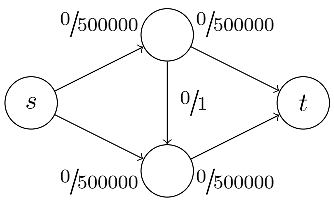
3 Performance of the Ford-Fulkerson algorithm (p. 26)

3.1 Flow network problems (p. 28)

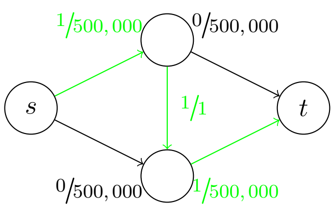
1. Give an example of a small flow network $G = (V, E)$ with integer capacities such that $1,000,000$ or more steps might be required to obtain a maximum flow f.

Solution: Initial:

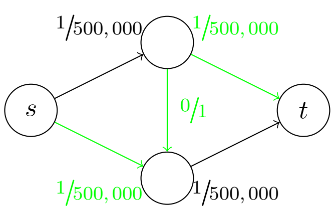


In each step, we send a flow of value 1 along the path with capacity 1 (displayed in green), using skew symmetry in the even steps.

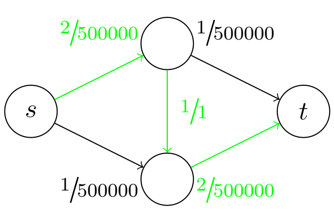
Step 1



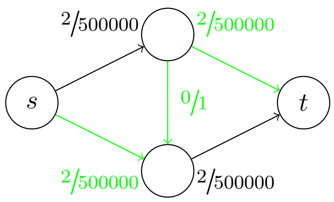
Step 2



Step 3

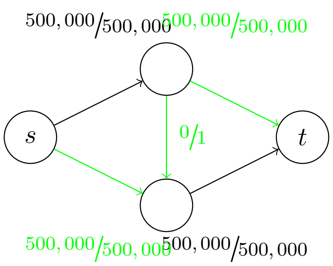


Step 4



...

<u>Step 1,000,000</u>



2. Develop a flow network to check whether there exists a $n \times m$ matrix with its entries either equal to 0 or 1 such that the i-th row sum equals r_i, for $i = 1, \dots, n$, the i-th column sum c_i, for $i = 1, \dots, m$ (with $\sum_{i=1}^{n} r_i = \sum_{i=1}^{m} c_i$).

   Solution: We construct a flow network $G = (V, E, c)$. Set $V = \{R_1, \dots, R_n, C_1, \dots, C_m, s, t\}$, $E = \{(R_i, C_j), (s, R_i), (C_j, t) | i = 1, \dots, n; j = 1, \dots, m\}$, $c(R_i, C_j) = 1$, $c(s, R_i) = r_i$ and $c(C_j, t) = c_j$. We then try to find a max-flow f from s to t in G. The value of $f(R_i, C_j)$ corresponds to whether or not there should be a 1 in the (i, j)-th entry of the matrix. Then, a matrix with the desired properties exists if and only if $|f| = \sum_{i=1}^{n} r_i$.

3. A binary matrix M is rearrangeable if and only if an arbitrary number of switches of its rows and columns can set all the diagonal entries equal to 1. ☆

   (a) Give an efficient algorithm to determine whether a matrix A is rearrangeable and discuss its time complexity.

   Solution: *Hint: you can use the previous exercice, but with different capacities.*

   (b) Give an example of a matrix A with at least one 1 on each row and column that is not rearrangeable.

4. A software house has to handle 3 projects, P1, P2, P3, over the next 4 months. P1 can only start in month 2, and must be completed after 3 months. P2 and P3 can only start at month 1, and must be completed, respectively, within 4 and 2 months. The projects require, respectively, 8, 10, and 12 man-months. For each month, 8 engineers are available. Due to the internal structure of the company, at most 6 engineers can work, at the same time, on the same project. Describe how to reduce this problem to the problem of finding a maximum flow on an appropriate graph. [Hint: Find a flow with $|f| = 30$.] ☆

Solution: We define $G = (V, E, c)$, where $V = \{s, m_1, m_2, m_3, m_4, p_1, p_2, p_3, t\}$. The vertices m_i's correspond to months and p_j's to the projects. We have edges from s to m_i for $i = 1, 2, 3, 4$ of capacity 8 (eight engineers per month), we have edges from p_1, p_2, p_3 to t of capacities 8,10,12 respectively (man-months per project) and we have edges from m_1 to p_2, p_3, from m_2 to p_1, p_2, p_3, from m_3 to p_1, p_2 and from m_4 to p_1, p_2 all of capacity equal to 6 (maximum 6 engineers per project).

