Algorithms and Probability (FS2025)
Week 11

Rui Zhang

May 7, 2025

Contents

1 Mini Quiz, Exercises, Corrections 1

2 Content 1

2.1 Finding Arbitrarily Long Paths 1

2.2 Copium: Finding Kinda, Sorta Long Paths 2

2.3 Flow 7

1 Mini Quiz, Exercises, Corrections

?????????????????????????

2 Content

2.1 Finding Arbitrarily Long Paths

Definition: Long Paths: Given a graph G and a number B, find out if there exists a path of length at least B in G.

Theorem: If we can solve the long paths problem for graphs on n vertices in $t(n)$ time, then we can solve the hamiltonian cycle problem in $t(2n-2) + O(n^2)$ time

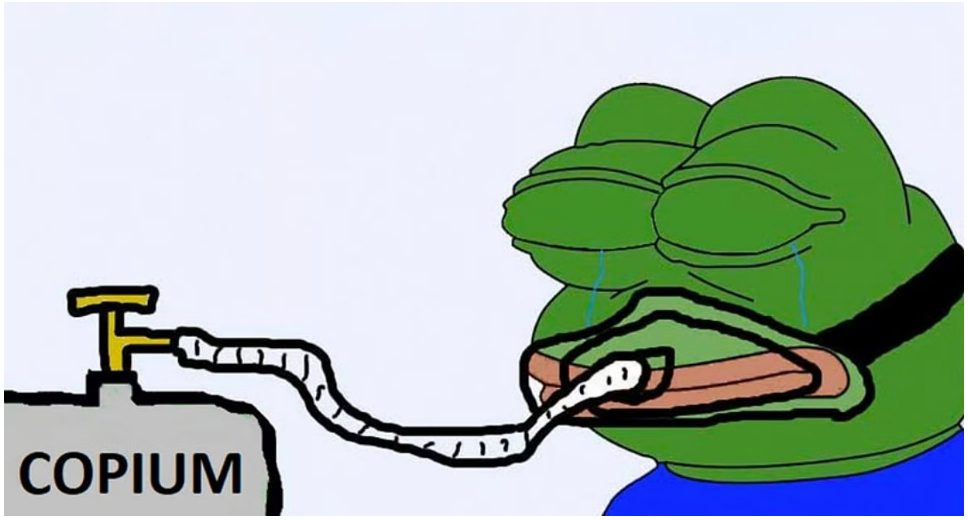


Figure 1: ambitious CS students realizing that we literally cannot have anything good in computer science because of the P = NP Problem.

2.2 Copium: Finding Kinda, Sorta Long Paths

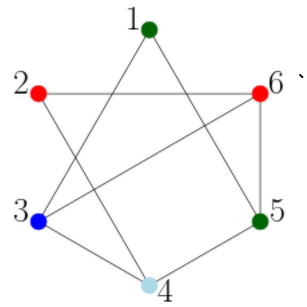
After learning that we likely won't come up with an efficient solution to Long Paths, we go through the 5 stages of grief. Instead, we look at the following:

> **Definition:** Kinda, Sorta Long Paths: Given a graph G find out if there exists a path of length at least $\log n$ in G.

Our idea is the following: We will develop a probabilistic algorithm which does multiple iterations of the following:

1. color the graph randomly (not the colorings we learnt)
2. Find all colorful paths (save their colors)
3. If one is of length $\log n$, return "true", otherwise repeat
4. return false

Quiz / Example:



Which of the following are colorful paths:

- 2,4,5 ✓
- 4,2,6 ✗
- 1,5,4 ✗
- 3,6,5 ✓
- 1,2,3 ✗
- 1,3,6 ✓
- 4,2,5 ✗
- 5,4,2 ✓
- 2,3,4 ✗
- 3,5,6 ✗

First, let's find an algorithm for finding colorful paths in a colored graph. Let us fix a number k. Given a graph $G = (V, E)$ and a coloring $\gamma : V \to [k]$ we want to find colorful paths and determine in particular if there exists a path of length $k-1$ (edges). To this end, we will use the following DP algorithm (yay, your favorite.)

> **Definition:** DP-Definition: Let $P_i(v)$ be our DP entry defined as:
>
> $$P_i(v) := \left\{ S \in \binom{[k]}{i+1} \mid \exists \text{ a colorful path which ends in } v \text{ and traverses exactly the colors in } S \right\}$$

And we define our base cases and recursion as:

Top Secret

Me

Figure 2: Pseudocode for the BUNT algorithm

Definition:

- Base Case:

  $\forall v \in V \quad P_0(v) = \{\{\gamma(v)\}\}$ (one single vertex)

- Recursion:

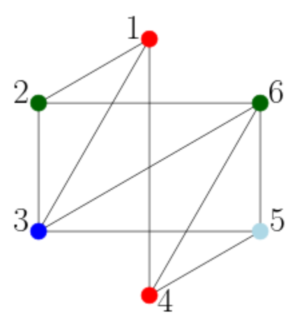
  $P_i(v) = \bigcup_{x \in N(v)} \{R \cup \{\gamma(v)\} \mid R \in P_{i-1}(x) \text{ and } \gamma(v) \notin R\}$

- Solution Extraction:

  $\bigcup_{v \in V} P_{k-1}(v) \stackrel{?}{=} \emptyset$

  .

