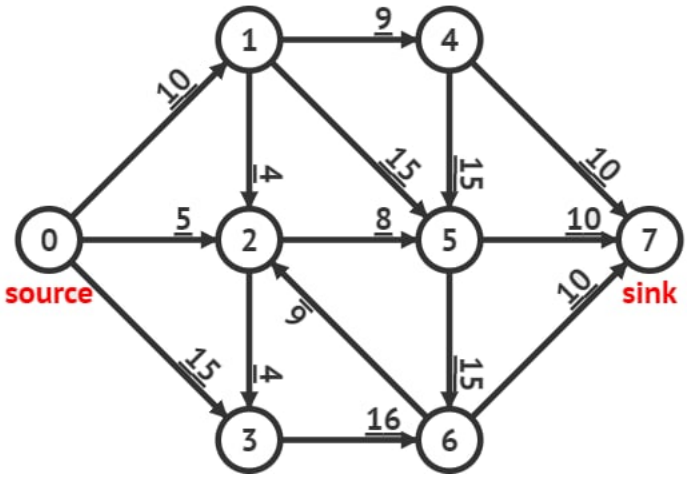
CS3230: Assignment for Week 9

Due: Monday, 17th Oct 2022, 7 pm SGT

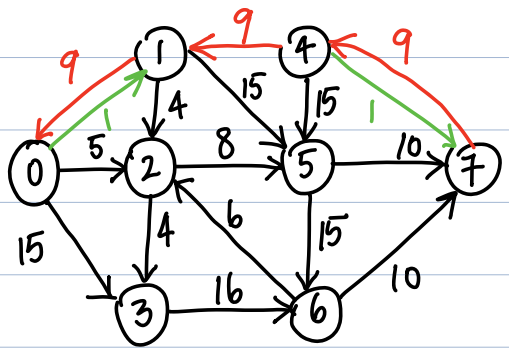
Please upload a PDF file containing your solution (hand-written & scanned, or typed) by 17th Oct, 7 pm on Canvas. You may discuss the problems with your classmates, though you should write up your solutions on your own. Please note the names of your collaborators in your submission. You may want to refer to the plagiarism policy from Lecture 0.

1. Consider the graph below

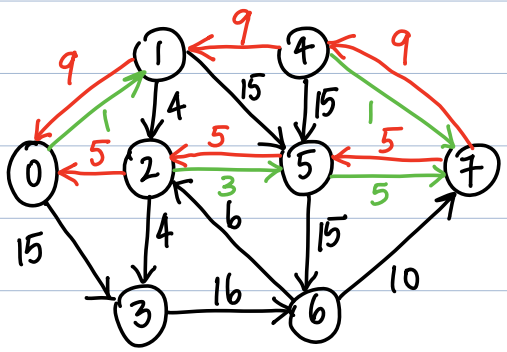


Run the Ford-Fulkerson algorithm to find the maximum flow from the node 0 to node 7. Write the process step by step, and deduce the final maxflow value.

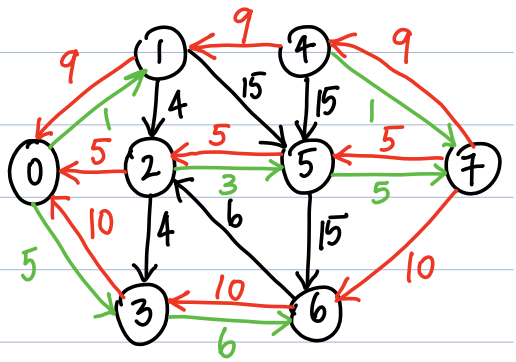
1. Consider the path 0-1-4-7. Then we have residual capacities:



