow f satisfies constraints
  - For each student s_i, the edge (s_i, b_j) describes food item b_j being assigned to s_i.
  - Ingoing and outgoing flow is at most 1 for each s_i
  - Food b_j cannot be assigned to more than q_j students cus of outgoing flow which is <= q_j.
- Food assignment satisfying the constraints corresponds directly to a flow through the network
  - Flow 1 is assigned to edge (s_i, b_j) iff student s_i gets food b_j.
  - Flow from source and flow to target node results in flow conservation being achieved.
- Max flow through this network satisfies the maximum number of students, because the definition of a max flow says that it maximizes the total flow out of source node (which corresponds to the number of students satisfied)

Ford Fulkerson with Irrational Capacities

https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/07DemoFordFulkersonPathological.pdf



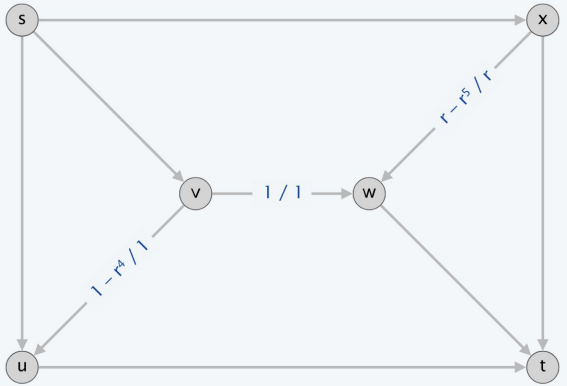
Ford Fulkerson with Irrational Capacities

https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/07DemoFordFulkersonPathological.pdf

after augmenting path 1: $1 - r^0, 1, r - r^1$ (flow = 1)

after augmenting path 5: $1 - r^2, 1, r - r^3$ (flow = $1 + 2r + 2r^2$)

after augmenting path 9: $1 - r^4, 1, r - r^5$ (flow = $1 + 2r + 2r^2 + 2r^3 + 2r^4$)



Theorem. The Ford-Fulkerson algorithm may not terminate; moreover, it may converge a value not equal to the value of the maximum flow.

Pf.

- Using the given sequence of augmenting paths, after $(1 + 4k)^{th}$ such path, the value of the flow

$$\begin{aligned} &= 1 + 2\sum_{i=1}^{k} r^i \\ &\le 1 + 2\sum_{i=1}^{\infty} r^i \\ &= 3 + 2r \\ &< 5 \end{aligned}$$

$$r = \frac{\sqrt{5} - 1}{2}$$

- Value of maximum flow = $200 + 1$. ▪

Ford Fulkerson with Irrational Capacities Demo

But is it really a concern?

Ford Fulkerson with Irrational Capacities Demo

But is it really a concern?

Computers can’t represent anything but (small) integers or (dyadic) rationals exactly.

Task

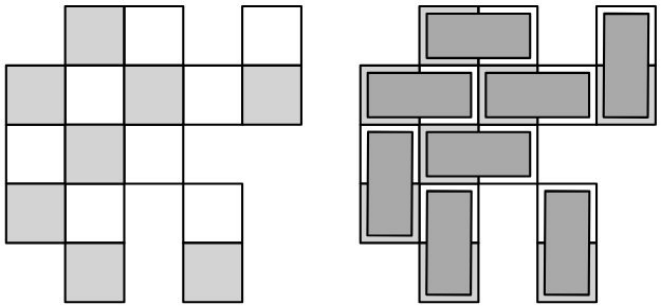
Let G = (V, E), be an unweighted, undirected graph and u, v, w $\in$ V . Give an efficient algorithm that determines whether there is a walk from u to w that passes through v such that no edge of G is traversed more than once.

Let v correspond to the source and create a new sink node t that has an edge to both u and w then solve the max flow problem. If and only if the flow has size at least 2 there is an edge disjoint uw-path via v. If the flow is at least 2, then such a path is obtained by concatenating the edge disjoint two flow paths from v to t leaving out the last edge to t. Else, there is a cut of size at most 1 that separates v from u and w (by the max-flow min-cut theorem). As any uw-path via v, has to cross that cut twice, no such path can be edge disjoint.

Task-2

Consider an incomplete $n \times n$ checkerboard, i.e., where some tiles are cut out. The incomplete checkerboard is given by an $n \times n$ array C with $C[i][j] = 0$ if the tile at position $(i, j) \in \{0, \ldots, n–1\}^2$ has been cut out, else $C[i][j] = 1$. We want to answer the question whether we can place domino pieces, each of which covers *exactly* two adjacent tiles on the checkerboard, such that all tiles are covered (for instance in the example below the answer is yes). More precisely, we want to cover every tile (i, j) with $C[i][j] = 1$ with some domino piece, such that domino pieces *do not overlap* and *only cover existing tiles* ($C[i][j] = 1$ with $i, j \in \{0, \ldots, n-1\}^2$). Give an efficient algorithm that answers this question and argue why it is correct. *(7 Points)*

Remark: You may use algorithms from the lecture as black-box.



The problem corresponds to finding a perfect matching in a bipartite graph, where the nodes of the bipartition correspond to the white and black tiles of the checkerboard respectively with an edge between horizontally or vertically adjacent tiles. The incomplete checkerboard can be covered if and only if there is a perfect matching. We have seen in the lecture that the question whether a bipartite graph has a perfect matching can be answered efficiently by transforming it into a flow problem and using the Ford-Fulkerson algorithm.