<u>Quiz / Example</u>:



What is $P_2(1)$ in this graph with these colors?

- $P_2(1) = \{\{b, g, r\}, \{g, g, r\}, \{b, c, r\}, \{c, r, r\}\}$ ×
- $P_2(1) = \{(r, g, b), (r, g, g), (r, b, g), (r, b, c), (r, r, c), (r, r, g)\}$ ×
- $P_2(1) = \{\{b, g, r\}, \{b, c, r\}\}$ ✓

- $P_2(1) = \{(r, g, b), (r, b, g), (r, b, c)\}$

<u>Runtime Analysis "BUNT":</u>

Fix i and a vertex v. Then

- go through all neighbors $\deg(v)$
- check all $R \in P_{i-1}(X)$ for all neighbors $|P_{i-1}(X)| \leq \binom{k}{i-1+1} \overset{?}{=} \binom{k}{i}$
- check all $v \in R$ for all vertices for checking $\gamma(v) \in R$ $\quad |R| = i$

$\hookrightarrow O\left(\binom{k}{i} \deg(v)\, i\right)$

<u>Over all vertices:</u>

$$O\Big(\underbrace{\sum_{v \in V} \deg(v)}_{2m} \binom{k}{i}\, i\Big) = O\left(\binom{k}{i}\, i\, m\right)$$

Theorem: The runtime of "BUNT" is $O\left(\binom{k}{i} \cdot i \cdot m\right)$ and the total runtime of the algorithm is

$$O\left(\sum_{i=1}^{k-1} \binom{k}{i} \cdot i \cdot m\right) \leq O\left(2^k km\right)$$

where we used the equality $\sum_{i=0}^{k} \binom{k}{i} = 2^n$

We are still not done, we proposed this algorithm with the idea in mind that we color the graph somehow and run this previous algorithm. Thus, our idea is the following: we assign colors randomly and independently to every vertex in each round / iteration with $k = \log n + 1$ and then apply the algorithm with the same k. Each iteration will take:

- $O(n)$ time to color the vertices randomly
- $O\left(2^{\log n} (\log n) m\right)$ (dominant factor) $(k = \log n)$

Now, how often do we want to repeat this in total?

$k!$ possible ways to color paths of length k colorfully
k^k possible total ways to color

$$p_{error} \leq 1 - \frac{k!}{k^k} \leq 1 - e^{-k}$$

$e^k = \sum_{n=0}^{\infty} \frac{k^n}{n!} \overset{n=k}{\geq} \frac{k^k}{k!}$

$\Rightarrow e^{-k} \leq \frac{k^k}{k!}$

we may have more than one path

$p_{error} = 0$ for graphs without colorful paths

Applying Theorem 2.74:

$\varepsilon = e^{-k}$. We want $\delta = e^{-\lambda}$

Then $N = \varepsilon^{-1} \ln \delta^{-1} = e^k \ln e^{\lambda} = \lambda e^k$

Monte Carlo?

Los Vegas?

$$O(N \cdot 2^k km) = O(N 2^{\log n}(\log n)m) = O(\lambda (2e)^{\log n}(\log n)m)$$

Quiz / Example:

This full algorithm we have shown either returns "yes" or "no":

- Output "yes" is always correct ✓
- Output "no" is always correct ✗

The BUNT subprocedure we have shown also either returns "yes" or "no":

- Output "yes" is always correct ✓
- Output "no" is always correct ✓

Assume we now want to apply this algorithm differently. We want to find paths of length $k = 2 \cdot \log n$ instead and we will repeat the random coloring process $N = n \cdot e^k$ times What is the runtime and the error probability for this algorithm then?

log is binary log

$\lambda = n,\ k = 2\log n$

$\Rightarrow O(N \cdot 2^k km) = O(\lambda e^{2\log n} 2^{2\log n} m)$

$= O(n e^{2\log n} n^2 m)$

$= O(n^3 e^{2\log n} m)$

$= O(n^3 n^{2\log e} m)$

$= O(n^{3+2\log e} m)$

$P[\text{fail}] \leq \delta$

$N = ne^k = \varepsilon^{-1} \ln \delta^{-1} = e^k \ln(\delta^{-1})$

$\Rightarrow \ln(\delta^{-1}) = n \Rightarrow \delta = e^{-n}$

2.3 Flow

Definition: A Network is a tuple $N = (V, A, c, s, t)$, where

- (V, A) is a directed graph (where the edges are the pipes and the vertices are the intersections)
- $s \in V$ is the source (the point where water magically spawns out of)
- $t \in V \setminus \{s\}$ is the target (the point where water disappears into the abyss)
- $c : A \to \mathbb{R}_0^+$ is the capacity function (which denotes how much water can flow through each pipe)