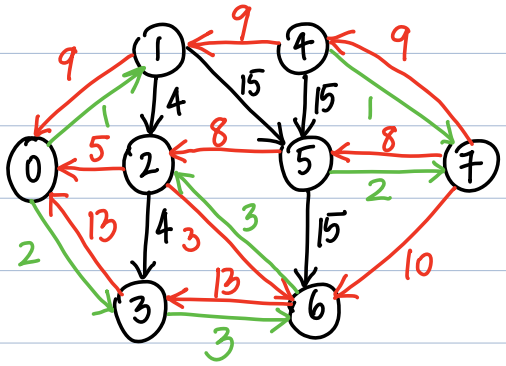
0-2-5-7



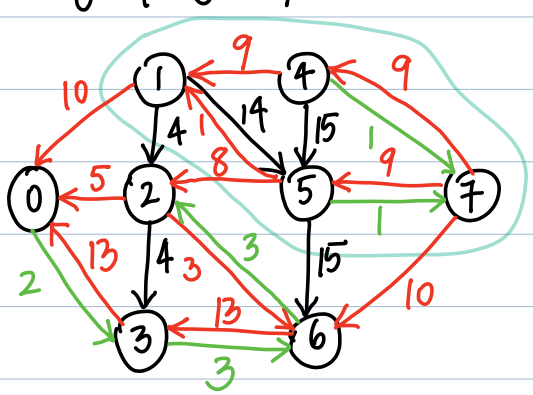
0-3-6-7



0-3-6-2-5-7



0-1-5-7



Max Flow: $9+8+10+1=28$

2. Recall the men and women matchmaking problem that was discussed in Lecture 1, with n men and n women. Now, instead of a preference ranking, man i provides a set of acceptable partners $M_i \subseteq [n]$, and similarly woman i provides $W_i \subseteq [n]$. Additionally, there are also n houses that a pair can stay in, and woman i has a set of acceptable houses $H_i \subseteq [n]$.

   A matching between a man, a woman, and a house is acceptable if both people have each other in their set of acceptable partners, and the house is in the woman's set of acceptable houses. Formally, a matching $[m, w, h]$ is acceptable if $w \in M_m, m \in W_w$, and $h \in H_w$.

   You want to find N, the maximum number of acceptable matchings $[m, w, h]$ such that each person/house is in at most 1 matching. You are given the function $findMaxFlow(G, s, t)$, which takes in a directed graph G with flow capacities on each edge, a source node s and a sink node t, and outputs the value of the maximum flow from s to t in G. Note that $findMaxFlow$ uses the Edmonds-Karp algorithm and has corresponding runtime.

   (a) Using $findMaxFlow$ in a non-trivial way, design an algorithm to find N and analyze the runtime of your algorithm. You do not need to formally prove correctness.

   (b) Suppose $findMaxFlow$ uses the Ford-Fulkerson algorithm instead. Is this faster? What is the runtime of your algorithm with this?

3. The Ice Cream Factory is gearing up for the holiday season, and after several years with mediocre sales, they have decided to focus only on their most popular product, ice cream with flavour of chocolate and vanilla. In order to make a perfect such ice cream, it is important that the mixing machine receives equal amounts of vanilla and chocolate ice cream.

   There are two separate creaming machines in the factory which produce respectively chocolate and vanilla ice cream, and the resulting creamy goodness is stored in two separate tanks. From there, it can be transported to the mixing machine through pipes, however the dimension of a pipe gives an upper bound for how much ice cream can pass through it each minute. These pipes meet in welding points, where streams of ice cream go from one pipe to another. Streams can also merge or split into other streams at such points, if more than two pipes meet here. It is *not* important to keep the flavors separate during the transportation, since they will eventually be mixed anyways.

   Given a map of the pipe system (a weighted graph where the vertices are the welding points, edges are the pipes, their weights are the respective capacities of the pipes, and the tanks and the mixing machine are special vertices), design an algorithm to decide how many liters of ice cream the factory can produce each minute. Analyze its runtime. You may assume there are at most P pipes and capacity of each pipe will be less than X. Also, you can assume you have Dinic's algorithm which runs in $O(V^2E)$ for max flow as a library code.

2(a) Let G be a graph with vertices $V := \{s, d\} \cup \{m_i, w_i, h_i \mid i \in \mathbb{Z}^+, i \leq n\}$ and let directed unit weighted edges be

$$E := \{(s, m_i) \mid i \in \mathbb{Z}^+, i \leq n\}$$
$$\cup \{(m_i, w_j) \mid i, j \in \mathbb{Z}^+, i, j \leq n, m_i \in M_j, w_j \in W_i\}$$
$$\cup \{(w_i, h_j) \mid i, j \in \mathbb{Z}^+, i, j \leq n, h_j \in H_i\}$$
$$\cup \{(h_i, d) \mid i \in \mathbb{Z}^+, i \leq n\}$$

then note that any path $s - m_i - w_j - h_k - d$ corresponds to a valid matching $[m_i, w_j, h_k]$ by construction of E. As edges are unit-weighted, we have that maximum flow in G corresponds to the maximal acceptable matchings so we are done.