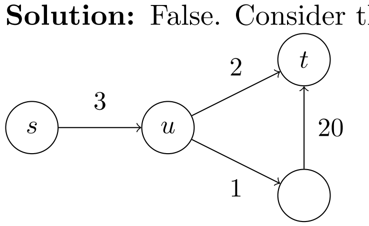
5. Consider a directed graph G and some functions $w : V \to \mathbb{R}^+$, $b : V \to \mathbb{R}^+$ and $p : E \to \mathbb{R}^+$. Let $c : V \to \{black, white\}$ be a 2-coloring of the vertices of G. Indicate how we can find a coloring that minimizes ☆

$$\sum_{v \in V, c(v) = white} w(v) + \sum_{v \in V, c(v) = black} b(v) + \sum_{(v,u) \in E, c(v) \neq c(u)} p((v, u)).$$

Solution: *Hint: write this as a flow network problem and use a min-cut to set the color of vertices.*

6. True or False? A flow network has a unique minimum cut (S, T) if there are no two edges with the same capacity. ☆

Solution: False. Consider the following graph



The minimum cut is not unique: $(\{s\}, V \setminus \{s\})$ and $(\{s, u\}, V \setminus \{s, u\})$ are minimum cuts.

7. An edge e is upwards critical if increasing $c(e)$ increases the max flow and downwards critical if decreasing $c(e)$ decreases the max flow. True of False? ☆

(a) Every flow network with $|f| > 0$ contains at least one upwards critical edge.

> **Solution:** False. Consider the graph
>
> $$s \xrightarrow{1} u \xrightarrow{1} t$$
>
> Clearly, increasing (s, u) or (u, t) doesn't increase the max flow.

(b) Every flow network with $|f| > 0$ contains at least one downwards critical edge.

> **Solution:** True. For every min-cut (S, T), every edge from S to T is downward critical due to the third point of Max-flow Min-cut theorem.

4 The Edmonds-Karp algorithm (p. 29)

4.1 Flow problems (p. 30)

1. Show that a maximum flow in a network can always be found via $|E|$ augmenting paths (Hint: determine the paths after finding the maximum flow).

> **Solution:** First, find the maximum flow in a random flow network problem and highlight the possible paths.
>
> We can decompose any feasible flow f on G into at most $|E|$ cycles and $s \to t$ paths, as follows:
>
> (a) Find a (simple) $s \to t$ path p with positive (net) flow. (If $|f| > 0$, at least one such path has to exist.)
>
> (b) Anti-augment the flow on p until an edge on p has zero flow. Let k be the amount of flow by which we anti-augmented.
>
> (c) Add $(p; k)$ as an item of the flow decomposition. Repeat above 3 steps until no more paths can be found. At this point we're done, however for completeness' sake we go through steps (d) - (f).
>
> (d) At this point, there can still be edges with positive net flow, belonging to cycles in the graph. We find such a cycle c with positive (net) flow, by first finding any edge e with $f(e) > 0$ and following outgoing positive flow edges until we detect a cycle (by encountering e again).
>
> (e) Anti-augment the flow on c until an edge on c has zero flow. Let k be the amount of flow by which we anti-augmented.
>
> (f) Add $(c; k)$ as an item of the flow decomposition. Repeat above 3 steps until no more cycles can be found.

Each time we anti-augment, we reduce the flow on at least one (*critical*) edge to 0. Thus we can only anti-augment at most $|E|$ times, we can only add up to $|E|$ $s \to t$ paths and cycles to the flow decomposition. As such, there are at most $|E|$ augmenting paths. Note further that after every anti-augmentation f keeps being a flow. In fact, we've proven what is called "Flow decomposition theorem":

Every flow can be decomposed in at most $|E|$ augmenting paths and cycles.

2. Let f be a flow in $G = (V, E)$, f^* a maximum flow and $G_f(\Delta)$ the residual network of G for the flow f with all edges (u, v) with $c_f(u, v) < \Delta$ removed. Show that $|f^*| \le |f| + \Delta|E|$ if there is no path from s to t in $G_f(\Delta)$. [Hint: Consider a particular cut (S', T') and note that $|f^*| \le c(S, T)$ for any cut (S, T).] ☆

Solution: Let S consist of all the nodes reachable from s in $G_f(\Delta)$. Assume there is no path from s to t in $G_f(\Delta)$, then $(S, V \setminus S)$ is a cut. We also have $c_f(S, V \setminus S) \le \Delta|E|$, as there are at most $|E|$ edges from S to $V \setminus S$ in the residual network G_f, each of residual capacity $\le \Delta$. The value of maximum flow in G_f is $|f^*| - |f|$ due to Lemma 2.1. As $(S, V \setminus S)$ is a cut, we must have in G_f that $|f^*| - |f| \le c_f(S, V \setminus S) \le \Delta|E|$, which gives what we needed to show.

3. Using the notation of the previous exercise, assume $c(e) \in \{1, 2, \ldots, C\}$ for $e \in E$. Consider the Capacity Scaling Algorithm:

   $f = 0$; $\Delta = 2^{\lfloor \log_2 C \rfloor}$
   while $\Delta \ge 1$ **do**
       while There exists a path p from s to t in $G_f(\Delta)$ **do**
           augment f with f_p
       end while
       $\Delta = \Delta/2$
   end while

   (a) Does this algorithm end after a finite number of iterations and does it yield a maximum flow?

   Solution: *Hint: Ford-Fulkerson*

   (b) Give an upper bound on the number of times that the outer and inner while loop is executed (use the previous exercise).

   (c) What is the overall time complexity?