Definition: Let $N = (V, A, c, s, t)$ be a network. A flow in N is defined as a function $f : A \to \mathbb{R}$, such that

- For all $e \in A$, $0 \leq f(e) \leq c(e)$
- For all $v \in V \setminus \{s, t\}$ we have

$$\sum_{u \in V \text{ s.t. } (u,v) \in A} f(u, v) = \sum_{u \in V \text{ s.t. } (v,u) \in A} f(v, u)$$

  which we call conservation of flow (note that again, this does not apply for the source and the target)

Definition: The value of a flow f is defined as

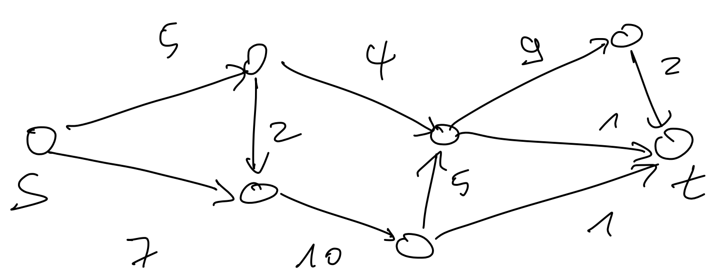
$$\text{val}(f) := \mathit{netoutflow}(s) := \sum_{v \in V \text{ s.t. } (s,v) \in A} f(s, u) - \sum_{u \in V \text{ s.t. } (u,s) \in A} f(u, s)$$

Definition: f is called an integer flow, iff $f(e) \in \mathbb{Z} \; \forall e \in A$

Lemma:

$$\text{netinflow(t)} := \sum_{u \in V \text{ s.t. } (u,t) \in A} f(u, t) - \sum_{v \in V \text{ s.t. } (t,v) \in A} f(u, s) = \text{netoutflow}(s)$$

Examples:

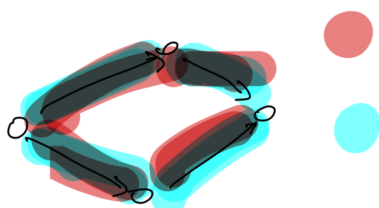


Proof. We can alternatively prove that

$$
\begin{aligned}
0 &= \operatorname{netoutflow}(s) - \operatorname{netinflow}(t) \\
&= \left(\sum_{v \in V \text{ s.t. } (s,v) \in A} f(s,u) - \sum_{u \in V \text{ s.t. } (u,s) \in A} f(u,s) \right) \\
&\quad - \left(\sum_{u \in V \text{ s.t. } (u,t) \in A} f(u,t) - \sum_{v \in V \text{ s.t. } (t,v) \in A} f(u,s) \right) \\
&= \sum_{v \in V} \left(\sum_{u \in V \text{ s.t. } (v,u) \in A} f(v,u) - \sum_{u \in V \text{ s.t. } (u,v) \in A} f(u,v) \right) && (\star) \\
&= \sum_{(v,u) \in A} f(v,u) - \sum_{(u,v) \in A} f(u,v) && (\star\star) \\
&= 0
\end{aligned}
$$

□

$\star\star$:



$\star$: $\sum_{u \in V \text{ s.t. } (v,u) \in A} f(v,u) - \sum_{u \in V \text{ s.t. } (u,v) \in A} f(u,v)$

is outflow − inflow = 0

for all $v \neq s,t$

only for $v = s,t$ do we have it.

Definition: An s-t-Cut for a network (V, A, c, s, t) is a partition (S, T) of V ($S \uplus T = V$) such that $s \in S$ and $t \in T$. The capacity of this cut is defined as

$$\operatorname{cap}(S,T) := \sum_{(u,w) \in (S \times T) \cap A} c(u,w)$$

↳ only edges going from S <u>to</u> T!