Runtime Analysis:

Construction of G: $O(n)$ vertices, $O(n^2)$ edges $\Rightarrow O(n^2)$.

Edmonds-Karp: F iterations of BFS, which is $O(E)$,

where $F :=$ maximum flow $\leq n$. $O(n^3)$.

(b) Same runtime: $O(EF) = O(n^3)$

3. Let $G(V,E)$ be the graph of the map with capacity function $C: E \to \mathbb{Z}^+$ by assumption. Let s be a new vertex and let s_c and s_v be the vertices corresponding to the chocolate and vanilla tanks respectively.

Define $V' := V \cup \{s\}$, $E' = E \cup \{(s,s_c),(s,s_v)\}$)
Define $G' := (V', E')$ and define $C_k : E' \to \frac{1}{2}\mathbb{Z}^+ \cup \infty$ by

$$C_k((u,v)) = \begin{cases} k & \text{if } u=s \text{ and } v=s_c \text{ or } s_v \\ c((u,v)) & \text{otherwise} \end{cases}$$

so that $\text{Danic}(G_k, C_\infty) =: F$ is the maximum production of ice cream unconditional on the ratio of chocolate and vanilla.

Algorithm

```
maxIceCream (Graph G, CapacityFunction C)
    int max = Danic(G', C∞)
    int min = 0
    while max-min > 1 :
        mid = ⌊(max+min)/2⌋, k = min/2
        flow = Danic(G', Ck)
        if flow == 2k:
            min = flow
        else:
            max = flow
    return min
```

Runtime: $O(\log F \cdot V^2E) = O(\log(PX) \cdot V^2E)$ where F is the max flow since we iterate at most $\log F$ times, each with cost $O(V^2E)$.

Proof of Correctness

We desire the following invariants:

① $\exists$ some valid configuration for a flow of $2(\text{min})$ satisfying the ratio constraint.

② $2 \times \text{max}$ is an upper bound for $\text{maxIceCream}(G, C)$.

Initialization

Clear that $\text{Danic}(G', C_\infty)$ yields an upper bound and 0 is a lower bound.

Maintenance.

Note that clearly $\text{min} < \text{mid} < \text{max}$

Observe that a flow obtained with flow $= 2k = \text{mid}$ must fully utilize (s, s_c) and (s, s_v), both of which have capacity mid. Hence ① is satisfied.

Clearly we cannot have flow $> 2k$ so flow $< 2k$, in which case (s, s_c) and (s, s_v) are not both fully utilised, so that ② is satisfied

Furthermore note that after each iteration, max-min halves, so this algorithm will terminate.

Termination.

Upon termination, we have that $\max - \min \leq 1$. Note that $k \in \frac{1}{2}\mathbb{Z}$ so $\min, \max \in \mathbb{Z}$. We now show that $\text{maxIceCream}(G, c) = \min$.

For brevity denote $\text{maxIceCream}(G, c) =: M$. From ① and ② we have that $\max > M \geq \min$.

First note that when we set $c(s, s_c) = c(s, s_v) = \frac{\max}{2}$, (s, s_c) and (s, s_v) are not both fully utilised so there exists some cut with capacity $\gamma < \max$. If the cut excludes (s, s_v) and (s, s_c) then $\gamma \in \mathbb{Z}$ so $\gamma \leq \min$.

If $M > \min$ then by maximality of M, the minimum cut(s) has capacity M. But $M \in \mathbb{Z}$ and all edge capacities excluding (s, s_c) and (s, s_v) are integers, so the minimum cut must be $\{\{s\}, V \setminus \{s\}\}$.

If the cut in * excludes (s, s_v) and (s, s_c) then $\gamma \leq \min < M \leq \gamma$, a contradiction. WLOG if the cut includes (s, s_v) then the same cut has capacity $:= \gamma' < \gamma$. Write $M = \min + \delta$ so $\gamma' = q + \frac{\delta}{2}$ for some $q \in \mathbb{Z}$, but $q + \frac{\delta}{2} < \min + \delta$ because $q \leq \min$ so $\gamma' < M \leq \gamma'$, a contradiction.

Hence $\text{maxIceCream}(G, c) = \min$.