In Class Exercise

https://n.ethz.ch/~ahmala/anw/material/In_Class_Exercise_12.pdf

Aufgabe 1 – *Arbeitsgruppen*

Zu einem Kongress werden Mitarbeiter von m Firmen gesandt. Dabei sendet die i-te Firma c_i viele Mitarbeiter. Während des Kongresses sollen die anwesenden Personen in bis zu r verschiedene Arbeitsgruppen mit jeweils höchstens 5 Mitgliedern aufgeteilt werden, wobei keine zwei Mitarbeiter derselben Firma in derselben Arbeitsgruppe sein sollen.

Wir sollen eine solche Aufteilung mit Hilfe von Fluss-Algorithmen finden.

(a) Definieren Sie dazu ein geeignetes Netzwerk $N = (V, A, c, s, t)$ und zeigen Sie, dass es genau dann eine mögliche Aufteilung wie oben gibt, wenn maxflow(N) eine gewisse (welche?) Eigenschaft hat.

(b) Angenommen, wir erhalten als Eingabe die Zahlen $m, r, c_1, \ldots, c_m$ sowie die Listen $L_1, \ldots, L_m$ mit den Namen aller Teilnehmer des Kongresses. Wir können davon ausgehen, dass keine zwei Teilnehmer den gleichen Namen haben. Geben Sie einen Algorithmus an, der dies als Eingabe verwendet, und als Ausgabe entweder r Listen, $G_1, ..., G_r$ gibt, wobei G_i die Namen aller Teilnehmer der i:ten Gruppe in einer gültigen Zuordnung enthält, oder antwortet „Zuordnung nicht möglich“.

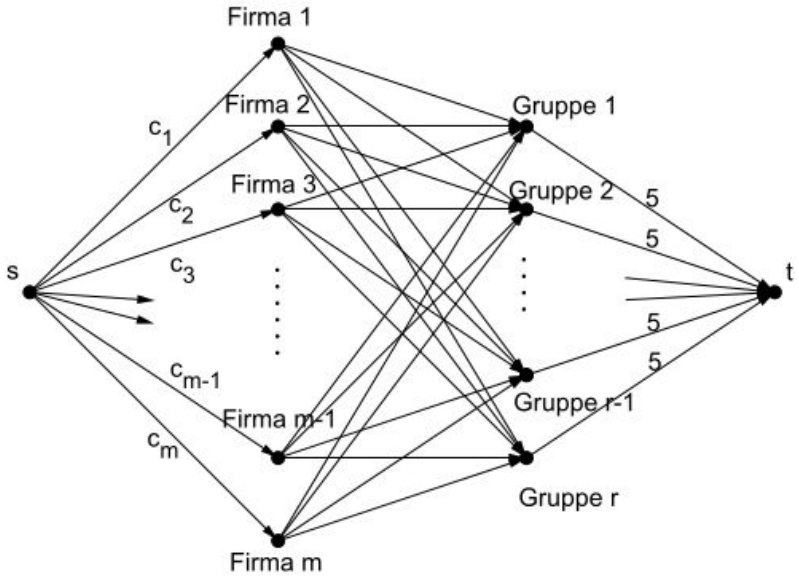


Abbildung 1: Die unbeschrifteten Kanten haben Kapazität 1.

Maximum Flow in a Dynamic Network

In this problem, you will design an algorithm that takes the following inputs:

- A flow network $F = (G, c)$, where $G = (V, E)$ is a graph with source vertex s and target vertex t, and c is a capacity function mapping each directed edge of G to a nonnegative integer;
- A maximum flow f for F; and
- A triple (u, v, r), where u and v are vertices of G and r is a nonnegative integer $\neq c(u, v)$.

The algorithm should produce a maximum flow for flow network $F' = (G, c')$, where c' is identical to c except that $c'(u, v) = r$. The algorithm should run in time $O(k \cdot (V + E))$, where $|c(u, v) - r| = k$. The algorithm should behave differently depending on whether $r > c(u, v)$ or $r < c(u, v)$.

General Results About Flow Networks

- Increasing the capacity of a single edge (u, v) by a positive integer k can result in an increase of at most k in the max flow
- Decreasing the capacity of a single edge (u, v) by a positive integer k can result in a decrease of at most k in the max flow

Case 1: r > c(u, v)

In this problem, you will design an algorithm that takes the following inputs:

- A flow network $F = (G, c)$, where $G = (V, E)$ is a graph with source vertex s and target vertex t, and c is a capacity function mapping each directed edge of G to a nonnegative integer;
- A maximum flow f for F; and
- A triple (u, v, r), where u and v are vertices of G and r is a nonnegative integer $\neq c(u, v)$.

The algorithm should produce a maximum flow for flow network $F' = (G, c')$, where c' is identical to c except that $c'(u, v) = r$. The algorithm should run in time $O(k \cdot (V + E))$, where $|c(u, v) - r| = k$. The algorithm should behave differently depending on whether $r > c(u, v)$ or $r < c(u, v)$.

Case 2: r < c(u, v)

In this problem, you will design an algorithm that takes the following inputs:

- A flow network $F = (G, c)$, where $G = (V, E)$ is a graph with source vertex s and target vertex t, and c is a capacity function mapping each directed edge of G to a nonnegative integer;
- A maximum flow f for F; and
- A triple (u, v, r), where u and v are vertices of G and r is a nonnegative integer $\neq c(u, v)$.

The algorithm should produce a maximum flow for flow network $F' = (G, c')$, where c' is identical to c except that $c'(u, v) = r$. The algorithm should run in time $O(k \cdot (V + E))$, where $|c(u, v) - r| = k$. The algorithm should behave differently depending on whether $r > c(u, v)$ or $r < c(u, v)$.