We have that

Lemma: Let f be a flow and (S, T) an s-t-Cut in a network. Then we have that

$$\operatorname{val}(f) \leq \operatorname{cap}(S,T)$$

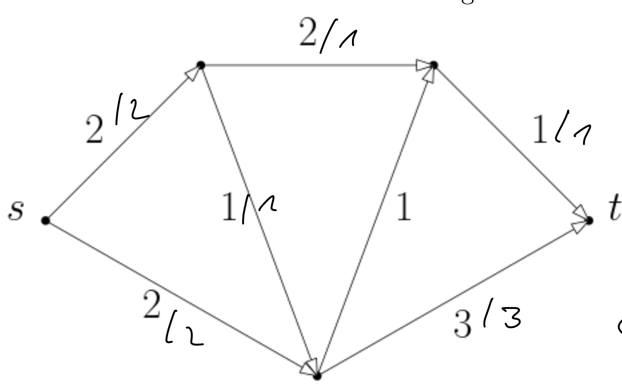
and with these definitions comes the formal version of the

Definition: (Maxflow-Mincut-Theorem) For every network (V, A, c, s, t), we have that

$$\max_{f \text{a flow in } N} \operatorname{val}(f) = \min_{(S,T)\ s\text{-}t\text{-Cut in } N} \operatorname{cap}(S, T)$$

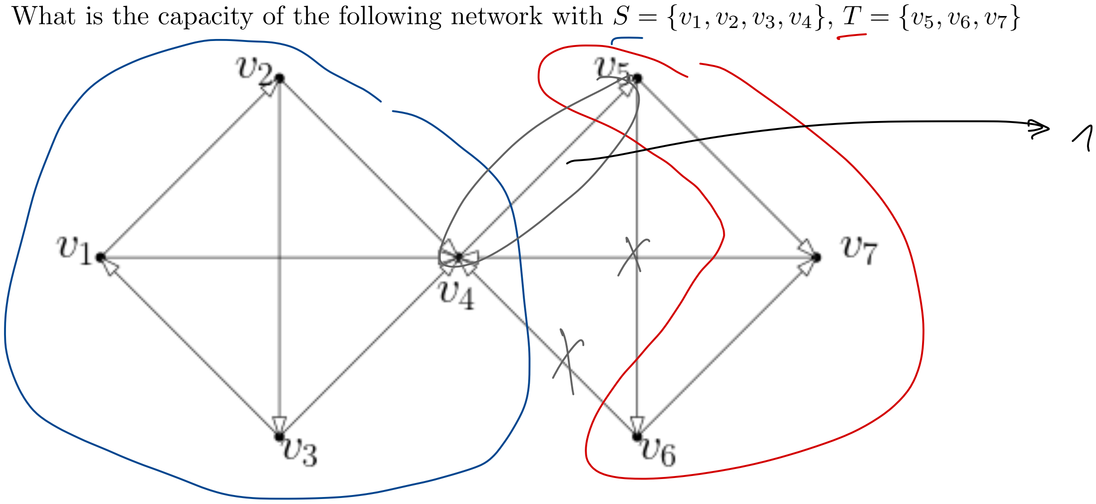
Quiz Time!

- $N = (V, A, c, s, t)$ be a network and S and T and s-t-Cut. We have that $cap(S, T) \geq 0$ ✓, since $c: A \to \mathbb{R}_0^+$
- What is the maximum flow in the following network?

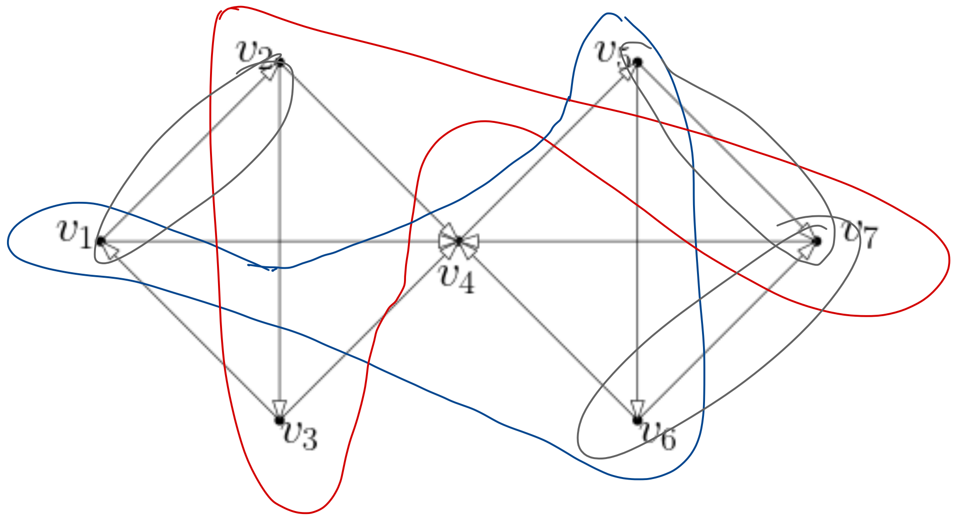


$\max\limits_{f \text{ flow}} \operatorname{val}(f) = 4$, since

- What is the capacity of the following network with $S = \{v_1, v_2, v_3, v_4\}$, $T = \{v_5, v_6, v_7\}$



- What is the capacity of the following network with $S = \{v_1, v_4, v_5, v_6\}$, $T = \{v_2, v_3, v_7\}$



3